Reg. No..................................

[12CCP304]

KARPAGAM UNIVERSITY

(Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021
(For the candidates admitted from 2012 onwards)

M.Com. DEGREE EXAMINATION, NOVEMBER 2013

Third Semester

COMMERCE (COMPUTER APPLICATIONS)

JAVA

Time: 3 hours   Maximum : 100 marks

PART – A (15 x 2 = 30 Marks)
Answer ALL the Questions

1. What is object-oriented Programming?
2. Write any two features of java?
3. What is called garbage collection?
4. Define: class.
5. Write short notes on constructor overloading
6. How to declare final class?
7. What is abstract?
8. What is the string constructor?
9. What is the usage of iterator?
10. Write about two kinds of streams?
11. How to run the applet program?
12. Mention any two difference between init () and stop ()?
13. What is an exception?
14. Define: Thread.
15. List out the user interface elements of AWT?

PART B (5 X 14= 70 Marks)
Answer ALL the Questions

16. a) Distinguish between C++ and Java.
Or
b) Describe about looping statements in Java.

17. a) What is an interface and how to implement the interface? Explain it.
Or
b) Briefly explain about Classes and Methods in Java.

18. a) Discuss about operations on string.
Or
b) Illustrate on List Implementations.

19. a) Explain the Reader and Writer classes in Java.
Or
b) What are the Graphic class in Java and explain it.

20. a) Write briefly about exception handling in Java.
Or
b) Explain about basic components in Applet.

Reg. No................................

[13CCP305]

KARPAGAM UNIVERSITY

(Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021
(For the candidates admitted from 2013 onwards)

M.Com. DEGREE EXAMINATION, NOVEMBER 2014

Third Semester

COMMERCE (COMPUTER APPLICATIONS)

JAVA

Time: 3 hours Maximum : 60 marks

PART – A (10 x 2 = 20 Marks)
Answer any TEN Questions

1. What is object-oriented Programming?
2. What is type casting?
3. How to declare the variables in java.
4. Define: class.
5. Write about a constructor?
6. Write the syntax of interface.
7. Point out importance of package in java.
8. Define: Stack.
9. Mention the utilities in java.
10. What is an applet?
11. What are the three types of entities in stream tokenizer?
12. How to define the Font class?
13. What is an exception?
14. What is single thread?
15. Define: context

PART B (5 X 8= 40 Marks)
Answer ALL the Questions

16. a) Discuss about abstract class and methods in java.
Or
b) Briefly explain about Classes and Methods in Java.

17. a) Write a program to find maximum and sum of an array.
Or
b) Describe about Packages.

18. a) Elucidate on Input/output Stream classes.
Or
b) What are the Graphic class in Java and explain it.

19. a) Write briefly about exception handling in Java.
Or
b) Explain about basic components in AWT.

20. **Compulsory : -**

Analysis between C++ and Java Programming.

Reg. No.................................

[15CCP303]

KARPAGAM UNIVERSITY

Karpagam Academy of Higher Education
(Established Under Section 3 of UGC Act 1956)
COIMBATORE – 641 021
(For the candidates admitted from 2015 onwards)

M.Com., DEGREE EXAMINATION, NOVEMBER 2016
Third Semester

COMMERCE (COMPUTER APPLICATIONS)

JAVA

Time: 3 hours   Maximum : 60 marks

PART – A (20 x 1 = 20 Marks) (30 Minutes)
(Question Nos. 1 to 20 Online Examinations)

(Part - B & C 2 ½ Hours)

PART B (5 x 6 = 30 Marks)
Answer ALL the Questions

21. a. Write short notes on Data types of Java.
Or
b. Explain in brief about the concept of handling arrays in Java.

22. a. Explain the types of Inheritance with example.
Or
b. Write a program to find whether the given number is prime or not using Constructor.

23. a. Explain different levels of Access protection in Java.
Or
b. What are Interfaces? How to define and implement Interfaces?

24. a. Explain the use of Priorities in thread with example.
Or
b. What is Synchronization? Explain it with example.

25. a. What is an applet? How do applets differ from application programs?
Or
b. Explain the methods used for drawing line and rectangles.

PART C (1 x 10 = 10 Marks)
CASE STUDY (Compulsory)

26. Write a Java application for retail stores management which wants to automate the following process
i. Booking product for customer   ii. Bill generation for customer
iii. Inventory status check and updation.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2016 onwards)

DEPARTMENT OF COMMERCE (CA)

SUBJECT : **JAVA**

SEMESTER : III

SUBJECT CODE: **16CCP304** **CLASS** : **II M.COM CA**

FIRST INTERNAL EXAMINATIONS-ANSWER KEY

PART A (20 *1=20 Marks)

1. Space
2. Classes
3. Abstraction
4. Data members
5. Methods
6. Inheritance
7. Local
8. Final class
9. = =
10. Cannot change its value
11. ?:
12. Array
13. Import
14. Constructor
15. Return type
16. Classes
17. Only in the same class
18. Java.lang
19. Java.util
20. Exception

PART B (3 *2=6 Marks)

21. (i) Byte Code

Java bytecode is the result of the compilation of a Java program, an intermediate representation of that program which is machine independent.

(ii)Java Environment

The Java bytecode gets processed by the Java virtual machine (JVM) instead of the processor. It is the job of the JVM to make the necessary resource calls to the processor in order to run the bytecode.

22. "Command Line arguments".

The command line arguments are handled using main() function arguments where **argc** refers to the number of arguments passed, and **argv[]** is a pointer array which points to each argument passed to the program.

23. Package Declaration

To make a Java package named pkg :

- Make a directory named pkg .
- Put all the .java files for the classes and interfaces in the directory pkg .
- Begin each of the .java files with a package declaration.
- Compile the files by running javac from pkg 's parent directory.

PART C(3 *8=24 Marks)

24. (a) Advantages of Object Oriented Programming

- Platform Independent
- Simple
- Robust
- Distributed
- Portable
- Dynamic
- Secure
- Performance
- Interpreted

(b) Control Structure in Java

The control statement is used to control the flow of execution of the program. This execution order depends on the supplied data values and the conditional logic. Java contains the following types of control statements:

1- Selection Statements
2- Repetition Statements
3- Branching Statements

Selection statements:

If Statement:

This is a control statement to execute a single statement or a block of code, when the given condition is true and if it is false then it skips **if** block and rest code of program is executed.

Syntax:

```
 if(conditional_expression){
 <statements>;
 ...;
 ...;
}
```

If-else Statement:

The **"if-else"** statementis an extension of if statement that provides another option when 'if' statement evaluates to "false" i.e. else block is executed if **"if"** statement is false.

Syntax:

```
  if(conditional_expression){
 <statements>;  ...;
 ...;
 }
  else{
```

```
<statements>;
....;
....;
 }
```

Switch Statement:

In a switch block there can be one or more labeled cases. The expression that creates labels for the case must be unique. The switch expression is matched with each case label. Only the matched case is executed ,if no case matches then the default statement (if present) is executed.

Syntax:

```
switch(control_expression){
case expression 1:
<statement>;
case expression 2:
<statement>;
 ...
 ...
case expression n:
<statement>;
default:
<statement>;
}//end switch
```

Repetition Statements:

While loop statements:

This is a looping or repeating statement. It executes a block of code or statements till the given condition is true. The expression must be evaluated to a boolean value. It continues testing the condition and executes the block of code. When the expression results to false control comes out of loop.

Syntax:

```
 while(expression){
<statement>;
...;
...;
}
```

do-while loop statements:

This is another looping statement that tests the given condition past so you can say that the do-while looping statement is a past-test loop statement. First the **do** block statements are executed then the condition given in **while** statement is checked. So in this case, even the condition is false in the first attempt, do block of code is executed at least once.

Syntax:

```
do{
<statement>;
...;
...;
}while (expression);
```

for loop statements:

This is also a loop statement that provides a compact way to iterate over a range of values. From a user point of view, this is reliable because it executes the statements within this block repeatedly till the specified conditions is true.

Syntax:

```
for (initialization; condition; increment or decrement){
  <statement>;
  ...;
  ...;
  }
```

initialization: The loop is started with the value specified.

condition: It evaluates to either 'true' or 'false'. If it is false then the loop is terminated.

increment or decrement: After each iteration, value increments or decrements.

25. (a)Sum of series

AIM:

To write a java program for find the sum of any number of integers

ALGORITHM:

Step 1: Start the process.

Step 2 : Import input output package.

Step 3 : Create class name as series.

Step 4 : In main function create the object name as "br" buffered reader class.

Step 5 : Enter the value from user using readLine () method.

Step 6: Get the values for the required variables and convert into integer type.

Step 7: To calculate mul=mul*x;sum=sum+mul;

Step 8 : Display the result.

Step 9: Stop the Process

```
import java.util.*;

import java.io.*;

import java.lang.*;

class series

{

public static void main(String[] args) throws IOException

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the n number of values:");
int n=Integer.parseInt(br.readLine());

System.out.println("Enter the value of x:");

int x=Integer.parseInt(br.readLine());

int i,sum=1,mul=1;

for(i=1;i<=n;i++)

{

mul=mul*x;

sum=sum+mul;

}

System.out.println("Sum of Series=" + sum);

}}
```

OUTPUT:

```
Enter the n number of values: 5

Enter the value of x: 1

Sum of Series=6
```

(b) Prime or not

AIM:

To check if a number is prime or not by talking the number as input from the keyboard.

ALGORITHM:

```
Step 1 : Start the process.
Step 2 :  Import input output Stream.
Step 3 : Create class name as prime.
Step 4 : In main function create the object name as "br" buffered reader class.
Step 5 : Enter the value from user using readLine () method.
Step 6 : Declare the variables as count=0,1 type int.
Step 7 : To check whether the given number is prime or not using n/2 and n%i==0
         using for loop.
Step 8 : Stop the process and Display the result.
```

SOURCE CODE

```
import java.util.*;

import java.io.*;

import java.lang.*;

class prime

{

public static void main(String [] args) throws IOException

{

int i,n,count=0;
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the value of n :");

n=Integer.parseInt(br.readLine());

for(i=2;i<n;i++)

{

if(n% i==0)

count++;

}

if(count!=0)

{

System.out.println("The given number is not prime number");

}

else

{

System.out.println("The given number is prime number");

}

}

}
```

OUTPUT:

```
Enter the value of n : 1

The given number is prime number
```

26. (a) Final Keyword

The **final keyword** in java is used to restrict the user. The java final keyword can be used in many contexts. Final can be:

1. variable
2. method
3. class

Final Variable

Any variable which is declared by using the final keyword is called final variable. Final variables can be declare with static keyword in java and treated as constant. A final variable can only be explicitly assigned once.

Final method

To define final keyword with method declaration. It means a method with final keyword is called final method. Final methods are faster than non-final methods because they are not required to be resolved during run-time and they are bonded on compile time.

Final Class

A class with final keyword is known as final class in java. Final class is complete in nature and cannot be inherited. Several classes in Java are final e.g. String, Integer and other wrapper classes.

(b)Packages in Java

A Package can be defined as a grouping of related types (classes, interfaces, enumerations) providing access protection and namespace management.

Existing packages in Java are

java.lang − bundles the fundamental classes

java.io − classes for input , output functions are bundled in this package

Since the package creates a new namespace there won't be any name conflicts with names in other packages. Using packages, it is easier to provide access control and it is also easier to locate the related classes.

Creating a Package

While creating a package, to choose a name for the package and include a package statement along with that name at the top of every source file that contains the classes, interfaces, enumerations, and annotation types that to include in the package.

The package statement should be the first line in the source file. There can be only one package statement in each source file, and it applies to all types in the file.

If a package statement is not used then the class, interfaces, enumerations, and annotation types will be placed in the current default package.

To compile the Java programs with package statements, you have to use -d option as shown below.

```
javac -d Destination_folder file_name.java
```

Reg No................

[16CCP304]

KARPAGAM UNIVERSTIY

(Established Under Section 3 of UGC Act 1956)

Coimbatore – 641021

(For the candidates admitted from 2015 onwards)

II M. Com. CA

Third Semester

First Internal Test, July - 2017

JAVA

Time: 2 hours
Date:

Maximum: 50 marks

PART –A (20*1=20 Marks)
Multiple choice Questions

1. Objects take up ____________in the memory
   (a)Space (b)Address (c)Memory (d)bytes
2. ______________is a collection of objects of similar type
   (a)Objects (b)methods (c)classes (d)messages
3. ________________refers to the act of representing essential features without including the background details or explanations
   (a)Encapsulation (b)inheritance (c)Dynamic binding (d)Abstraction
4. Attributes are sometimes called____________
   (a)data members (b)methods (c)messages (d)functions
5. The functions operate on the datas are called____________
   (a)Methods (b)data members (c)messages (d)classes
6. ____________is the process by which objects of one class acquire the properties of objects of another class
   (a)Polymorphism (b) encapsulation (c)data binding (d)Inheritance
7. A__________ variable is known only in the method that declares the variable.
   (a)Local (b)Global (c)Static (d)Auto
8. A class that cannot be subclassed is called a _______
   (a)final class (b)final vairable (c)final keyword (d)final method
9. The equal comparison operator in Java is ________.
   (a)<> (b)!= (c)== (d)^=
10. What is final variable?
    (a)cannot change its value (b)value can be changed
    (c)value cannot be assigned (d)value can assign
11. Which one of the following is conditional operator?
    (a) ?: (b)?; (c)?. (d):?
12. _______ is an object that contains elements of same data type.
    (a)Array (b)Structure (c)Class (d)Object
13. ________ is a keyword
    (a)import (b) loop (c)export (d)none
14. Which of the following is a method having same name as that of its class?
    (a)finalize (b)delete (c)class (d)constructor
15. Constructor does not have any _________
    (a)return type (b)void (c)object (d)class
16. A package is container of_____
    (a)Methods (b)Objects (c)Classes (d)Variables
17. If a variable is declared as private , then it can be used in _______
    (a)Any class of any package (b)Any class of same package
    (c)Only in the same class (d)Only subclass in that package
18. which package is imported implicitly?
    (a)java.applet (b)java.util (c)java.lang (d)java.io
19. Math class is in ....package.
    (a)java.io (b) java.lang (c) java.util (d)java.applet
20. An _______ is a condition that is caused by a runtime error in the program
    (a)throw (b)exception (c)handle (d)catch

PART –B (3*2=6 Marks)
Answer All the question

21. Write in short notes on:
    (i) Byte code
    (ii) Java Environment
22. Write brief on "Command line arguments".
23. How to declare the Package.

PART –C (3*8=24 Marks)
Answer All the question

24. (a)List out the unique advantages of Object Oriented Programming.
    (Or)
    (b)Describe about Control Structure in Java.
25. (a) Write a program to find sum of series $1+X+X^2+X^3+.....+X^n$
    (Or)
    (b)Write a program to find prime number or not
26. (a)When does the "final" keyword mean in front of a variable, a method and a class.
    (Or)
    (b)Explain the Packages in Java.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2016 onwards)

DEPARTMENT OF COMMERCE (CA)

SUBJECT : JAVA

SEMESTER : III

SUBJECT CODE: 16CCP304 CLASS : II M.COM CA

SECOND INTERNAL EXAMINATIONS-ANSWER KEY

PART A (20 *1=20 Marks)

1. Class name
2. 5
3. Buffered output stream
4. File
5. Main
6. Abstract window toolkit
7. Applets
8. Appletviewer
9. <applet></applet>
10. Font.Bold
11. Update()
12. x-y co-ordinates
13. runtime exceptions
14. exception and error
15. catch
16. throw
17. nested classes
18. checked exceptions
19. long
20. new born

PART B (3 *2=6 Marks)

21. Exception are handled in java

An exception is an abnormal condition that arises in a code sequence at run time. A Java exception is an object that describes an exceptional condition that has occurred in a piece of code. When an exceptional condition arises, an object representing that exception is created and thrown in the method that caused the error.

22. Difference between preemptive scheduling and time slicing

- Under preemptive scheduling, the highest priority task executes until it enters the waiting state or dead state or a higher priority task comes into existence
- Under time slicing,a task executes for a predefined slice of time and then reenters a pool of ready tasks.
- The scheduler then determine which task should executes next ,based on priority and other factors.

23. Method invocation in an Applet

Executable applet is nothing but the .class file of applet, which is obtained by compiling the source code of the applet. Compiling the applet is exactly the smae as compiling an application using following command.

javac appletname.java

The compiled output file called appletname.class should be placed in the same directory as the source file.

PART C(3 *8=24 Marks)

24. (a)Keywords in Exception Handling

Try Block

It is a try block nested inside another try block. If inner try does not have a catch handler then the stack is unwound and the next try block catch handlers are inspected for a match.

```
class NestTry
{
public static void main(String args[])
{
```

```
try
{
int a=args.length;
int b=42/a;
System.out.println("a="+a);
try
{
if(a==1)
        a=9/(a-a);
if(a==2)
{
        int c[]={1};
        c[10]=99;
}
}
catch(ArrayIndexOutOfBoundException e)
{
System.out.println("Array index out of bound" +e);
}
}
catch(ArithmeticException e)
{
System.out.println("Divide by 0"+e);
}
}
}
```

throw Clause

It is possible for the program to throw an exception explicitly rather than being thrown by the Java runtime system.

Syntax:

Throw expression;

Here expression is an object of type Throwable or subclass of Throwable. There are 2 ways to create Throwable object

1. Using a parameter into catch clause
2. creating one with the new operator.

Eg.

```
throw new NullPointerException("demo");
throw new ArtithmeticException();
```

The flow of execution stops immediately after throw statement.

```
class throwtest
{
static void xyz()
{
try
{
Throw new NullPointerException("Exception thrown explicitly");
}
catch (NullPointerException e)
{
System.out.println("Caught inside xyz");
throw e;
}
}
public static void main(String args[])
{
try
{
xyz();
}
catch(NullPointerException e)
{
System.out.println("Caught inside  main"+ e);
}
}
}
```

Output

```
Caught inside xyz
Caught inside main: java.lang.NullPointerException: Exception thrown explicitly
```

(b)

EXCEPTION HANDLING

AIM:

To write a program to display message when mark exceeds hundred

ALGORITHM:

```
Step 1: Start the process.
Step 2: Specify class name which implements Exception class.
Step 3: In the main class create try block to check the condition
        whether mark exceeds 100.
Step 4: Create catch block to catch the exception thrown by the try block
Step 5: Display the result.
```

Step 6: Stop the process.

SOURCE CODE

```
import java.io.*;
import java.lang.Exception;
class myExcept extends Exception
{
myExcept(String message)
{
super(message);
}
}
class numexp
{
public static void main(String args[]) throws IOException
{
BufferedReader h=new BufferedReader(new InputStreamReader(System.in));
int m1,m2,m3,no;
String name;
System.out.println("enter the register number");
no=Integer.parseInt(h.readLine());
System.out.println("enter the name");
name=h.readLine();
System.out.println("enter the three marks:");
m1=Integer.parseInt(h.readLine());
m2=Integer.parseInt(h.readLine());
m3=Integer.parseInt(h.readLine());
try
{
if(m1>100|m2>100|m3>100)
{
```

```
throw new myExcept("marks should be less than 100");
}
}
catch(myExcept e)
{
System.out.println("\n");
System.out.println(e.getMessage());
System.exit(0);
}
System.out.println("name:"+name+"\t number:"+no+"\t mark 1:"+m1+"\t mark
2:"+m2+"\t mark 3:"+m3);
}}
```

OUTPUT:

```
Enter the register number: 101
Enter the name: Kichu
Enter the three marks:
60
70
80
Name: kichu      number: 101     mark 1: 60      mark 2: 70      mark 3: 80
Enter the register number: 102
Enter the name: Kavi
Enter the three marks:
100
200
300
Marks should be less than 100
```

RESULT :

The above program has been executed successfully and the output is verified.

25. (a)Reading Console input

There are few ways to read input string from your console/keyboard. The following smaple code shows how to read a string from the console/keyboard by using Java.

Note: Java Console Example to read password

```
import java.io.*;
class ReadPasswordTest{
public static void main(String args[]){
Console c=System.console();
System.out.println("Enter password: ");
char[] ch=c.readPassword();
String pass=String.valueOf(ch);//converting char array into string.
}
public class ConsoleReadingDemo {

        public static void main(String[] args) {
BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
     System.out.print("Please enter user name : ");
     String username = null;
     try {
        username = reader.readLine();
     } catch (IOException e) {
        e.printStackTrace();
     }
     System.out.println("You entered : " + username);
     Scanner in = new Scanner(System.in);
     System.out.print("Please enter user name : ");
     username = in.nextLine();
     System.out.println("You entered : " + username);
      Console console = System.console();
```

```
        username = console.readLine("Please enter user name : ");
        System.out.println("You entered : " + username);
    } }
```

The last part of code used java.io.Console class. We cannot get Console instance from System.Console() when running the demo code through Eclipse. Because eclipse runs your application as a background process and not as a top-level process with a system console.

(b)Writing Console Output

```
File file = new File("test.txt");
FileOutputStream fis = new FileOutputStream(file);
PrintStream out = new PrintStream(fis);
System.setOut(out);
System.out.println("First Line");
System.out.println("Second Line");
System.out.println("Third Line");
System.out.println("Fourth Line");
```

(b) Threads

Multithreading allows multiple tasks to execute concurrently within a single program. The advantage of multiple threads in a program is that it utilizes system resources better because other threads can grab CPU time when one line of execution is blocked.

MULTIPLICATION TABLE

AIM:

To write a program to display multiplication table using thread

ALGORITHM:

Step 1: Start the process.

Step 2: Specify class name which implements Runnable interface.

Step 3: Define method name which displays multiplication table.

Step 4: In the main class create the thread for different number and start the thread by using the thread object

Step 5: Display the result.

Step 6: Stop the process.

SOURCE CODE

```
import java.io.*;
class table implements Runnable
{
int n;
public table (int x)
{
n=x;
}
public synchronized void run()
{
for(int i=1;i<=5;i++)
{
System.out.println(i+"*"+n+"="+(i*n));
}
}
}
public class multi
{
public static void main(String args[])
{
table t1=new table(3);
table t2=new table(4);
table t3=new table(9);
Thread t=new Thread(t1);
t.start();
t=new Thread(t2);
t.start();
t=new Thread(t3);
t.start();
}
}
```

OUTPUT:

```
1*3=3
1*4=4
1*9=9
2*3=6
2*4=8
2*9=18
3*3=9
3*4=12
3*9=27
4*3=12
4*4=16
4*9=36
5*3=15
5*4=20
5*9=45
```

RESULT :

The above program has been executed successfully and the output is verified.

26. (a)Explain the process involved in drawing ellipses and circle in graphics

Drawing Ellipses and Circles and Ovals

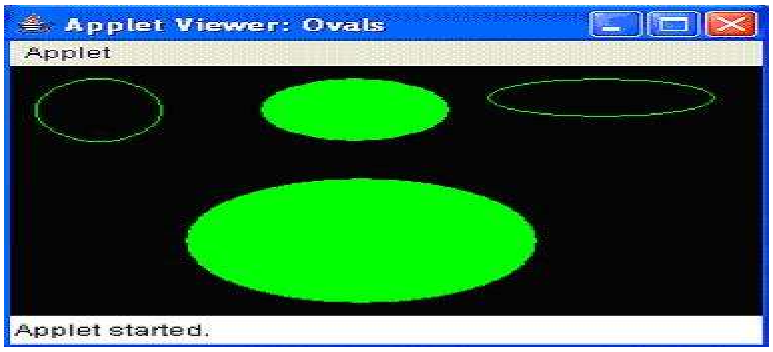
- **public abstract void drawOval (int x, int y, int width, int height):-** The drawOval() method draws an oval in the current color within an invisible bounding rectangle from (x, y) to (x+width, y+height). You cannot specify the oval's center point and radii. If width and height are equal, you get a circle. If width or height is negative, nothing is drawn.
- **public abstract void fillOval (int x, int y, int width, int height):-** The fillOval() method draws a filled oval in the current color within an invisible bounding rectangle from (x, y) to (x+width-1, y+height-1). You cannot specify the oval's center point and radii. Notice that the filled oval is one pixel smaller to the right and bottom than requested. If width or height is negative, nothing is drawn.

Example:

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Ovals" width=300 height=200>
</applet>
*/
public class Ovals extends Applet
{
    public void init()
    {
            setBackground(Color.black);
            setForeground(Color.green);
    }
    public void paint(Graphics g)
    {
            g.drawOval(10, 10, 50, 50);
            g.fillOval(100, 10, 75, 50);
            g.drawOval(190, 10, 90, 30);
            g.fillOval(70, 90, 140, 100);
    }
```

}

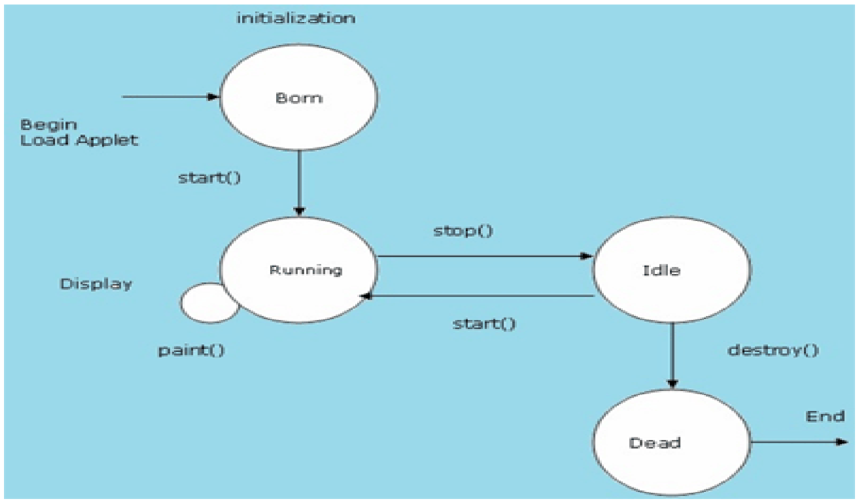
Output:



(b) Life Cycle of an applet

Various states, an Applet, undergo between its object creation and object removal (when the job is over) is known as life cycle. Each state is represented by a method. There exist 5 states represented by 5 methods. That is, in its life of execution, the applet exists in one of these 5 states.

These methods are known as "callback methods" as they are called automatically by the browser whenever required for the smooth execution of the applet. Programmers just write the methods with some code but never call.



Four methods in the Applet class:

- init: This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.

- start: This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.

- stop: This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.

- destroy: This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.

- paint: Invoked immediately after the start() method, and also any time the applet needs to repaint itself in the browser. The paint() method is actually inherited from the java.awt**.**

Register Number__________

[16CCP304]

KARPAGAM UNIVERSITY

Karpagam Academy of Higher Education

(Established under Section 3 of UGC Act 1956)

Coimbatore-641021

(For the candidates admitted from 2016 onwards)

M.Com CA- Third Semester

Second Internal Examination- September 2017

JAVA

Date & Session: 14.09.2017

Duration: 2 Hrs

Max Marks: 50 Marks

PART – A (20 * 1 = 20 Marks)

Answer All the Questions

1. In the first part of the thread, _____ is created for thread.
   a) class b)object c) classname d) packagename
2. A thread is always in one of the following ___ states.
   a)5 b) 4 c) 3 d) 2
3. The _______ class maintains a buffer that is written to when you write to the stream.
   a) DataInputStream b) DataOutputStream
   c) BufferedInputStream d) BufferedOutputStream
4. Java also uses the ______ class to manipulate files.
   a) stream b) File c) String d) Array
5. When a Java program starts up, ____ thread begins running immediately.
   a) program b) main c) function d) input
6. AWT stands for ________.
   a)Abstract Window Toolkit b) Absolute Window Toolkit
   c) Absolute Windowing Toolkit d) Abstract Windowing Toolkit
7. __________ are small applications that are accessed on an internet server.
   a) utilities b) networks c) applets d) bean
8. The compiled applet is tested using___________.
   a) word b) dos c) notepad d) applet viewer
9. The complete Applet tag of HTML is ________.
   a) <APPLET> </APPLET> b) <APPLET /APPLET>
   c) APPLET/APPLET> d) <APPLET APPLET>
10. Font style can be specified using the constants as ________.
    a) Font.BOLD b) FONT.BOLD c) FONT.Bold d) font.bold
11. ____ method clears the screen area.
    a) update() b)delete c) destroy() d) backspace
12. The Graphics class can be used to draw figures and images using __________.
    a) X-Y coordinates b) X coordinates c) Y coordinates d) shapes
13. Unchecked exceptions are extensions of ________.
    a) checked exceptions b) unchecked exceptions
    c) runtime exceptions d) IO Exception
14. The two subclass of throwable class are __________.
    a) Exception and Error b) Exception and handler
    c) throw and throwable d) try and catch
15. ______________ specifies the type of exception to be caught.
    a) Catch b) try c) exception d) block of code
16. ______ keyword is used to identify the list of possible exceptions that a method might throw.
    a) throw b) try c) catch d) block of code
17. Inner classes are_________.
    a) anonymous classes b) nested classes c) sub classes d) derived classes
18. The compiler checks that the method must either handle the exception or pass it to the caller method is ______.
    a) checked exceptions b) unchecked exceptions
    c) runtime exceptions d) IO Exception
19. What is the data type for the parameter of the sleep() method?
    a) long b) int c)byte d) double
20. Creation of thread object is said to be ________ state.
    a) newborn b) runnable c) running d) blocked

PART – B (3 * 2 = 6 Marks)

Answer All the Questions

21. How the exceptions are handled in java?
22. What is the difference between preemptive scheduling and time slicing?
23. What is the order of method invocation in an Applet?

PART – C (3 * 8 = 24 Marks)

Answer All the Questions

24. (a) Explain following keywords used in Exception Handling.

(i) try (ii) catch (iii) throw

(Or)

(b) Write a program to create and exception for mark out of bounds. If mark is greater than 100 throw an exception

25. (a) Write in detail on the following.

(i) Reading console input (ii) Writing console output

(Or)

(b) Define threads. Explain multiplication table using multithreading with suitable program.

26. (a) Explain the process involved in drawing ellipses and circle in graphics.

(Or)

(b) Discuss in detail about life cycle of an Applet.

50 Copies

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF COMMERCE

LECTURE PLAN

STAFF NAME	**: K.GOMATHI**		
SUBJECT NAME	**: JAVA**	**SUBJECT CODE**:	**16CCP304**
SEMESTER	**: III**	**CLASS**	**: II M.COM CA**

UNIT-I

Sl No.	Lecture Duration (Hour)	Topics to be Covered	Support Materials
1	1	**Introduction Of Object-Oriented Programming** ➢ Object-Oriented Programming Concepts ➢ Recapitulating Procedure-Oriented Programming	**T1:**19-26 **R2:**1-7 **R3**: 3-8 **R4**:1-3 **W2**
		➢ Structured versus Object –Oriented Approach	**R1:**17-25 **R2:**:7-8
2	1	➢ OOPs Language	
3	1	**The Java Language** ➢ Features of Java	**R2**:10-12 **R3**: 13-15 **R5:** 3-5
4	1	➢ Java Environment ➢ The Java Architecture	**R2**:11-15
5	1	➢ Java Development Kit ➢ Types of Java Program	**R1:**25-31 **R2**:15-17 **R3**: 27-30 **R5:**4
6	1	**Variable Declaration & Arrays** ➢ Data Types In Java ➢ Java Tokens ➢ Variable Declaration	**T1:**47-60 **R1:**42-56 **R2**:20-24 **R3**:35-37,51-54 **R5:** 9-12 **W1**
7	1	➢ Type Casting and conversion	**T1:**62-66

8	1	➢ Arrays	**R1:**57-67 **R2**:24-28 **R3**:59-61 **R4:**33-40
9	1	**Operators** ➢ Operators Introduction ➢ Operator Precedence	**T1:**67-87 **R1:** 74-98 **R2**:31-38 **R3**:69-82 **R5:**12-13
10	1	**Control Statements** ➢ Control Statements-Introduction ➢ Selection constructs	**T1:**90-97 **W1** **R1:**100-108 **R2**:41-51 **R3**:88-123 **R4:** 22-25
11	1	➢ Iteration constructs ➢ Jump Statements	**T1:**90-97 **R1:**109-127 **R2**:47-51**R3**:111-123 **R5:** 26-31
12	1	Recapitulation & Important Questions Discussion	
Total No .Of Hours			**12 Hours**
		UNIT –II	
1	1	**Introduction to Classes** ➢ Class Fundamentals ➢ Declaring Objects	**T1:**106-107 **R1:**129-134 **R2:**:54-56 **R3**:129-130
2	1	➢ Assigning Object Reference Variables ➢ Methods	**T1:**108 **R1:**134 -145 **R2:**56-59 **R3**:130-133
3	1	Constructors ➢ Parameterized Constructors	**T1:**114-115 **R1:**145-149 **R2**:59-61 **R3**:137-138
4	1	➢ The this Keyword ➢ Garbage Collection ➢ Finalize() Method	**R1:149-151 R2:65-66 R5:79,90**
5	1	**Methods and Classes** ➢ Overloading Methods	**T1:**115-116 **R2**:70-72 **R1**:156-159 **W2**
6	1	➢ Overloading Constructor	**R1:** 159-162 **R2**:72-74

			W2
7	1	➢ Returning Objects ➢ Recursion	R1:168 -172 R2:78-81
8	1	➢ Static ➢ Final ➢ Nested and Inner Class	R1:176-184 T1:123-124 R2:84 -85
9	1	➢ Command line Arguments	R1:188 R2:85-86
10	1	**Inheritance** ➢ Basics of Inheritance ➢ Super Class Variable And Subclass Object ➢	**T1:**118-122 **R1:**190-196 **R2**:89-93
11	1	➢ Method Overriding ➢ Final Keyword	**T1:**120-122 **R1:** 208-209 **R2**:100-103
12	1	Recapitulation & Important Questions Discussion	
Total No .Of Hours			**12 Hours**
UNIT-III			
Sl No.	**Lecture Duration (Hour)**	**Topics to be Covered**	**Support Materials**
1	1	**Packages and Access Modifiers** ➢ Packages –An Introduction	**T1:**129-130 **W1** **R1:**224-227 **R2**:161-16 **R3**:188 **R5:** 84
2	1	➢ Package Declaration ➢ Access Protection	**R2**:164-165 **R3**:192-193 **R5:** 84
3	1	➢ Importing Packages	**T1:**132-143 **R1:**227-230 **R2**:167-174
4	1	Interfaces ➢ Defining an Interface ➢ Implementing Interfaces	
5	1	**Exception Handling : Introduction** ➢ Fundamentals of Exception handling	**T1**:166 **R1:**250 **R2**:120-121 **W1**
6	1	➢ Hierarchy Of The Exception Classes	**R2:**121
7	1	➢ Types Of Exception	**T1**:166 **R1:** 251-252
8	1	➢ Try and Catch ➢ Multiple Catch	**T1:166-175** **R2:161-166**

9	1	➢ Nested Try ➢ Throw	**R3:188-193**
10	1	➢ Throws ➢ finally	
11	1	**Built in Exception**	R2:133
12	1	Recapitulation & Important Questions Discussion	
Total No .Of Hours			**12 Hours**
UNIT IV			
1	1	**Multithreaded Programming** ➢ The Thread Moel	**T1**:181-185 **R2**:138-146
2	1	➢ Life cycle of Thread	
3	1	➢ **Thread Creation**	**R2:**140
4	1	➢ **Multiple thread**	
5	1	➢ Thread Priorities	**T1**:188-190 **R1:**292-297 **R2**:149-152
6	1	➢ Synchronization	
7	1	➢ Inter thread Communication ➢ Suspending,	**R2**:152-155
8	1	➢ Resuming and Stopping Threads	**R2**:155-158
9	1	**Input Output Classes** ➢ Input and Output Operations	**T1:**219-220 **R2**:253-254 **R3**:293
10	1	➢ Input Stream Classes	**T1**:224-225 **R1:** 314-318 **R2**:258-263 **R5:**223-225
11	1	➢ Output Stream Classes	
12	1	Recapitulation & Important Questions Discussion	
Total No .Of Hours			**12 Hours**
UNIT-V			
1	1	**Applet** ➢ Applet Basics	**T1**:253-256 **R1:**328-331
2	1	➢ Building Applet Code	**R2**:292 **R3**:237-245 **R5:**100 **W1**
3	1	➢ Applet Life Cycle ➢ Creating an executable applet	**T1:**262-267 **R1:**705-712 **R2**:293 **R4:**110-

			111
4	1	➢ Designing a Web Page	**T1**:268-272 **R1:**712-715
5	1	➢ Running the applet ➢ Getting input from the user	**R2**:302-304 **R4:**112-115
6	1	**Graphics Programming** ➢ The Graphics Class	**R2**:300-302
7	1	➢ Line and Rectangle ➢ Circle and Ellipse	
8	1	➢ Using Controls loops in applet ➢ Drawing bar charts	**R1:**695-702 **R5:**150-158
9	1	Recapitulation & Important Questions Discussion	
10		Previous year ESE Question Papers	
11	1	Previous year ESE Question Papers	
12		Previous year ESE Question Papers	
Total No .Of Hours			**12 Hours**
Total Planned Hours			**60 Hours**

Text Book

1. Partrick Naughton. (2002). *Java Hand Book*. New Delhi: McGraw Hill Publishing Company Limited.

Reference Books

1. Herbert Schildt. (2014).*Java Complete Reference*, 9th edition, Tata McGraw hill.
2. Permanand Mohan. (2013). *Fundamentals of Object-Oriented Programming in Java*, 1st edition, Createspace Independent Publishing
3. Balagurusamy,E. (2014). *Programming with Java*, 5th edition, Tata McGraw-hill Publishing Company limited.
4. Daniel Liang,Y. (2015). *Introduction to Java Programming*, 10th edition, Pearson Prentice Hall.

Website

1. http://www.tutorialspoint.com/java/
2. http://docs.oracle.com/javase/tutorial/java/
3. http://javabeginnerstutorial.com/core-java/
4. http://www.learnjavaonline.org/

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF COMMERCE (CA)

SUBJECT: : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 **CLASS : II M.COM CA**

JAVA (16CCP304) - UNIT II

	QUESTIONS	OPT1	OPT2	OPT3	OPT4	ANSWER
1	It takes no parameters	Default Constructors	Copy Constructors	Parameter Constructor	Function	Default Constructors
2	It is required when objects are required to perform a similar task	Method Overriding	Polymorphism	Static Binding	Method Overloading	Method Overloading
3	It is used to refer to the current object	this reference	that reference	dot	Arrow	this reference
4	Which of the following is a valid identifier?	area	Class	9X	8+9	area
5	A literal character is represented inside a pair of ______	single quotes	double quotes	brackets	paraenthesis	single quotes
6	short is a signed _________ type	8 bit	16 bit	32 bit	64 bit	16 bit
7	Single precision is specified by ________keyboard	int	double	float	char	float
8	An _________ is a group of like_typed variables that are referred to by a common name	instance	array	class	Method	instance
9	The operators which have single operand is called ______	binary	unary	ternary	logical	binary
10	The operators which come after the operand is called __	postfix	prefix	superfix	superfix	postfix
11	A ________ is the one that describes the general attributes of an object, including types of each attribute and the methods that can	object	variable	methods	functions	object
12	Which is invalid?	int a;	float x,y,z;	INT abc;	double a;	INT abc;
13	Which of these data type requires the most amount of memory?	long	Int	Short	byte	long
14	To declare an int variable number with initial value 2, you write	int number = 2L;	int number = 2l;	int number = 2;	int number = 2.0;	int number = 2;
15	What is result of 45 / 4?	10	11	11.25	12	11
16	Which of the following assignment statements is correct?	char c = 'd';	char c = =100;	char c = "d";	char c = "100";	char c = 'd';
17	The equal comparison operator in Java is __________.	<>	!=	==	^=	==

18	To add number to sum, you write (Note: Java is case_sensitive).	number += sum;	number = sum + number;	sum = Number + sum;	sum += number;	sum += number;
19	What are the values of X and Y. if x=5 and y=++x	x=6 and y=6	x=5 and y=6	x=5 and y=5	x=0 and y=5	x=6 and y=6
20	A___________ variable is known only in the method that declares the variable.	Local	Global	Static	Auto	Local
21	Which of the following is a valid identifier?	$343	Class	9X	8+9	$343
22	To declare a constant MAX_LENGTH inside a method with value 99.98, you write	final MAX_LENGTH = 99.98;	final float MAX_LENGTH = 99.98;	double MAX_LENGTH =	final double MAX_LENGTH =	final double MAX_LENGTH = 99.98;
23	Which of the following is a constant, according to Java naming conventions?	MAX_VALUE	Test	read	ReadInt	MAX_VALUE
24	A class that cannot be subclassed is called a _______	final class	final vairable	final keyword	final method	final class
25	Inheritance is the process of	using classes in the established standard Java	using features from an existing class.	combining data and the methods, which process	dividing a program into multiple related files for	using features from an existing class.
26	Which of the following expression results in a value 1?	2 % 1	15 % 4	25 % 5	37 % 6	37 % 6
27	To assign a double variable d to an float variable x, you write	x = (long)d	x = (int)d;	x = d;	x = (float)d;	x = (float)d;
28	Which of the following assignment statements is illegal?	float f = _34;	int t = 23;	short s = 10;	int t = (int)false;	int t = (int)false;
29	If you attempt to add an int, a byte, a long, and a double, the result will be a _____ value.	byte	int;	long;	double;	double;
30	Which of the following is the correct expression of character 4?	4	"4"	'\0004'	‘4’	‘4’
31	What declarations are required for every Java application?	nd the main( ) method dec	A Class declarations	Only main() declarations	public access modifier	A class and the main( ) method declarations
32	Which of the following assignment statements is correct?	char c = 'd';	char c = =100;	char c = "d";	char c = "100";	char c = 'd';
33	Which of the Boolean expressions below is incorrect?	(true) && (3 => 4)	!(x > 0) && (x > 0)	(x > 0) || (x < 0)	(x !== 0) || (x = 0)	(true) && (3 => 4)
34	Which of the following is the correct expression that evaluates to true if the number x is between 1 and 100 or the number is	1 < x < 100 && x < 0	((x < 100) && (x > 1)) || (x < 0)	((x < 100) && (x > 1)) && (x < 0)	(1 > x > 100) || (x < 0)	((x < 100) && (x > 1)) || (x < 0)
35	The "less than or equal to" comparison operator in Java is __________.	<	<=	=<	<<	<=
36	The equal comparison operator in Java is __________.	<>	!=	==	^=	==
37	Suppose x=10 and y=10 what is x after evaluating the expression (y > 10) & (x++ > 10).	9	10	11	12	11
38	What is final variable?	cannot change its value	value can be changed	value cannot be assigned	value can assign	cannot change its value
39	The __________ method parses a string s to a double value.	double.parseDouble(s);	Double.parsedouble(s);	double.parseDouble(s);	Double.parseDouble(s);	Double.parseDouble(s);
40	The __________ method returns a raised to the power of	Math.power(a,b)	Math.exponent(a,b)	Math.pow(a,b) ;	Math(a.b)	Math.pow(a,b) ;
41	If a program compiles fine, but it produces incorrect result, then the program suffers __________.	compilation error	runtime error	logic error	Syntax error	logic error

42	Analyze the following code: boolean even = false; if (even = true) { System.out.println("It is even!"); }	The program has a syntax error.	The program has a runtime error.	The program runs fine, but displays nothing.	The program runs fine and displays It is even!.	The program runs fine and displays It is even!.
43	Variables of type boolean are given the value ________ by default.	1	0	true	FALSE	FALSE
44	Which one of the following is conditional operator?	?:	?;	?.	:?	?:
45	The number used to refer to a particular element of an array is called the element's _________	Pointer	Index	0	1	Index
46	________ is an object that contains elements of same data type.	Array	Structure	Class	Object	Array
47	What is the representation of the third element in an array called a?	a[2]	a(2)	a[3]	a(3)	a[2]
48	Which of the following is correct?	int[] a = new int[2];	int a[] = new int[2];	int[] a = new int(2);	int a() = new int[2];	int[] a = new int[2];
49	Which of the following statements is valid?	int i = new int(30);	double d[] = new double[30];	char[] c = new char[4]{'a', 'b', 'c', 'd'};	char[] c = new char();	double d[] = new double[30];
50	the length of a string by calling the ____ method	strlen()	len()	length( )	none	length( )
51	What are the two parts in executing a Java program and their purpos	Java compiler and Java interpreter	Java Compiler and Source Code	Java Interpreter an Source Code	JVM and throws	Java compiler and Java interpreter
52	_________ is a keyword	import	loop	export	none	import
53	Which of the following is a method having same name as that of it's class?	finalize	delete	class	constructor	constructor
54	Constructor does not hava any __________	return type	void	object	class	return type
55	Which function is used to perform some action when the object is to be destroyed?	Finalize()	delete()	main()	void()	Finalize()
56	Which of these access specifiers must be used for main() method?	private	public	protected	import	public
57	Arrays in Java are implemented as?	class	object	variable	methods	object
58	Which of this keyword must be used to inherit a class?	super	this	extent	extends	extends
59	Which of these is correct way of inheriting class A by class B?	class B + class A {}	class B inherits class A {}	class B extends A {}	class B extends class A {}	class B extends A {}
60	Which of this keyword can be used in subclass to call the constructor of superclass?	super	this	extent	extends	super

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2016 onwards)

DEPARTMENT OF COMMERCE (CA)

SUBJECT: : JAVA

SEMESTER : III

SUBJECT CODE: 16CCP304 CLASS : II M.COM CA

JAVA (16CCP304) - UNIT III

	QUESTIONS	OPT1	OPT2	OPT3
1	Which of these keywords is used to define packages in Java?	pkg	Pkg	package
2	Which of these is a mechanism for naming and visibility control of a class and its content?	Object	Packages	Interfaces
3	members can be accessed by a different class in the different package?	Public	Protected	Private
4	Which of the following is correct way of importing an entire package ‘pkg’?	import pkg.	Import pkg.	import pkg.*
5	Which of the following package stores all the standard java classes?	lang	java	util
6	Which of these keywords is used to define interfaces in Java?	interface	Interface	intf
7	A package is container of _____	Methods	Objects	Classes
8	If a variable is declared as private , then it can be used in _______	Any class of any package	Any class of same package	Only in the same class

9	which package is imported implicitly?	java.applet	java.util	java.lang
10	Math class is in ………package.	java.io	java.lang	java.util
11	An _________ is a condition that is caused by a runtime error in the program	throw	exception	handle
12	The data type wrapper classes are in……….package	java.lang	java.io	java.util
13	Exception can be generated by the ___________ or manually by the code	Throwable class	Java runtime system	object
14	All exception types are subclasses of the built_in class ____________	Throwable	RuntimeException	StackTree
15	All exception classes are divided into ________ groups	3	4	2
16	The _______ defines the exceptions which are not expected to be caught	java.lang.Error	java.lang.Math	java.lang.Throwable
17	When an exception occurs within a java method, the method creates an exception object and hands it over to the runtime systewm is called	catching the exception	throwing an exception	handle the exception
18	When java method throws an exception the java runtime system searches all the methods in the call stack to find one that can handle	catching the exception	throwing an exception	handle the exception
19	Inner classes are	anonymous classes	nested classes	sub classes
20	The errors are printed by ____________	Stack Trace	StackTree	Message
21	Packages denote _____	classes and interfaces	class	classes
22	Choose the correct statement for package decleration	package package_name;	packagepackage.name	package_name
23	To compile and running packages from a command line	javac package_name/classname.	javac package	java classname

24	* symbol denotes	wildcard character	asterick	multiplication
25	All exception classes are divided into ________ groups	3	4	2
26	If a class includes an interface but does not fully implement the method is ____	partial implementations	callback()	final()
27	One interface can inherit another by use of the keyword	extends	scope	braces
28	Java exception handling is managed via _____ keyword	5	4	3
29	A _____ statement cannot catch an exception thrown by another try statement	catch	try	throw
30	Throwable overrides the _________ method	toString()	String()	get()
31	The general form of throw statement is	throw ThrowableInstance;	throw;	throw classname;
32	____ clause lists the types of exceptions that a method might throw	throw	IO exception	try
33	_______ creates a block of code that will executed after a try/catch statements.	finally	try	catch
34	The ________ class does not define any methods of its own.	exception	java	throw
35	Choose the correct statement for displaying output	system.out.println("**")	System.out.println("**")	system.out.Println("**")
36	Exception performs ______________ tasks	3	4	5
37	Each of Exception's predefined class provide ______________ constructors	3	4	5
38	Once an exception has been thrown, it must be ________ by an exception handler	caught	try	catch

39	____________ is an important subclass of exception	RuntimeException	AarithmeticException	NullException
40	There are ___________ ways of creating Throwable object	3	4	5
41	Certain block of code necessarily has to be run no matter of what exceptions occurs. Those codes are identified using the keyword	throw	final	finally
42	When an exceptional conditional arises, an object representing that exeption is created and _____ in the method that caused the error.	thrown	catch	try
43	Program statements that you want to monitor for exceptions are contained within ________ block	try	catach	final()
44	If an exception occurs within the try block, then it is _____	thrown	thwows	catch
45	Your code can catch this exception using _____ and handle it in some rational manner.	catch	IO exception	try
46	System-generated exceptions are automatically ______ by the Java run-time system	thrown	throw	try
47	Manually, throw an exception , use the keyword ______	throw	threw	throws
48	The compiler checks that the method must either handle the exception or pass it to the caller mehod is	checked exceptions	unchecked exceptions	runtime exceptions
49	The ______________ Keyword is used to specify a block of code that should be guarded against all exceptions.	Catch	try	exception
50	Unchecked exceptions are extensions of ________	checked exceptions	unchecked exceptions	runtime exceptions
51	The two subclass of throwable class are __________	Exception and Error	Exception and handler	throw and throwable
52	Exception performs ______________ tasks	3	4	5

53	When java method throws an exception the java runtime system searches all the methods in the call stack to find one that can handle	catching the exception	throwing an exception	handle the exception
54	When an exception occurs within a java method, the method creates an exception object and hands it over to the runtime systewm is called	catching the exception	throwing an exception	handle the exception
55	The _______ defines the exceptions which are not expected to be caught	java.lang.Error	java.lang.Math	java.lang.Throwable
56	One try block can be ____ inside the another try block	nested	merged	placed
57	______________ specifies the type of exception to be caught.	Catch	try	exception
58	All exception types are subclasses of the built_in class ____________	Throwable	RuntimeException	StackTree
59	Exception can be generated by the ___________ or manually by the code	Throwable class	Java runtime system	object
60	_____________ keyword is used to identify the list of possible exceptions that a method might throw.	throw	try	catch

OPT4	ANSWER
Package	package
import	Packages
default	Public
Import pkg.*	import pkg.*
java.packages	java
Intf	interface
Variables	Classes
Only subclass in that package	Only in the same class

java.io	java.lang
java.applet	java.lang
catch	exception
java.applet	java.lang
catch	Java runtime system
LocalizedMessage	Throwable
6	2
java.lang.IOException	java.lang.Error
get the exception	throwing an exception
get the exception	catching the exception
derived classes	nested classes
Error	Stack Trace
interface	classes and interfaces
package package.name	package package_name;
package.classname.javac	javac package_name/classname.java

special character	wildcard character
6	2
greed()	partial implementations
inheritance	extends
2	5
throws	catch
put()	toString()
throw throws	throw ThrowableInstance;
catch	throw
throw	finally
package	exception
system.Out.println("**")	System.out.println("**")
2	4
2	2
send to source file	caught

Subclasses of Throwable	RuntimeException
2	2
try	finally
finally	thrown
throw	try
IO Exception	thrown
throw	catch
catch	thrown
catch	throw
IO Exception	runtime exceptions
block of code	try
IO Exception	runtime exceptions
try and catch	Exception and Error
2	4

get the exception	catching the exception
get the exception	throwing an exception
java.lang.IOException	java.lang.Error
chained	nested
block of code	Catch
LocalizedMessage	Throwable
catch	Java runtime system
block of code	throw

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF COMMERCE (CA)

SUBJECT: : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 CLASS : II M.COM CA

JAVA (16CCP304) - UNIT IV

	QUESTIONS	OPT1	OPT2	OPT3	OPT4	ANSWER
1	The concept of reading and writing data as ______________ of either bytes or characters	stream	file	java.io	reader	stream
2	What is the mechanisam defind by java for the Resources to be used b	priority	parameters	arguments	Synchronization	Synchronization
3	To support input and output package ________is used	java.util	java.awt	java.lang	java.io	java.io
4	What is the data type for the parameter of the sleep() method?	long	int	byte	double	long
5	What is the mechanism defined by java for the Resources to be used b	priority	parameters	arguments	Synchronization	Synchronization
6	Garbage collector thread belongs to which priority?	high-priority	low-priority	middle-priority	highest-priority	low-priority
7	When a Java program starts up, ____ thread begins running immediately	program	main	function	input	main
8	The ____ method causes the thread from which it is called to suspend execution for the specified period of milliseconds	wait()	notify()	sleep( )	run()	sleep( )
9	To implement Runnable, a class need only implement a single method called ____	wait()	notify()	sleep( )	run()	run()
10	A ____ is an object that is used as a mutually exclusive lock to achieve synchronization	monitor	thread	process	applet	monitor
11	Garbage collector thread belongs to which priority?	high-priority	low-priority	middle-priority	highest-priority	low-priority
12	InputStream suports certain methods, all of which throw an IOException on error conditions	ByteStreams	InputStream	OutputStream	Character streams	InputStream
13	When a Java program starts up, ____ thread begins running immediately	program	main	function	input	main
14	The ____ method causes the thread from which it is called to suspend execution for the specified period of milliseconds	wait()	notify()	sleep( )	run()	sleep( )
15	To implement Runnable, a class need only implement a single method called ____	wait()	notify()	sleep( )	run()	run()

16	A ____ is an object that is used as a mutually exclusive lock to achieve synchronization	monitor	thread	process	applet	monitor
17	What is the data type for the parameter of the sleep() method?	long	int	byte	double	long
18	_________ allows multiple tasks to execute concurrently within a single program	threads	multithreading	execution	scheduling	multithreading
19	________ method is used to avoid collision	synchronization	multithreading	runnable	deadlock	synchronization
20	_____ allows a running program to perform several tasks apparently at the same time	synchronization	multithreading	runnable	deadlock	multithreading
21	Any single path of execution within a program that can be executed independently, is called a	thread	multithreading	runnable	deadlock	thread
22	______ has its own stack, priority and virtual set of registers	thread	multithreading	runnable	deadlock	thread
23	The ________ class provides a buffered stream of input	DataInputStream	DataOutputStream	BufferedInputStream	BufferedOutputStream	BufferedInputStream
24	The ________ class maintains a buffer that is written to when you write to the stream	DataInputStream	DataOutputStream	BufferedInputStream	BufferedOutputStream	BufferedOutputStream
25	The _____________ class is designed primarily for printing output data as text	print	primtln	PrintStream	write	PrintStream
26	Runnable interface is defined in one of the JDK packages called	java.util	java.awt	java.lang	java.io	java.lang
27	The method which specifies only one method, the run() method is	runnable	multithreading	synchronization	deadlock	runnable
28	_________ method establishes the new entry point for another, concurrent thread of execution	run()	runnable	sleep( )	wait()	run()
29	DataInput is ___________________	an abstract class	used to read primitive data types	an interface that defines method to open files	an interface that defines method to read primitive data types	an interface that defines method to read primitive data types
30	Which of the following statements are valid?	new DataInputStream();	new DataInputStream(new File("in.dat"));	new DataInputStream("in.dat ");	new DataInputStream(new FileInputStream("in.dat"	new DataInputStream();
31	Which method is used to call the run() method	start()	run()	wait()	sleep()	start()
32	In the first part of the thread, _____ is created for thread	class	object	classname	packagename	class
33	A thread is always in one of the following ___ states	5	4	3	2	5
34	______ programs create thread objects to perform concurrent tasks	java	c++	c	BASIC	java
35	Creation of thread object is said to be ________ state	newborn	runnable	running	blocked	newborn
36	______ is an independent line of execution within a program	thread	multithreading	runnable	Synchronisation	thread

37	Every ______ has a priority value associated with it	thread	multithreading	runnable	Synchronisation	thread
38	_______ method is used to serialize access to shared resources	thread	multithreading	runnable	Synchronisation	synchronization
39	The state which is ready for execution and is waitng for the CPU time	runnable	newborn	running	dead	runnable
40	A thread is said to be in _______ when it is suspended, sleeping or waiting in order to satisfy some condition	blocked state	newborn state	running state	dead state	blocked state
41	The suspended thread can be revived by using the	resume() method	runnable interface	multithreading	run() method	resume() method
42	System.out refers to the	standard output stream	system output stream	sequential output stream	standard output string	standard output stream
43	System.in refers to the	standard input stream	system input stream	sequential input stream	standard input string	standard input stream
44	System.err refers to	standard error stream	stand error stream	standard execution stream	standard error string	standard error stream
45	An input stream that contains methods for reading the java standard types is	DataInputStream	DataOutputStream	BufferedInputStream	BufferedOutputStream	DataInputStream
46	An output stream that contains methods for reading the java standard types is	DataInputStream	DataOutputStream	BufferedInputStream	BufferedOutputStream	DataOutputStream
47	Input stream that reads from a string	StringBufferInputStream	standard input stream	system input stream	Stream Tokenizer	StringBufferInputStream
48	The lowest level input method is	read()	write()	read() and write()	run()	read()
49	The method which automatically reads a sequence of characters from the input stream and returns them in an object of type String is	readLine()	readline()	readl()	readln()	readLine()
50	________ class in java does not specify how information is retrieved from or stored in files	stream	File	String	Array	File
51	The ________ class also defines platform_dependent constants that can be used to separate the diredtory and the file components in	stream	File	java.io	reader	File
52	The method to check for directory is _________	isFile()	isDirectory()	File	String	isDirectory()
53	The ________ class define byte input streams that are connected to files	InputStream	OutputStream	FileInputStream	FileOutputStream	FileInputStream
54	The ________ class define byte output streams that are connected to files	InputStream	OutputStream	FileInputStream	FileOutputStream	FileOutputStream
55	The FileInputStream class provides an implementation for the _________ methods defined in its superclass InputStream	read()	write()	update()	replace()	read()
56	The FileOutputStream class provides an implementation for the _________ methods defined in its superclass OutputStream	read()	write()	update()	replace()	write()
57	The method __________ is used to write string value	readChars()	writeChars()	read()	write()	writeChars()

58	The ________ class provides a buffered stream of input	DataInputStream	DataOutputStream	BufferedInputStream	BufferedOutputStream	BufferedInputStream
59	The ________ class maintains a buffer that is written to when you write to the stream	DataInputStream	DataOutputStream	BufferedInputStream	BufferedOutputStream	BufferedOutputStream
60	Java also uses the ______ class to manipulate files	stream	File	String	Array	File

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF COMMERCE (CA)

SUBJECT: : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 CLASS : II M.COM CA

JAVA (16CCP304) - UNIT 5

	QUESTIONS	OPT1	OPT2	OPT3	OPT4	ANSWER
1	AWT stands for _________	Abstract Window Toolkit	Absolute Window Toolkit	Absolute Windowing Toolkit	Abstract Windowing Toolkit	Abstract Window Toolkit
2	__________ are small applicationsthat are accessed on an internet server	utilities	networks	applets	bean	applets
3	The compiled applet is tested using ___________	word	dos	notepad	applet viewer	applet viewer
4	The __________ tag is used to start an applet from both HTML and JDK applet viewer	Html	JDK	applet	title	applet
5	Applet basically is a Java class defined in the _____ package of JDK	java.awt	java.lang	java.applet	java.util	java.applet
6	Color class also defines _________ common colors as constants	Canvas	Frame	Dialog	Panel	Dialog
7	The Applet class which is in the java.applet package inherits the properties of the _______ class which is in the java.awt package	Container	Componenet	Panel	List	Panel
8	The Panel class inherits the properties of the _________ class in the java.awt package	Container	Componenet	Panel	List	Container
9	The container class inherits the properties of the ______________ class	Container	Componenet	Panel	List	Componenet
10	An _______ is a window based event driven program	Html	JDK	applet	title	applet
11	The _______ and _______ method executes only once	stop() and destroy()	start() and stop()	init() and paint()	init() and destroy()	init() and destroy()
12	Immediately after calling init() methodthe browser calls the __________________ method	stop()	start()	init()	destroy()	start()
13	The ________ method also called when the user returns to an HTML page that contains the applet	paint()	init()	destroy()	start()	start()
14	The ________ methodis called each time your applet's output is redrawn	stop()	start()	init()	paint()	paint()
15	The ________ method acalled when the user moves from the HTML page that contains an applet	paint()	init()	stop()	destroy()	stop()

16	The _______ method that is used to release additional resource	paint()	init()	destroy()	start()	destroy()
17	There are ______ main methods defined in java.awt.Component	2	4	5	3	3
18	The _____ method is defined by the AWT and is usually called by the applet for screen updating	paint()	init()	stop()	repaint()	repaint()
19	________ class cannot be created directly by using constructors	Panel	Container	Componenet	Graphics	Grapahics
20	In java color is encapsulated by the ________ class	Container	Componenet	Graphics	Color	Color
21	Color class also defines _________ common colors as constants	10	13	12	14	13
22	Methods of ________ class can also be used in the Graphiocs class methods to set and get the background and foreground colors	Container	Componenet	Panel	List	Componenet
23	There are ___________ common terms that are used when describing fonts	2	4	5	3	5
24	The java.applet package defines _______ inetrfaces	2	4	5	3	3
25	The user cannot have their HTML document,applet code,data and web browser at _____________ different locations	2	4	5	3	4
26	The loop() method plays the audio clip automatically while __________ plays it only once	paint()	play()	init()	start()	play()
27	The audio clip can be stopped by calling the ______ method	paint()	init()	stop()	repaint()	stop()
28	The _________ interface provides the inter_communication between an applet and the parent container	AppletContext	AppletStub	getApplet	showDocument	AppletStub
29	The _________ inetface gives the information about the applet's execution environment	AppletStub	getApplet	AppletContext	showDocument	AppletContext
30	The setBackground() is the part of the class ______	Graphics	AppletStub	Component	Container	Component
31	If you want to assign a vlaue 99 to a variable called number, which of the following lines you will use within an applet tag?	number=99	param = number value=99	param name = number value=99	param number =99	param name = number value=99
32	____ parameters are passed to drawArc method	4	5	6	3	3
33	____ is the default color for drawing graphics color	white	black	red	green	black
34	how many colors does a GIF image can have?	180	256	3600	4800	256
35	when a portion of a applet window is to be redrawn ____ method is used	paint()	start()	update()	repaint()	update()
36	______ method is used set the background color	setbackGround()	Setcolor()	setBackGround()	setBackground()	setBackground()
37	_______is the distance from the base line to the top of the character	font size	ascent	descent	baseline	ascent

38	_____________is the distance from the base line to the bottom of the character	font size	ascent	descent	baseline	descent
39	To get the URL of the applet, you use __________.	getCodeBase()	getDocumentBase()	returnCodeBase()	returnDocumentBase()	getCodeBase()
40	To get the image file at a specified URL, you use __________.	getImage(url)	createImage(url)	url.getImage()	url.createImage()	getImage(url)
41	The ________ method of class Graphics draw a line between two points.	Line	Putline	drawline	getline	drawline
42	When init() method calls immediately, the browser calls the	start() method	paint() method	applet	applet viewer	start() method
43	________ method clears the screen area	update()	delete	destroy()	backspace	update()
44	The Graphics class can be used to draw figures and images using	X-Y coordinates	X coordinates	Y coordinates	shapes	X-Y coordinates
45	applets are flexible for using	parameters	variables	constants	shapes	parameters
46	_______ method used for screen updating	repaint()	paint()	update()	start()	repaint()
47	In applet to retrieve a parameter ______ is used	getparameter() method	getApplet	applet tag	variables	getparameter() method
48	When an applet is terminated the sequence of method calls takes place	stop() and destroy()	only stop()	only destroy()	init() and destroy()	stop() and destroy()
49	The first method called by any applet is	void init()	void()	void main()	main()	void init()
50	_______ method used to remove from view	void init()	void hide()	void main()	void show()	void hide()
51	Lines are drawn by means of	drawLine() method	drewLine() method	drawline method	drawline() method	drawLine() method
52	The method used to display an outlined is	drawRect()	Rect()	drawline method	drawline() method	drawRect()
53	To draw a rounded rectangle	drawRoundRect()	drawcircle()	drawcircleRect()	circleRect()	drawRoundRect()
54	To drawan ellipse	drawOval()	drawcircle()	drawEllipse()	drawline() method	drawOval()
55	Arcs can be drawn with	fillArc()	frameArc()	fillAngle()	frameAngle()	fillArc()
56	To draw shaped figures	drawPolygon()	drawcircle()	drawline()	drawOval()	drawPolygon()
57	The complete Applet tag of HTML is	<APPLET> </APPLET>	<APPLET /APPLET>	APPLET/APPLET>	<APPLET APPLET>	<APPLET> </APPLET>
58	Font style can be specified using the constants as	Font.BOLD	FONT.BOLD	FONT.Bold	font.bold	Font.BOLD
59	Audio clips and video clips can be played from within an	applet	web browser	applet viewer	component	applet

60	_____ can respond to events generated by users such as mouseclick and keypress	applet	web browser	applet viewer	component	applet

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2016 onwards)

DEPARTMENT OF COMMERCE (CA)

SUBJECT: : JAVA

SEMESTER : III

SUBJECT CODE: 16CCP304 **CLASS : II M.COM CA**

JAVA (16CCP304) - UNIT I

	QUESTIONS	OPT1	OPT2	OPT3	OPT4	ANSWER

	Questions	opt1	opt2	opt3	opt4	answer
1	Java is a ___________ language	structured programming	object oriented	procedural oriented	machine	object oriented
2	OOPS follows______________ approach in program design	bottom_up	top_down	middle	top	bottom_up
3	Objects take up ______________in the memory	Space	Address	Memory	bytes	Space
4	_________________is a collection of objects of similar type	Objects	methods	classes	messages	classes
5	The wrapping up of data & function into a single unit is known as _______________	Polymorphism	encapsulation	functions	data members	encapsulation
6	__________________refers to the act of representing essential features without including the background details or explanations	Encapsulation	inheritance	Dynamic binding	Abstraction	Abstraction
7	Attributes are sometimes called______________	data members	methods	messages	functions	data members
8	The functions operate on the datas are called______________	Methods	data members	messages	classes	Methods
9	______________is the process by which objects of one class acquire the properties of objects of another class	Polymorphism	encapsulation	data binding	Inheritance	Inheritance
10	__________________means the ability to take more than one form	Polymorphism	encapsulation	data binding	Inheritance	Polymorphism
11	The process of making an operator to exhibit different behaviors in different instances is known as ________________	function overloading	operator overloading	method overloading	message overloading	operator overloading
12	pow () is associated with which class	Math class	Input stream class	Object class	function name	Math class
13	x=x+1 is equivalent to	++x	x++	x=x-1	x	x++

14	All collection classes are available in	java.io package	java.lang package	java.awt package	java.util package	java.util package
15	Keyword _________ indicates that method do not return any value.	Static	Final	void	null	void
16	_________ is used to define the objects	class	functions	methods	datamembers	class
17	An _________ is a single instance of a class that retains the structure and behaivour as defined by a class	class member	object	instances	datamembers	object
18	A _________ is a message to take some action on an object	member	variable	method	class	method
19	Java does not have _________ statement	goto	if	do	do while	goto
20	Execution of the program is always begins with	Main method	class contain main method	parent class	default package	Main method
21	The ________ is the basic unit of storage in a Java program	identifier	variable	class	object	variable
22	byte belongs to _________ type.	character	Boolean	floating	integer	integer
23	In Java an int is _____ bits	16	64	52	32	32
24	byte is a signed ______ type	16 bit	8 bit	32 bit	64 bit	8 bit
25	The ________ statement is often used in switch statement	break	end	do	exit	break
26	The keywords private and public are known as _________ labels	Static	Dynamic	Visibility	const	Visibility
27	The class members that have been declared as ________ can be accessed only from within the class	Private	Public	Static	protected	Private
28	The class members that have been declared as ________ can be accessed from outside the class also	Private	Public	Static	protected	Public
29	The class variables are known as ________	Functions	members	objects	variable decleration	objects
30	Command to execute a compiled java program is	javac	java	run	execute	java
31	File produced by the java compiler contains _________	ASCII	Class	Pnemonics	ByteCodes	ByteCodes
32	The file produced by java compiler ends with _______ file extension	Java	html	class	applet	class
33	Objects are instantiated from__________	Java	methods	groups	class	class
34	Which of the following lines is not a Java comment?	/** comments */	// comments	– comments	/* comments */	– comments
35	Which of the following statements is correct?	system.out.println('Welcome to Java');	System.out.println("Welcome to Java");	System.println('Welcome to Java');	System.out.print('Welcome to Java');	System.out.println("Welcome to Java");

36	A block is enclosed inside __________.	Parentheses	Braces	Brackets	Quotes	Braces
37	Wich of the following is a correct signature for the main method?	static void main(String[] args[])	public static void main(String[] args)	public void main(String[] args)	public static void main(Strings[] args)	public static void main(String[] args)
38	The Java programming language was developed by ______	James Gosling	James	John Gosling	James Gossor	James Gosling
39	__________ translates the Java sourcecode to bytecode files that the interpreter can understand	javac	java	javap	jdk	javac
40	In java the functions are called as _________	fields	method	variables	packages	method
41	JAR file contains the compressed version of	.java file	.class file	.jsp file	.java	.class file
42	Main method parameter has which type of data type	int	char	string	double	string
43	Java interpreter is	JVM	Javac	Compiler	JAR	JVM
44	The __________ method terminates the program.	System.terminate(0);	System.halt(0);	System.exit(0);	System.stop(0);	System.exit(0);
45	Java has no ______ function.	malloc, free	calloc	super	package	malloc, free
46	Java supports __________ inheritance	single	multiple	single and multiple	extends	single
47	Java does not have _________	sturct	package	import	java.io	sturct
48	__________ is a access specifier	static	void main	public	temp	public
49	Java is a __________ type language.	Weak	strong	correct	incorrect	strong
50	Data type Short occupies _________ bytes.	1	2	4	8	2
51	Code Reusability is characterized by	baseclass	Subclass	Derived class	Inheritance	Inheritance
52	Classes are the	Members	Algorithms	Templates	Methods	Templates
53	_________ contains the executable files for the development tools contained in the JDK	bin	lib	jre	jar	bin
54	It enables us to ignore the non_essential	Inheritance	Encapsulation	Abstraction	DataBinding	Abstraction
55	It is the most powerful feature of any programming technique	top_down	bottom up	Code reusability	Security	Code reusability
56	Encapsulation is also known as	Abstraction	Information hiding	Polymorphism	Inheritence	Information hiding
57	Well defined entities that are capable of interacting with themselves	Encapsulation	Message Passing	Abstraction	Binding	Message Passing

58	The data or variables,defined within a class are called	Variables	Class variables	Data variables	Instance Variables	Instance Variables
59	Class is a _______Construct	Hierarchical	Logical	Physical	Hybrid	Logical
60	To access instance variables of an object______operator is used	Dot Operator	Logical operator	Relational Operator	Boolean Operator	Dot Operator
61	Variables declared as static are______variables	Member variables	Instance	Global	Local	Global
62	These are the foundation of encapsulation	Functions	Instance methods	procedures	operators	Instance methods
63	The java compiler	creates executable	translates java source code to byte code	create classes	produces java interpreter	translates java source code to byte code

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF COMMERCE (CA)

SUBJECT : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 CLASS : II M.COM CA

POSSIBLE QUESTIONS – UNIT I

PART A (1 Mark)
(Online Examinations)

PART B (2 Marks)

1. Give any 4 differences between C++ and Java.
2. What gives java it's "write once and run anywhere" nature?
3. How to Compile and Execute a Java Program.
4. Write short notes on i) Byte Code ii) Java Environment
5. Write a Java code for Basic Program Output.
6. Define Tokens
7. List out the operators in java.
8. Mention types of java program and explain it.
9. What are literals
10. List down the data types in Java

PART C (6 Marks)

1. Explain in detail about various control statements.
2. Discuss: Arrays in java.
3. Differentiate between Object Oriented paradigm and structured programming.
4. Explain in detail about tokens and variables in JAVA.
5. What are all the types of JAVA program, Explain about variable declaration and arrays with example.
6. Explain in detail about the features and architecture of JAVA.
7. Illustrate the working of control statements in JAVA with appropriate examples.
8. Discuss Java Methods with example program.
9. Elucidate variables in java with the use of operators each with suitable examples.
10. Write a program in JAVA to find out a given number is prime number or not.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF COMMERCE (CA)

SUBJECT : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 **CLASS : II M.COM CA**

POSSIBLE QUESTIONS – UNIT II

PART A (1 Mark)
(Online Examinations)

PART B (2 Marks)

1. How to create objects? Give an example
2. Define Method overloading
3. What do we declare a method or class abstract
4. Write a note on "this" keyword
5. Explain the use of "final" keyword
6. Explain about "super" keyword
7. Write brief on “Command Line Argument”.
8. What do you mean by a class?
9. What is an object?
10. Give two differences between overloading and overriding.

PART C (6 Marks)

1. Elucidate about the command line argument with suitable example.
2. Explain : i)this Keyword ii)finalize() method
3. Discuss the following with suitable examples.
4. Defining i)a class ii) instance variables iii) class variables iv) instance methods
5. Illustrate Method Overloading with suitable program.
6. Explain the following with suitable program i)Garbage Collection ii)final classes
7. Describe about the inheritance in java
8. What is overloading? Explain about Method Overloading with examples.
9. Explain the following: i)Static ii)final iii)Recursion
10. Compare between Method overloading and Method Overriding.
11. Explain about inheritance and its types.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)

DEPARTMENT OF COMMERCE (CA)

SUBJECT : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 CLASS : II M.COM CA

POSSIBLE QUESTIONS – UNIT III

PART A (1 Mark)
(Online Examinations)

PART B (2 Marks)

1. What is an Exception?
2. How to declare the Packages.
3. What are packages and how it is used?
4. Define and give the syntax for interface.
5. What is Interface?
6. What are Java Packages?
7. How an Exception is handled in java?
8. What is the use of finally block?
9. Write short notes on access Protection
10. What is the difference between exception and error?

PART C (6 Marks)

1. Discuss Package with suitable examples.
2. What is an Exception? Explain how to throw, catch and handle Exceptions
3. How will you declare a package and import it, Explain.
4. Clarify in detail about the types of exception with an example program to handle array out of bounds exception.
5. Clarify in detail about the basics of exception with its types and an example program
6. Discuss in detail about java.net package with example
7. Explain the significance of using packages in Java.
8. Elaborate in detail on the following.(i) Exception handling in Java
   (ii) Distinguish between Error and Exception
9. Discuss Packages and Sub-Packages.
10. Explain following keywords used in Exception Handling.
    (i) try (ii) catch (iii) throw

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)

DEPARTMENT OF COMMERCE (CA)

SUBJECT : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 **CLASS : II M.COM CA**

POSSIBLE QUESTIONS – UNIT IV

PART A(1 Mark)
(Online Examinations)

PART B (2 Marks)

1. What is Thread prioritization?
2. How Threads are created in Java?
3. What do you mean by Thread Scheduling?
4. What is the difference between preemptive scheduling and time slicing?
5. Differentiate wait and sleep methods in java?
6. Define Dead Lock. A Dead Lock
7. Write note on FileInputStream class
8. Define Multithread Programming
9. What is Synchronization?
10. What is the need of Thread Priorities?

PART C (6 Marks)

1. Explain the procedure involved in incorporating priorities for threads.
2. Explain a various types of modifiers in I/O Applets.
3. Explicate about Inter thread communication.
4. Explicate in detail about method of reading and writing files in I/O Applets.
5. Explain the use of thread methods yield(),stop() and sleep().
6. Define threads. Explain multiplication table using multithreading with suitable program.
7. Explicate about Inter thread communication.
8. Explain about the applet skeleton and order of applet initialization and termination.
9. Describe the usefulness of suspend () method, resume () method and stop () method in threading.
10. Write in detail on the following.
    (i) Reading console input (ii) Writing console output

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)

DEPARTMENT OF COMMERCE (CA)

SUBJECT : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 **CLASS : II M.COM CA**

POSSIBLE QUESTIONS – UNIT V

PART A (1 Mark)
(Online Examinations)

PART B (2 Marks)

1. Draw the life cycle of Java Applets.
2. Write any two AWT Controls.
3. Distinguish between init () and start () methods
4. Write the difference between an applet and an application.
5. What is the order of method invocation in Applet.
6. Define AWT
7. Name Component subclasses that support painting.
8. What is the difference between the 'Font' and 'FontMetrics' class?
9. What is the difference between the paint() and repaint() methods?
10. How are the elements of different layouts organized?

PART C (6 Marks)

1. Distinguish between an applet and an application
2. Explain in detail about the concept of drawing lines in java.
3. Discuss about the steps involved in running an applet
4. Explain the process involved in drawing ellipses and circle in graphics.
5. Discuss in detail about life cycle of an Applet.
6. Write a program to create an applet and draw the shape.
7. Describe on AWT color system in Java.
8. Explain the concept and problems with Native methods in I/O Applets.
9. List out and explain a various methods of the Applet Class.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF COMMERCE (CA)

SYLLABUS

Semester – III

		L	T	P	C
16CCP304	**JAVA**	4	-	-	4

Course Objectives

- ❖ To understand Object oriented concepts like data abstraction, encapsulation, etc.
- ❖ To solve the real world scenarios using top down approach.
- ❖ To understand various Java programming constructs.

Learning Outcome

On successful completion of this course the student should be able to:

- ❖ To solve computational problems using basic constructs like if-else, control structures, array, and strings.
- ❖ To implement relationships between classes
- ❖ To demonstrate various collection classes.
- ❖ to demonstrate programs on exceptions, multithreading and applets.
- ❖ To write programs that solves simple business problems.

Unit – I

An overview of Java: Object oriented programming – Java features – Java environment - Data types, variables and arrays. Operators- Expressions - Control Statements: Branching statements – Iteration statements – Jump statements – Sample java program.

Unit – II

Classes – Objects – Methods – Constructors – The this keyword – finalize () method – Overloading methods – Returning objects – Recursion – Static – Final – Nested inner classes – Command line arguments – Inheritance.

Unit – III

Packages and Interfaces: Packages – Access protection – Importing packages – Interfaces – Exception handling: Fundamentals – Exception types – Try and catch – Multiple catch – Nested try – throw – throws – finally – Build in exception.

Unit – IV

Multithread programming: Thread model –Life cycle of thread – Creating thread – Multiple threads – Thread priorities – Synchronization – Inter thread Communication – Suspending, Resuming and Stopping threads – I/O Applets, and other topics.

Unit – V

The Applet Class: Basics – Building applet code – Applet life cycle – Creating an executable applet – Designing a web page – Running the applet – Getting input from the user – Graphics programming: The graphic class – Lines and rectangles – Circles and ellipses – Using control loops in applets – Drawing bar charts.

Text Book

1. Partrick Naughton. (2002). *Java Hand Book*. New Delhi: McGraw Hill Publishing Company Limited.

Reference Books

1. Herbert Schildt. (2014).*Java Complete Reference*, 9th edition, Tata McGraw hill.
2. Permanand Mohan. (2013). *Fundamentals of Object-Oriented Programming in Java*, 1st edition, Createspace Independent Publishing
3. Balagurusamy,E. (2014). *Programming with Java*, 5th edition, Tata McGraw-hill Publishing Company limited.
4. Daniel Liang,Y. (2015). *Introduction to Java Programming*, 10th edition, Pearson Prentice Hall.

Website

1. http://www.tutorialspoint.com/java/
2. http://docs.oracle.com/javase/tutorial/java/
3. http://javabeginnerstutorial.com/core-java/
4. http://www.learnjavaonline.org/

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)

Coimbatore - 641021.

(For the candidates admitted from 2016 onwards)

DEPARTMENT OF COMMERCE(CA)

SUBJECT : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 CLASS : II M.COM CA

Unit – I

An overview of Java: Object oriented programming – Java features – Java environment - Data types, variables and arrays. Operators- Expressions - Control Statements: Branching statements – Iteration statements – Jump statements – Sample java program.

UNIT-I

1. INTRODUCTION TO JAVA

1.1 Introduction to Object Oriented Programming

Object Oriented Programming or OOP is the technique to create programs based on the real world. Unlike procedural programming, here in the OOP programming model programs are organized around objects and data rather than actions and logic. Objects represent some concepts or things and like any other objects in the real Objects in programming language have certain behavior, properties, type, and identity. In OOP based language the principal aim is to find out the objects to manipulate and their relation between each other.

Class - It is the central point of OOP and that contains data and codes with behavior. In Java everything happens within class and it describes a set of objects with common behavior. The class definition describes all the properties, behavior, and identity of objects present within that class. As far as types of classes are concerned, there are predefined classes in languages like C++ and Pascal. But in Java one can define his/her own types with data and code.

Object - Objects are the basic unit of object orientation with behavior, identity. As we mentioned above, these are part of a class but are not the same. An object is expressed by the variable and methods within the objects. Again these variables and methods are distinguished from each other as instant variables, instant methods and class variable and class methods.

Methods - We know that a class can define both attributes and behaviors. Again attributes are defined by variables and behaviors are represented by methods. In other words, methods define the abilities of an object.

Inheritance - This is the mechanism of organizing and structuring software program. Though objects are distinguished from each other by some additional features but there are objects that share certain things common. In object oriented programming classes can inherit some common behavior and state from others. Inheritance in OOP allows to define a general class and later to organize some other classes simply adding some details with the old class definition. This saves work as the special class inherits all the properties of the old general class and as a programmer you only require the new features. This helps in a better data analysis, accurate coding and reduces development time.

Abstraction - The process of abstraction in Java is used to hide certain details and only show the essential features of the object. In other words, it deals with the outside view of an object (interface).

Encapsulation - This is an important programming concept that assists in separating an object's state from its behavior. This helps in hiding an object's data describing its state from any further modification by external component. In Java there are four different terms used for hiding data constructs and these are public, private, protected and package. As we know an object can associated with data with predefined classes and in any application an object can know about the data it needs to know about. So any unnecessary data are not required by an object can be hidden by this process. It can also be termed as information hiding that prohibits outsiders in seeing the inside of an object in which abstraction is implemented.

Polymorphism - It describes the ability of the object in belonging to different types with specific behavior of each type. So by using this, one object can be treated like another and in this way it can create and define multiple level of interface. Here the programmers need not have to know the exact type of object in advance and this is being implemented at runtime.

The Java Language

1.3 Features of java

Object Oriented

To be an Object Oriented language, any language must follow at least the four characteristics.

- **Inheritance**: It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding the additional features as needed.
- **Encapsulation:** It is the mechanism of combining the information and providing the abstraction.
- **Polymorphism:** As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the methods.
- **Dynamic binding**: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

As the languages like Objective C, C++ fulfills the above four characteristics yet they are not fully object oriented languages because they are structured as well as object oriented languages. But in case of java, it is a fully Object Oriented language because object is at the outer most level of data structure in java. No stand alone methods, constants, and variables are there in java. Everything in java is object even the primitive data types can also be converted into object by using the wrapper class.

Platform Independent

The concept of Write-once-run-anywhere (known as the Platform independent) is one of the important key feature of java language that makes java as the most powerful language. Not even a single language is idle to this feature but java is closer to this feature. The programs written on one platform can run on any platform provided the platform must have the JVM.

Simple

There are various features that makes the java as a simple language. Programs are easy to write and debug because java does not use the pointers explicitly. It is much harder to write the java programs that can crash the system. Java provides the bug free system due to the strong memory management. It also has the automatic memory allocation and deallocation system.

Robust

Java has the strong memory allocation and automatic garbage collection mechanism. It provides the powerful exception handling and type checking mechanism as compare to other programming languages. Compiler checks the program whether there any error and interpreter checks any run time error and makes the system secure from crash. All of the above features make the java language robust.

Distributed

The widely used protocol like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access the files from any remote machine on the internet rather than writing codes on their local system.

Portable

The feature Write-once-run-anywhere makes the java language portable provided that the system must have interpreter for the JVM. Java also has the standard data size irrespective of operating system or the processor. These features make the java as a portable language.

Dynamic

While executing the java program the user can get the required files dynamically from a local drive or from a computer thousands of miles away from the user just by connecting with the Internet.

Secure

Java does not use memory pointers explicitly. All the programs in java are run under an area known as the sand box. Security manager determines the accessibility options of a class like reading and writing a file to the local disk. Java uses the public key encryption system to allow the java applications to transmit over the internet in the secure encrypted form. The Bytecode Verifier checks the classes after loading.

Performance

Java uses native code usage, and lightweight process called threads. In the beginning interpretation of bytecode results the performance slow but the advance version of JVM uses the adaptive and just in time compilation technique that improves the performance.

Interpreted

Java is an interpreted language as well. With an interpreted language such as Java, programs run directly from the source code.
The interpreter program reads the source code and translates it on the fly into computations. Thus, Java as an interpreted language depends on an interpreter program.
The versatility of being **platform independent** makes Java to outshine from other languages.

1.4 Java Environment

The Java Virtual Machine

- At the heart of Java's network-orientation is the Java virtual machine, which supports all three prongs of Java's network-oriented architecture: platform independence, security, and network-mobility.
- The Java virtual machine is an abstract computer.
- Its specification defines certain features every Java virtual machine must have, but leaves many choices to the designers of each implementation.
- For example, although all Java virtual machines must be able to execute Java bytecodes, they may use any technique to execute them.
- Also, the specification is flexible enough to allow a Java virtual machine to be implemented either completely in software or to varying degrees in hardware.
- The flexible nature of the Java virtual machine's specification enables it to be implemented on a wide variety of computers and devices.
- A Java virtual machine's main job is to load class files and execute the bytecodes they contain.

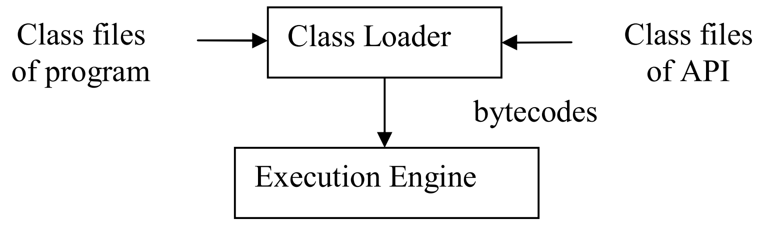


Figure A basic block diagram of the Java virtual machine.

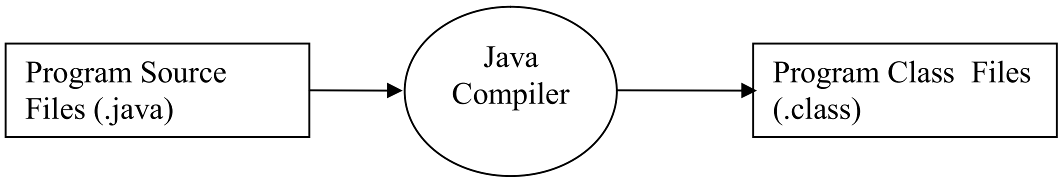
- As the program runs, the virtual machine compiles to native and optimizes just these heavily used areas.
- The rest of the code, which is not heavily used, remains as bytecodes which the virtual machine continues to interpret.
- Sometimes the Java virtual machine is called the Java interpreter; however, given the various ways in which bytecodes can be executed, this term can be misleading.
- While "Java interpreter" is a reasonable name for a Java virtual machine that interprets bytecodes, virtual machines also use other techniques (such as just-in-time compiling) to execute bytecodes.
- Therefore, although all Java interpreters are Java virtual machines, not all Java virtual machines are Java interpreters.

1.5 Java Architecture

Java's architecture arises out of four distinct but interrelated technologies:

- The Java programming language
- The Java class file format
- The Java Application Programming Interface
- The Java virtual machine

When you write and run a Java program, you are tapping the power of these four technologies. The source files written in the Java programming language, compile the source to Java class files, and run the class files on a Java virtual machine. When you write your program, you access system resources (such as I/O, for example) by calling methods in the classes that implement the Java Application Programming Interface, or Java API. As your program runs, it fulfills your program's Java API calls by invoking methods in class files that implement the Java API. You can see the relationship between these four parts in Figure 1-1.



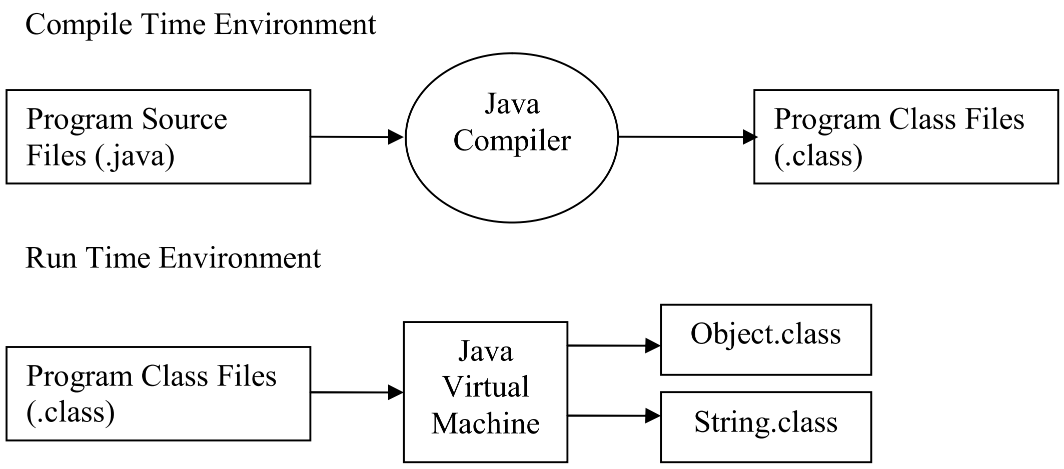


Figure 1-1. The Java programming environment.

Together, the Java virtual machine and Java API form a "platform" for which all Java programs are compiled. In addition to being called the Java runtime system, the combination of the Java virtual machine and Java API is called the Java Platform (or, starting with version 1.2, the Java 2 Platform). Java programs can run on many different kinds of computers because the Java Platform can itself be implemented in software. As you can see in Figure, a Java program can run anywhere the Java Platform is present.

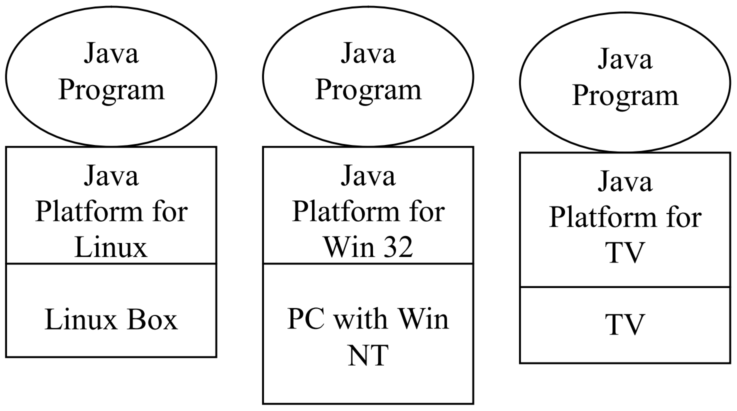


Figure Java programs run on top of the Java Platform.

1.6 Java development Kit

The **Java Development Kit** (**JDK**) is a Sun Microsystems_product aimed at Java developers. Since the introduction of Java, it has been by far the most widely used Java SDK. On 17 November 2006, Sun announced that it would be released under the GNU General Public License (GPL), thus making it free software. This happened in large part on 8 May 2007; Sun contributed the source code to the OpenJDK.

The JDK has as its primary components a collection of programming tools, including:

- java – the loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, no longer comes with Sun JDK.
- javac – the compiler, which converts source code into Java bytecode

- appletviewer – this tool can be used to run and debug Java applets without a web browser

1.7 Types of java program

Broadly Java programs can be categorized in two main groups

- **Applets**
- **Applications**

Applets

It is a small program embedded in a web page and is run when that page is browsed using a web browser. Applets are graphical in nature and tend to contain controls such as buttons, labels, text fields etc.,

Applications

These are stand-alone programs written in Java. They are invoked using a JVM which resides within a local operating system. Unlike Applets, applications can access the local file system or establish connections with other machines on the network.

A Sample Java Program

Write source code

The following Java program is developed under Microsoft Windows. The process on other operating system should be similar but will not be covered here. Select a directory which should contain your code. I will use the directory c:\temp\java which will be called "javadir".

Open a text editor which supports plain text, e.g. notepad under Windows and write the following source code. You can start notepad via Start->Run-> notepad and pressing enter.

```
// The smallest Java program possible
```

```
public class HelloWorld {
        public static void main(String[] args) {
                System.out.println("Hello World");
        }
}
```

Save the source code in your directory "javadir" under the name "HelloWorld.java". The name of a Java source file must always equals the class name (within the source code) and end with .java. In our case the filename must be HelloWorld.java because the class is called HelloWorld.

Compile the code

Switch to the command line, e.g. under Windows Start-> Run -> cmd. Switch to the "javadir" directory with the command cd javadir, for example in my case cd c:\temp\java. Use the command dir to see that the source file is in the directory.

Type **javac HelloWorld.java** .

Check the content of the directory with the command "dir". The directory contains now a file "HelloWorld.class". If you see this file you have successfully compiled your first Java source code into byte-code.

Run the code

Switch again to the command line, e.g. under Windows Start-> Run -> cmd. Switch to the directory jardir.

Type **java HelloWorld** .

The system should write "Hello World" on the command line.

Using the classpath

You can use the classpath to run the program from another place in your directory. Switch to the command line, e.g. under Windows Start-> Run -> cmd. Switch to any directory you want.

Type **java HelloWorld** .

If you are not in the directory in which the compiled class is stored then the system should result an error message **Exception in thread "main" java.lang.NoClassDefFoundError: test/TestClass**

Type **java -classpath "mydirectory" HelloWorld** . Replace "mydirectory" with the directory which contains the test directory. You should again see the "HelloWorld" output.

Variable Declarations and Arrays

1.8 Data type

Java programming language is a language in which all the variables must be declared first and then to be used. That means to specify the name and the type of the variable. This specifies that Java is a strongly-typed programming language. Like

```
int pedal = 1;
```

This shows that there exists a field named 'pedal' that holds a data as a numerical value '1'. The values contained by the variables determines its data type and to perform the operations on it.

There are seven more primitive data types which are supported by Java language programming in addition to int. A primitive data type is a data type which is predefined in Java. Following are the eight primitive data types:

int

It is a 32-bit signed two's complement integer data type. It ranges from -2,147,483,648 to 2,147,483,647. This data type is used for integer values. However for wider range of values use **long**.

byte

The byte data type is an 8-bit signed two's complement integer. It ranges from -128 to127 (inclusive). We can save memory in large arrays using byte. We can also use byte instead of int to increase the limit of the code.

short

The short data type is a 16-bit signed two's complement integer. It ranges from -32,768 to 32,767. **short** is used to save memory in large arrays.

long

The long data type is a 64-bit signed two's complement integer. It ranges from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Use this data type with larger range of values.

float

The float data type is a single-precision 32-bit IEEE 754 floating point. It ranges from 1.40129846432481707e-45 to 3.40282346638528860e+38 (positive or negative). Use a float (instead of double) to save memory in large arrays. We do not use this data type for the exact values such as currency. For that we have to use java.math.BigDecimal class.

double

This data type is a double-precision 64-bit IEEE 754 floating point. It ranges from 4.94065645841246544e-324d to 1.79769313486231570e+308d (positive or negative). This data type is generally the default choice for decimal values.

boolean

The boolean data type is 1-bit and has only two values: **true** and **false**. We use this data type for conditional statements. true and false are not the same as True and False. They are defined constants of the language.

char

The char data type is a single 16-bit, unsigned Unicode character. It ranges from 0 to 65,535. They are not same as ints, shorts etc.

The following table shows the default values for the data types:

Keyword	**Description**	**Size**
byte	Byte-length integer	8-bit
short	Short integer	16-bit
int	Integer	32-bit
long	Long integer	64-bit
float	Single-precision floating point	32-bit
double	Double-precision floating point	64-bit
char	A single character	16-bit
Boolean	A Boolean value	true or false

When we declare a field it is not always essential that we initialize it too. The compiler sets a default value to the fields which are not initialized which might be zero or null. However this is not recommended.

1.9 Java Tokens

In a Java program, all characters are grouped into symbols called **tokens**. Larger language features are built from the first five categories of tokens (the sixth kind of token is recognized, but is then discarded by the Java compiler from further processing). We must learn how to identify all six kinds of tokens that can appear in Java programs. In EBNF we write one simple rule that captures this structure:

token <= identifier | keyword | separator | operator | literal | comment

The different types of Tokens are:

1. Identifiers: names the programmer chooses
2. Keywords: names already in the programming language
3. Separators (also known as punctuators): punctuation characters and paired-delimiters
4. Operators: symbols that operate on arguments and produce results
5. Literals (specified by their **type**)
    - Numeric: **int** and **double**
    - Logical: **boolean**
    - Textual: **char** and **String**
    - Reference: **null**
6. Comments
    - Line

1.10 Variable Declaration

Declaring variables

A variable is a container that stores a meaningful value that can be used throughout a program. For example, in a program that calculates tax on items you can have a few variables - one variable that stores the regular price of an item and another variable that stores the total price of an item after the tax is calculated on it. Variables store this information in a computer's memory and the value of a variable can change all throughout a program.

Java data types		
Keyword	**Type of data the variable will store**	**Size in memory**
boolean	true/false value	1 bit
byte	byte size integer	8 bits
char	a single character	16 bits
double	double precision floating point decimal number	64 bits
float	single precision floating point decimal number	32 bits
int	a whole number	32 bits
long	a whole number (used for long numbers)	64 bits
short	a whole number (used for short numbers)	16 bits

Example:

char aCharacter; int aNumber;

You can assign a value to a variable at the same time that it is declared. This process is known as **initialization**:

1.11 Type Casting and Conversion

It is sometimes necessary to convert a data item of one type to another type. For example when it is necessary to perform some arithmetic using data items of different types (so called mixed mode arithmetic). Under certain circumstances Type conversion can be carried out automatically, in other cases it must be "forced" manually (explicitly).

In Java type conversions are performed automatically when the type of the expression on the right hand side of an assignment operation can be safely promoted to the type of the variable on the left hand side of the assignment. Thus we can safely assign: byte -> short -> int -> long -> float -> double The -> symbol used here should be interpreted as "to a". For example:

```
// 64 bit long integer
long myLongInteger;
// 32 bit standard integer
int myInteger;
myLongInteger = myInteger;
```

The extra storage associated with the long integer, in the above example, will simply be padded with extra zero.

1.12 Arrays

By definition, array is the static memory allocation. It allocates the memory for the same data type in sequence. It contains multiple values of same types. It also stores the values in memory at the fixed size. Multiple types of arrays are used in any programming language such as: one - dimensional, two - dimensional or can say multi - dimensional.

Declaration of an array:

int num[]; or int num = new int[2];

Sometimes user declares an array and its size simultaneously. You may or may not be define the size in the declaration time. such as:

int num[] = {50,20,45,82,25,63};

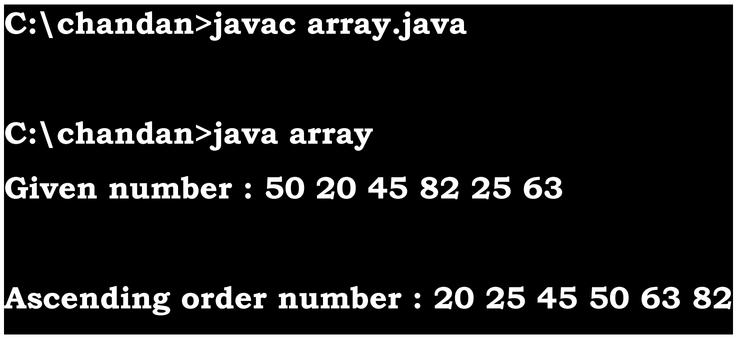
In this program we will see how to declare and implementation. This program illustrates that the array working way. This program takes the numbers present in the num[] array in unordered list and prints numbers in ascending order. In this program the sort() function of the java.util.*; package is using to sort all the numbers present in the num[] array. The Arrays.sort() automatically sorts the list of number in ascending order by default. This function held the argument which is the array name num.

Here is the code of the program:-

```
import java.util.*;
public class  array{
  public static void main(String[] args){
  int num[] = {50,20,45,82,25,63};
  int l = num.length;
  int i,j,t;
  System.out.print("Given number : ");
  for (i = 0;i < l;i++ ){
  System.out.print("  " + num[i]);
  }
  System.out.println("\n");
  System.out.print("Ascending order number : ");
  Arrays.sort(num);
    for(i = 0;i < l;i++){
```

```
  System.out.print("  " + num[i]);
  }
  }
}
```

Output of the program:



```
C:\chandan>javac array.java


C:\chandan>java array
Given number : 50 20 45 82 25 63


Ascending order number : 20 25 45 50 63 82
```

1.13 Operators and Control Statements

Operators

- Arithmetic operators
- Increment and Decrement operators
- Relational operators
- Logical operators
- Assignment operators
- Ternary operators
- Bitwise operators

Operators are special symbols that perform specific operations on one, two, or three operands, and then return a result.

Operator Precedence

Operators	**Precedence**
postfix	expr++ expr--
unary	++expr --expr +expr -expr ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

In general-purpose programming, certain operators tend to appear more frequently than others; for example, the assignment operator "=" is far more common than the unsigned right shift operator ">>>". With that in mind, the following discussion focuses first on the operators that we're most likely to use on a regular basis, and ends focusing on those that are less common. Each discussion is accompanied by sample code that you can compile and run. Studying its output will help reinforce what we have just learned.

Control Statements

Different types of control statements: the decision making statements (if-then, if-then-else and switch), looping statements (while, do-while and for) and branching statements (break, continue and return).

The control statements are used to control the flow of execution of the program . This execution order depends on the supplied data values and the conditional logic. Java contains the following types of control statements:

1. Selection Statements
2. Repetition Statements
3. Branching Statements

Selection constructs

1. **If Statement:** This is a control statement to execute a single statement or a block of code, when the given condition is true and if it is false then it skips **if** block and rest code of program is executed .

   Syntax:

   ```
   if(conditional_expression){
      <statements>;
      ...;
      ...;
   }
   ```

   Example: If n%2 evaluates to 0 then the "if" block is executed. Here it evaluates to 0 so if block is executed. Hence "**This is even number**" is printed on the screen.

```
int n = 10;
 if(n%2 = = 0){
      System.out.println("This is even number");
 }
```

2. **If-else Statement:**

The "**if-else**" statement is an extension of if statement that provides another option when 'if' statement evaluates to "false" i.e. else block is executed if "**if**" statement is false.

Syntax:

```
if(conditional_expression){
      <statements>;
      ...;
      ...;
 }
else{
      <statements>;
      ....;
      ....;
}
```

Example: If n%2 doesn't evaluate to 0 then else block is executed. Here n%2 evaluates to 1 that is not equal to 0 so else block is executed. So "**This is not even number**" is printed on the screen.

```
int n = 11;
 if(n%2 = = 0){
      System.out.println("This is even number");
 }
 else{
      System.out.println("This is not even number");
 }
```

3. **Switch Statement:**

This is an easier implementation to the if-else statements. The keyword "**switch**" is followed by an expression that should evaluates to byte, short, char or int primitive data types ,only. In a switch block there can be one or more labeled cases. The expression that creates labels for the case must be unique. The switch expression is matched with each case label. Only the matched case is executed ,if no case matches then the default statement (if present) is executed.

Syntax:

```
    switch(control_expression){
        case expression 1:
            <statement>;
        case expression 2:
            <statement>;
           …
           …
        case expression n:
            <statement>;
        default:
            <statement>;
    }//end switch
```

Example: Here expression "day" in switch statement evaluates to 5 which matches with a case labeled "5" so code in case 5 is executed that results to output **"Friday"** on the screen.

```
int day = 5;
 Switch (day) {
  case 1:
  System.out.println("Monday");
             break;
  case 2:
  System.out.println("Tuesday");
            break;
 case 3:
 System.out.println("Wednesday");
           break;
 case 4:
 System.out.println("Thursday");
           break;
 case 5:
 System.out.println("Friday");
          break;
 case 6:
 System.out.println("Saturday");
          break;
 case 7:
 System.out.println("Sunday");
```

```
            break;
    default:
             System.out.println("Invalid entry");
             break;
    }
```

Iteration Constructs

While Loop Statements:

This is a looping or repeating statement. It executes a block of code or statements till the given condition is true. The expression must be evaluated to a boolean value. It continues testing the condition and executes the block of code. When the expression results to false control comes out of loop.

Syntax:

```
while(expression){
    <statement>;
    ...;
    ...;
}
```

Example: Here expression i<=10 is the condition which is checked before entering into the loop statements. When i is greater than value 10 control comes out of loop and next statement is executed. So here i contains value "1" which is less than number "10" so control goes inside of the loop and prints current value of i and increments value of i. Now again control comes back to the loop and condition is checked. This procedure continues until i becomes greater than value "10". So this loop prints values 1 to 10 on the screen.

```
int i = 1;
```

```
//print 1 to 10
while (i <= 10){
   System.out.println("Num " + i);
   i++;
}
```

Do-While Loop Statements:

This is another looping statement that tests the given condition past so you can say that the do-while looping statement is a past-test loop statement. First the **do** block statements are executed then the condition given in **while** statement is checked. So in this case, even the condition is false in the first attempt, do block of code is executed at least once.

Syntax:

```
do{
   <statement>;
      ...;
      ...;
}while (expression);
```

Example: Here first do block of code is executed and current value "1" is printed then the condition i<=10 is checked. Here "1" is less than number "10" so the control comes back to do block. This process continues till value of i becomes greater than 10.

```
int i = 1;
 do{
    System.out.println("Num: " + i);
    i++;
}while(i <= 10);
```

For Loop Statements:

This is also a loop statement that provides a compact way to iterate over a range of values. From a user point of view, this is reliable because it executes the statements within this block repeatedly till the specified conditions is true .

Syntax:

```
for (initialization; condition; increment or decrement){
    <statement>;
    ...;
    ...;
}
```

initialization: The loop is started with the value specified.
condition: It evaluates to either 'true' or 'false'. If it is false then the loop is terminated.
increment or decrement: After each iteration, value increments or decrements.

Example: Here num is initialized to value "1", condition is checked whether num<=10. If it is so then control goes into the loop and current value of num is printed. Now num is incremented and checked again whether num<=10.If it is so then again it enters into the loop. This process continues till num>10. It prints values 1 to10 on the screen.

```
for (int num = 1; num <= 10; num++){
        System.out.println("Num: " + num);

  }
```

Jump Statements

Break statements:

The break statement is a branching statement that contains two forms: labeled and unlabeled. The break statement is used for breaking the execution of a loop (while, do-while and for). It also terminates the switch statements.

Syntax:

```
break;    // breaks the innermost loop or switch statement.
break label;   // breaks the outermost loop in a series of nested loops.
```

Example: When if statement evaluates to true it prints "data is found" and comes out of the loop and executes the statements just following the loop.

```
int num[]= {2,9,25,4,16};
int search=4;
for (int i=1; i<num.length;i++)
{
if num[i]==search)
{
System.out.println("data is found");
break;
}
}
```

Continue statements:

This is a branching statement that are used in the looping statements (while, do-while and for) to skip the current iteration of the loop and resume the next iteration .

Syntax:

```
        continue;
```

Example:

```
int num[]={2,9,1,4,25,50)
int search=4;
for(int i=1; i<num.length,i++){
if (num[i]!=search){
continue;
}
if(found==search){
System.out.println("data is found");
break;
}
}
```

Return statements:

It is a special branching statement that transfers the control to the caller of the method. This statement is used to return a value to the caller method and terminates execution of method. This has two forms: one that returns a value and the other that cannot return. The returned value type must match the return type of method.

Syntax:

```
        return;
```

return values;

return; **//**This returns nothing. So this can be used when method is declared with void return type.
return expression; //It returns the value evaluated from the expression.

Example: Here Welcome() function is called within println() function which returns a String value "Welcome to roseIndia.net". This is printed to the screen.

```
public static void Hello(){
System.out.println("Hello" + Welcome());
}
static String Welcome() {
return "Welcome to RoseIndia.net";
}
```

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF COMMERCE(CA)

SUBJECT : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 CLASS : II M.COM CA

Unit – II

Classes – Objects – Methods – Constructors – The this keyword – finalize () method – Overloading methods – Returning objects – Recursion – Static – Final – Nested inner classes – Command line arguments – Inheritance.

UNIT - II

Introduction to Classes

2.1 Introduction to Classes

- A class is a template that defines a type of object.
- Many objects can be constructed from a single class architecture.
- Class outlines the properties of an object.
- Objects created from the same class have similar characteristics.
- Class is a collection of data variables and methods.
- Java program must be included within class
- A program may contain any no. of classes.
- Class can be user-defined or built-in java packages.

Defining a class

- Class is defined using a keyword class followed by a user-defined class name

```
Class classname
{
[variable declarations;]
[method declarations;]
}
```

- The data or variable defined within a class is called instance variables.

(eg)

```
class Exampleclass                          //class name
{
  char c;                                 //data item
  int i;
  double dd;
  void exampleMethod1()                  //method
{
   System.out.println("Hello World");
}
 void exampleMethod2()
{
   System.out.println("Hello HITEC");
}
}
```

A class definition includes the following

Access Modifier - specifies the accessibility of class from other classes

Class Keyword Instance Fields – variables & constants that are used by objects of the class.

Constructors - Methods having the same as class, used to control the initial state of any class object.

Instance Methods – Define the function that can act upon objects in this class.

Class Fields – Variables & constants that belong to the class and are shared by all objects of particular objects.

Class Method - Methods that belong to the class and are used to control the values of class fields.

Class as Datatype

- Java has two types of data and two categories of variables.
- The datatype are primitive and non-primitive.
- Variables can either contain primitive values pr refer to objects.

2.2 Instance Variables

- These variables are declared in the class body.
- Instance variables belong to particular object.
- When an object is created, java automatically performs default initialization for all instance variables.

(eg)

```
class Movie
{
string title;
string rating;
double cost;
int numOfAwards=0;
.
}
```

- Title, rating, cost etc are instance variables.
- Each of them is created for every object of the class Movie.
- To access instance variable of an object use dot(.)operator

Syntax:

Object_name.instance_variable

2.3 Class Variables

- Instance variables are created every time when a new object is created and accessed with dot operator.
- If we want to define a member that is common to all objects and is accessed without using a particular object, static keyword is used.
- Members declared as static belong to the class as a whole rather than object.
- Class/Static variables can be accessed by using

  Class_name.class_variable

  (or)

  Object_name.class_variable

  (eg)

  ```
  Class Movie
    {
      -------
      -------
     static int numMovies;
    static double mincost=1.5;
      ------
  ```

```
            -------
        }
```

- Class variables are initialized when the class is loaded.

2.4 Instance Methods

- Java method is equivalent to function, procedure or subroutine in other language.
- Adding methods to classes
- Methods are declared inside the body of the class but immediately after the declaration of the instance variables & class variables.

```
returntype  methodname(parameter_list)
{
   method_body;
}
```

- returntype can be primitive type such as int or class type such as string or void.
- methodname begins with lowercase and compound words begins with uppercase.
- parameter_list is optional separated by commas.
- Return Value:
- To return a value, return statement is used which takes a single value or expression

(eg)

```
      String gettitle()
{
   return title;
}
  void printDetails()
{
 System.out.println("Title is:"+title);
System.out.println("Rating is:"+rating);
}
```

Method Accepting Parameters:

- A method can have 0 or any no. of arguments.
- Parameters list is always enclosed in parenthesis.

(eg)

```
 Void setDetails(string ntitle,string nrating,double ncost)
      {
        title=ntitle;
        rating=nrating;
        cost=ncost;
     }
```

2.5 Constructors

- Constructors are used to initialize member variables when object is created.
- Constructors also perform many operations like allocating memory block, opening files etc when an object is created.
- Constructor name is same as class name & do not have return type.

```
classname(parameter_list)
{
  constructor_body;
}
```

- If no constructors are declared in a class the compiler creates a default constructor

```
classname()
{
}
```

(eg)

```
     Movie()
     {
      System.out.println("A new movie object is created:");
    }
    Movie(string r)
    {
     Rating=r;
    }
```

2.6 Class Methods

- A class static method is similar to a class variable in that it is assigned to a class and not an object of that class.

```
Static returntype class_method_name(parameter_list)
{
 Method_body;
}
```

- To invoke a static method from another class, Class_name.class_method_name(parameter_list)
- Class methods can also be called by using an object reference before the dot operator as in the class if class variables Object_name.class_method_name(parameter_list)

(eg)

```
    Class Movie
    {
     -----------
    ------------
   static void printTotalMovies()
   {
    System.out.println("Total no. of movies are:"+Movie.numMovie);
  }
```

```
   -----------
   ------------
  }
```

The main() Method

- The main() method is an example of a class method.
- It is declared static because it is called by java runtime system before any objects are created.
- public static void main(String args[])
- Argument is a multi-dimensional string array which accepts any command line argument.
- The name argument can be any variable name that is not used within the body of the method.

2.7 Declaring Objects

- Each instance of the class (i.e.) each object of the class contains its own copy of the variables.
- Class declaration doesn't create an actual object and hence doesn't copy memory.
- To create objects of a class new operator is used.
- The new operator
- All class objects are created by using a new keyword followed by a constructor.
- The new operator performs the following functions
- Allocates memory for the new object.
- Calls the constructor to initialize the instance variables with the new object.
- Returns the reference of the newly created object.

Class_name.object_name;

Object_name=new constructor;

(or)

Class_name.object_name=new constructor;

(eg)

```
Class Movie3
{
string title;
string rating;
double cost;
int numOfAwards=0;
static int nummovies;
static double mincost=1.5;
static
{
System.out.println("Static Block");
mincost=1.27;
}
Movie(String ntitle)
{
numMovies++;
title=ntitle;
}
String gettitle()
{
return title;
}
void printDetails()
{
System.out.println("Title is:"+title);
System.out.println("Rating is:"+rating);
}
void setDetails(string ntitle,string nrating,double ncost)
{
```

```
Title=ntitle;
Rating=nrating;
Cost=ncost;
}
Static void printTotalMovies()
{
System.out.println("Total no. of movies are:"+Movies.numMovies);
public static void main(String args[])
{
Movie m1=new Movie("The Last Resort");
Movie m2=new Movie("Ice Age");
Movie.printTotalMovies();
String t=m1.gettitle();
System.out.println("Title of m1 is:"+t);
m2.setDetails("Dil Mangae More","PG",7.5);
m2.printDetails();
}
```

Object Reference Variables

- Object reference variables are declared in java with a statement that gives the type of object to when variables expected to refer.

  Class_name anyRef;

  (eg)

  String myString;

2.8 Garbage Collection

- It is a process that handles the memory deallocation.
- When an object is created memory is allocated for the subject.
- When there is no more reference to that object it is marked for garbage collection.

- When the garbage collector runs it searches for the marked memory and free it.
- To force an object to be marked for garbage collection simply remove all references to that object by setting it to null.

```
Movie m1=null;
Movie m2=null;
```

- JVM performs garbage collection in a low priority thread.
- When JVM has nothing else to do the garbage collector takes some CPU time.
- System.gc() method can be invoked to explicitly request the garbage collection.
- The Finalise() Method

  The method helps to optimize the disposing of an object.

```
finalize()
 {
  ----
  ----
 }
```

2.9 Method Overloading

- It enables to send different types of parameters to a method.
- It defines a method with one set of parameters.
- To overload that declare and define another versions with a different set of parameters.
- In java it is possible to overload methods that have the same name, provided they have different signatures.
- A method signature is formed by its name, with the no. & types of its arguments.
- It is used when objects are required to perform a similar task by using different parameters.

- When an object is called java matches the name, no. & type of parameters. This process is called as polymorphism.

(eg)

```
  Class Movie
  {
  double cost;
  void setCost()
 {
 cost=3.50;
 }
void setCost(double ncost)
{
cost=ncost;
}
public static void main(String args[])
{
Movie m1=new Movie();
m1.setCost();
m1.setCost(5.25);
}
}
```

2.10 Constructor Overloading

- If constructor is overloaded each one must have different signature.
- Constructor always have the name of the class and no return type.

(eg)

```
  Class Movie
  {
  String title;
  String rating;
```

```
 double cost;
 Movie()
 {
 System.out.println("A new movie object created");
 Movie(string ntitle)
 {
 title=ntitle;
 }
 Movie(String ntitle, String nrating,double ncost)
 {
 title=ntitle;
 rating=nrating;
cost=ncost;
}
void printDetails()
{
System.out.println("Title is:"+title);
System.out.println("Rating is:"+rating);
System.out.println("Cost is:"+cost);
}
}
Class test2
{
public static void main(String args[])
{
Movie m1=new Movie();
Movie m2=new Movie("The Last Resort");
Movie m3=new Movie("Ice Age","G",10.25);
m1.printdetails();
m2.printDetails();
m3.printDetails();
}
}
```

2.11 The 'this' Reference

- All instance method receives an implicit argument called this , which can be used inside any method to refer to the current object.
- Current object is the object on which the method was called.
- When need to pass a reference to the current object as an argument to another method.
- When we need to call a constructor from another constructor.

(eg)

```
Class Movie
{
String title;
String rating;
double cost;
void setDetails(Sstring ntitle,String nrating,double ncost)
{
this.title=title;
rating=nrating;
cost=ncost;
}
}
class test3
 {
 public static void main(String args[])
 {
 Movie m1=new Movie();
 m1.setDetails("Paradise Lost","PG",7.6);
 }
 }
```

Calling constructor from another constructor

```
Class Movie
 {
 String title;
 String rating;
 double cost;
 Movie(String ntitle)
 {
 this(ntitle);
 rating=nrating;
 cost=ncost;
 }
 }
 Class test4
 {
 public static void main(String a[])
 {
 Movie m1=new Movie("The Sixth Sense");
 Movie m2=new Movie("Pride and Prejudice","G",4.5);
 }
 }
```

2.12 Using Objects in Method

Objects as parameters

- When a primitive value is passed into a method a copy of its value is passed into the method argument.

- If the method changes the value of the argument , only the local argument is affected. when method terminates local argument is discarded.

```
int num=150;
obj.amethod(num);
System.out.println("num="+num);
public void amethod(int a)
{
If(a<0||a>100)
a=0;
System.out.println("arg:"+a);
}
```

- This manner of method calling is called call-by-value.

Call-by-reference

```
Class Movie
{
---------
----------
Movie(String ntitle,String nrating,double ncost)
{
title=ntitle;
rating=nrating;
cost=ncost;
}
}
Class test5
{
public static void main(String a[]);
{
```

```
Movie m1=new Movie(“Duplicate”,”G”,5.75);
System.out.println(“Original title is:”+m1.title);
xyz(m1);
System.out.println(“New title is:”+m1.title);
}
Static void xyz(Movie X)
{
x.title=”Behnur”;
}
}
```

- This method of calling is called call-by-reference.

Returning objects

- Method can return primitive datatype or an object.

(eg)

```
Class test
{
int a;
test(int i)
{
a=i;
}
test incrbyten()
{
test temp=new test(a+10);
return temp;
}
}
Class Retob
```

```
{
public static void main(String args[])
 {
 test ob1=new test(2);
 test ob2;
 ob2=ob1.incrbyten();
 System.out.println("ob1.a:"+ob1.a);
System.out.println("ob2.a:"+ob2.a);
}
}
```

2.13 Recursion

- Recursion is a process in which functions calls itself repeatedly until some specific condition is satisfied.

(eg)

```
Class factorial
{
int num;
int fact(int n) {
If(n==0)
return(1);
num=n*fact(n-1);
return num;
}}
Class factrecur
{
public static void main(String args[]){
int result;
factorial f1=new factorial();
result=f1.fact(n);
System.out.println(“Factorial=”+result);
}}
```

2.14 Access Modifiers

- Access to member variables and methods in a java class is accomplished three access modifiers.
- There are four access modifiers

  Default

  Private

  Protected

  Public
- Only classes in the same physical file or compilation unit have access to member variables and methods with default access specifier.
- Most restrictive of all the access modifiers.
- Variables and methods are accessible only within the class it is declared.
- Variables and methods are accessible only in the class in which they are declared and also in the classes derived from it(subclass).
- Last restrictive of all the access modifiers
- Variables and methods of all the accessible both inside and outside the class in which they declared.

2.15 Command Line Arguments

- Parameters are passed at the time of invoking it for execution.
- Any argument passed through command line as passed to array args[].
- Java programs that can receive and use the arguments provided in the command line

```
        public static void main(String args[])
```

(eg)

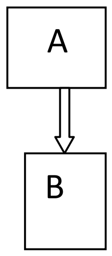
```
class test
{
public static void main(String args[])  {
  int cnt,i=0;
  String str;
  cnt=args.length;
  System.out.println("No. of arguments="+cnt)
 while(i<cnt) {
 str=args[i];
i=i+1;
System.out.println(i+"argument is"+str)
}}}
```

Inheritance

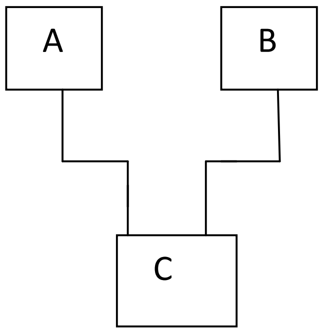
2.16 Basics Of Inheritance

- Reusability is one of the most important aspects of OOP paradigm.
- A class that is inherited is called super class or parent class or base class.
- The class that does inheriting is called sub class or child class or derived class.
- A class can be built on another class that is already defined and that exists.
- The child class inherits all the properties if the parent class (i.e.) it inherits all the member variables and methods in the parent class.

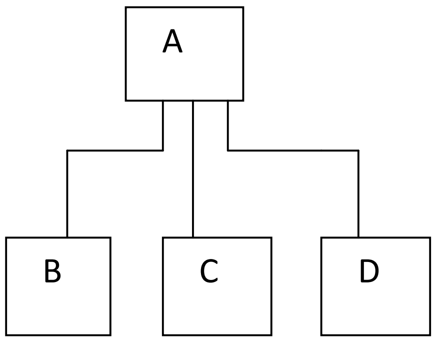
Single Inheritance



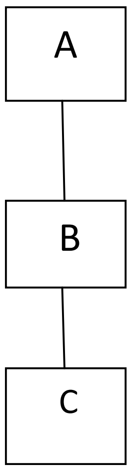
Multiple Inheritance



Hierarchical Inheritance



Multilevel Inheritance



- A classes in java can only extend one other class (i.e.) a class can have only one immediate super class called linear inheritance.
- Multiple inheritances were not supported in java.

Defining subclass

- The root class for all java object is java.lang.object class.

```
Class classname extends parameters
{
// body of the child class
}
```

(eg)

```
        public class inherit1
        {
        public static void main(String args[])
        {
        childclass ch=new childclass();
        System.out.println("ch.pi="+ch.pi);
        System.out.println("ch.ci="+ch.ci);
        ch.parentMethod();
        ch.childMethod();
```

```
}
}
class childclass extends parentclass
{
int ci;
Childclass()
{
ci=100;
}
void childMethod()
{
System.out.println("HELLO HITEC");
}
}
class Parentclass
 {
 int pi;
 Parentclass()
 {
 pi=10;
 }
 void parentMethod()
 {
 System.out.println("Hello World");
 }
 }
```

Member Access and Inheritance

- A subclass includes all if the members of its superclass but it cannot access those members if they are declared as private.

(eg)

```
class A
{
int i;
private int j;
void getData(int x,int y)
{
i=x;
j=y;
}
}
class B extends A
{
int total;
void sum()
{
Total=i+j;
}
}
class inherit2
{
public static void main(String args[])
{
B obj=new B();
obj.getData(10,12);
obj.sum();
System.out.println("Total="+obj.total);
}
}
```

2.17 Superclass Variable And Subclass Object

- A reference variable of a superclass can be assigned to any subclass derived from that superclass.
- The rule for reference values is that conversions up the inheritance are permitted(upcasting),but conversions down the hierarchy require explicit casting(downcasting).
  (<destination_type>) <reference>
- Casting requires compile time and runtime checks.

(eg)

```
class A{}
class B extends A{}
class inherits 3
{
public static void main(String args[])
{
A a1=new A();
B b1=new B();
a1=b1;
b1=(B)a1;
A a2=new A();
B b2=new B();
b2=(B)a2;
```

- When objects are compared references a2 refers to the object of the class A, which is parent of class B object of superclass cannot be assigned to object of subclass.

-

2.18 The Super Reference

- Super reference is useful only when a class has a parent class.
- The super keyword refers to the immediate parent class object.
- The super has two forms
- Super(parameter_list);- This method calls the superclass constructor.
- Super_membername;-It is used to access a member of the superclass that has been hidden by a member of a subclass.

Calling superclass constructors

- Subclass constructor is used to construct the instance variables if both the subclass & superclass.
- Super() invokes the constructor of the superclass.
- The parameters on the super() call must match the order and type of parameters declared in the superclass constructor.

(eg)

```
public class employee
{
int empno;
String name;
String address;
Employee(int e, String n, String a)
{
empno=e;
name=n;
address=a;
}
}
class worker extends employee
{
```

```
int noofhrs;
Worker(int c, String n, String a, int h)
{
super(e,n,a);
Noofhrs=h;
}
}
```

Calling superclass Members

- The second form always refer to the members of the superclass of the subclass in which it is used.

(eg)

```
class A
{
int i;
}
class B extends A
{
int i;
B(int a, int b)
{
super.i=a;
i=b;
}
}
class inherit 5
{
public static void main(String args[])
 {
 B b1=new B(1,2);
 System.out.println(b1.i);
 }
 }
```

2.19 Method Overriding

- A subclass inherits all the methods of its superclass
- Subclass can modify the behavior of a method in a superclass by overriding it
- To override a superclass method, subclass defines a method with exactly the same signature and return type.

(eg)

```
 public class inherit 8
 {
 public static void main(String args[])
 {
 Parentclass pa=new Parentclass();
 pa.parentMethod();
 childclass ch=new childclass();
ch.parentMethod();
ch.childMethod();
}
}
class Parentclass
 {
void parentMethod()
{
System.out.println("Hello Parent");
}
}
```

```
class Childclass extends Parentclass
{
void childMethod()
{
System.out.println("Hello Child");
}
void parentMethod()
{
System.out.println("Hello parent in the child class");
}
}
```

Differences between Overloading and Overriding

Overloading	**Overriding**
➢ Class can have more than one method with same name but differ in no, type, and order of the i/p parameter. ➢ Resolved at couple time.	➢ In subclass method has same name, parameter list and return type as superclass method. ➢ Resolved at runtime.

2.20 The final keyword

Final variables

- ➢ If a variable is declared as final modifications is not possible.

  Final double pi=3.14;

Final Methods

- ➢ All methods can be overridden by default in the subclass
- ➢ To prevent from overriding the members of the superclass are declared using the keyword final.

  Final void disp() {-------}

(eg)

```
        Class A
        {
        Final void show()
    {
    System.out.println("This is final method");
    }
    }
    class B extends A
    {
    void show()
    {
    System.out.println("Error");
    }
    }
```

Final classes

- To from class from being inherited it can be defined using a final keyword.

Final class A{………}

(eg)

```
        final class A
       {
       void show()
       {
       System.out.println("Class A");
       }
       }
       class B extends A
       {
       void disp()
```

```
{
System.out.println("Class B");
}
}
```

2.21 Abstract Classes and Interfaces

- In java, classes can be defined that are abstractions of real world objects. These classes are called abstract classes.

The Abstract Classes

- It is necessary to create a superclass that only defines a general form that can be shared by all of its subclasses.
- Abstract keyword is used before the class keyword in the class definition, these classes do not take part in object creation.
- Any class can be specified with the keyword abstract to indicate that it cannot be instantiated.

(eg)

```
Abstract class A
{
 ------
 ------
 }
```

- If object is created for the abstract class the compiler shows an error.

The Abstract Methods

- The methods with only method signature and no method body are called abstract methods.
- Since no implementation a subclass must override them.
- Abstract method defined in a abstract class must be implemented by all it subclasses.
- Only instance methods can be abstract

(eg)

```
 Abstract class A
 {
Abstract void amethod1();
Abstract void amethod2();
 }
Abstract class B extends A
{
void amethod1();
{
System.out.println("Implementing amethod1 in B");
}
void bmethod()
{
System.out.println("Another Completemethod");
}
}
class C extends B
{
void amethod2()
{
System.out.println("Implementing amethod 2 in C");
}
}
class inter 1
{
public static void main(String args[])
{
C obj=new C();
obj.amethod1();
}
}
```

Defining Interface

- An interface is a prototype for a class.
- An interface is like a fully abstract class.
- It is a collection of abstract method declaration and constants, methods in an interface are implicitly abstract.
- Interfaces do not specify any code to implement the methods.
- Any class that implements an interface must implement all of the methods specified in that interface.
- Interface offers multiple inheritances a class can implement many interface but can extend only one class.

Syntax:

```
Interface interface_name
 {
   Variable declarations;
    Method declarations;
 }
```

(eg)

```
interface item
{
static final int code=1001;
static final string name="fan";
void display();
}
```

- The class which implements this interface must define the code for display()

Implementing Interfaces

- While implementing interface the interface method must have public accessibility.
- If a class does not implement all of the interface methods then it must be declared as abstract.

Syntax

```
Class classname implements interface1,interface2…..
{
--------
--------
}
```

- A class can also extend another class and also implement one or more interface.

Syntax

```
Class classname extends superclass implements interface1,interface2…..
{
--------
--------
}
```

(eg)

```
interface Area
{
static final float pi=3.14;
float compute(float x, float y);
}
class Rectangle implements Area
 {
 public float compute(float x, float y);
 {
```

```
return(x*y);
}
}
class Circle implements Area
{
public float compute(float x, float y)
{
return(pi*x*x);
}
}
class inter 2
{
public static void main(String args[])
{
Rectangle rect =new Rectangle();
Circle cir=new Circle();
System.out.println("Area of Rectangle:"+rect.compute(10,20));
System.out.println("Area of Circle:"+cir.compute(10,0));
}
}
```

Variables in Interfaces

- Interface can be used to declare a set of constants that can be used in different classes.
- Such constants can be public, static, final.
- It can be accessed by a class that implements this interface as well as any other class.
- If a constant name is repeated in more than one interface it is fully qualified name.
- <interface name>.<constantname>
- These variables must be initialized with a constant value that cannot be changed by the implementing class.

(eg)

```
interface one
{
public static final int p=6;
int q=77;
void aone();
}
interface two
{
int p=9;
}
class A implements one, two
{
int p=4;
public void aone()
{
System.out.println("P="+this.p);
System.out.println("one.p="+one.p);
```

```
 System.out.println("q="+q)
 }
 }
 public class inter 3
 {
 public static void main(String args[])
 {
 A a=new A();
 a.aone();
System.out.println("two.p="+two.p);
}
}
```

Extending Interface

- Interface can also be extended using extends class.
- While interfaces are allowed to extend to other interfaces sub interfaces cannot define the method declared in the super interfaces.

(eg)

```
interface ItemConstants
{
int code=100;
String name="fan";
}
interface Item extends ItemConstants
{
void display();
}
class MyItem implements Item
{
public void display()
{
System.out.println("Code="+code);
System.out.println("Name="+name);
}
}
    class inter 5
     {
     public static void main(String args[])
     {
     MyItem i=new MyItem();
     i.display();
     }
     }
```

Interface Reference

- Although interface cannot be instantiated variables of an interface type can be declared like abstract classes.
- Any object of a class that implements the interface can be assigned to a variable of that interface type.

(eg)

```
interface X
{
void disp();
}
class A implements X
{
public void disp()
{
System.out.println("in A");
}
}
class B extends A
{
public void disp()
{
System.out.println(“in B”);
}
}
class C implements X
{
public void disp()
{
System.out.println(“in C”);
}
}
class inter 6
{
public static void main(String args[])
{
 X x;
x=new B();
x.disp();
x.new c();
x.disp();
}
}
```

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2015 onwards)

DEPARTMENT OF COMMERCE(CA)

SUBJECT : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 **CLASS : II M.COM CA**

Unit – III

Packages and Interfaces: Packages – Access protection – Importing packages – Interfaces – Exception handling: Fundamentals – Exception types – Try and catch – Multiple catch – Nested try – throw – throws – finally – Build in exception.

UNIT-III

Packages and Access Modifiers

3.1.1 Packages-An Introduction

- When software is developed all related files are stored into a single directory.
- It is easy to handle when software is small, if it is bigger it is difficult.
- In java the sort of problems can be avoided by using the packages.
- Package is a way of organizing files into different directories according to their functionality.
- In java package is a library of types(classes&interfaces)
- Multiple classes of larger programs are grouped together into packages.
- Packages can be conceptualized into the following ways:
    - Type Libraries
    - Hierarchical Organisation
    - Avoiding name collisions
    - Hiding the implementation

1 .Type Libraries:

- Packages are simply a set of libraries.
- Any program will atleast use the runtime libraries of the java API, some of which are java.lang, java.io, java.util, java.net, java.awt, java.applet.

2. Hierarchical Organisation:

- A package is a tool to help to organize the types you create for the program.
- With packages one can recognize a program into logically related groups of types and organize the groups hierarchically.
- Packages can contain other packages also.
- The entities contained in a package-classes, interfaces, sub-packages are called the members of the package.
- Java compilers & JVM use the package hierarchy as away to locate files or a disk.

3. Avoiding name collisions:

- Packages help to avoid class name collisions when we use the same class name as already defined.

  Eg: vector is a class defined by user.

- This name conflict with vector class from JDK, but this never happens
- JDK uses java.util as a package name for the vector class.

4. Hiding the implementations:

- Packages is a tool that can help you isolate interface from implementation.
- One can grant special access privileges to types within the same package.

  (Eg) JDK package.

3.1.2 Package Naming Convention

- Package used by a program came from many sources.
- Name should not conflict with the package name created by others. Of course one doesn't know the name of the package created by others.
- To solve this java comes with a recommended naming convention for packages.
- Java doesn't enforce any naming convention it is up to you to take measure for preventing naming conflict within java program.

3.1.3 The Package Declaration

- Packages are groups of related classes and interfaces.
- Built in packages user-defined packages.
- While writing java program class must be placed into a package.
- User can create own packages containing classes and interfaces.
- A class is placed into a packages by including a package declaration at the top of the source file.

Syntax:

```
        Package package-name;
            (or)
```

Package package-name.subpackage-name.

- It should be the first statement.

3.1.4 Creating Packages

```
Package media
Interface forever
 {
    .....
    .....
 }
Class Movie
{
 .....
}
Class Documentry
{
   ....
}
```

- Creates a packages named media
- Movie, forever, documentary are members
- . class files are generated by the compiler
- Directory is created namely media.

Classpath Variable

- CLASSPATH environment variable, one can indicate to java interpreter where it can find the classes that run
- The actual construct of CLASSPATH depends on the system that you are running.
- When an interpreter gets a class name it searches each directory in the CLASSPATH until it finds the class its looking for.

Location of the Class Files

- To set a CLASSPATH environment variable so that it includes home directory simply type

  C:\> SET CLASSPATH = J:\ASHISH\

  ASHISH it contains com/moviemagic/hits/commercial
- The order in which the directories are specified in the CLASSPATH variable is important.

3.1.5 THE Import STATEMENT

- In java source file two ways are there to refer to a class or interface defined in another packages
  1) Use fully qualified name of the class.
  2) Import that classes fully qualified name into our source file.
- Alternative is to import the class into the source file and then refer to the class by its name.
- Import statement must be placed at the beginning of a file before any class or interface defines

```
        import java.awt.Button;
                (or)
        import java.awt.*;
```

3.1.6 Illustration Package

(eg)

```
package mypack;
public class classA
{
  int addtwo (int a, int b)
 {
  return a+b;
  }
 }
```

```
package mypack;
public class classB
{
 static void greet()
 {
    System.out.println("Hi");
  }
}
```

Source file named as classA.java and classB.java

- Create directory named subpack1 in mypack and another directory subpack2 in subpack1.place classC.java in subpack2

```
package mypack.subpack1.subpack2;
 public class classC
{
      String x="class of the subpackages";
      void display()
     {
       System.out.println(x);
     }
   }
```

Mypack . subpack1 . subpack2

Set the CLASSPATH variable as

```
c:\> SET CLASSPATH=J:\ASHISH\JAVA.
```

```
import mypack .*;
import mypack.subpack1.subpack2.*;
class myclass
```

```
{
  public static void main(String args[])
{
ClassA a=new classA();
classC c=new classC();
classB.greet();
c.display();
int z= a.addtwo (9,45);
System.out.println(z);
}
}
```

3.1.7 The Java Languages Packages

Following packages comprise the standard Java development environment

- Java language packages are also known as java.lang contains classes that are core to the java languages
- The most of classes is object class, a class from which all others inherits
- The other classes include thread, threadDeath, threadGroup which are used to implement multi-threading capabilities and Data type wrappers, string, stringBuffer, system and runtime, Exception, Errors and throwable.
- Java.lang package also defines the Runnable interface to create threads
- Math class is a library of math routine and values such as pi

Java I/O Packages

- Java.io provides a set of i/p and o/p streams used to read and write date to files or other i/p and o/p sources.

Java Utility Package

- Java.util, contains a collection of utility classes. Among them are several generic data structures (Dictionary, stack, vector, hashtable)

- It also contains observer interface and observable class

Java Networking Package

- Java.net contains classes and interface that implement various networking capabilities
- Classes in this package implements URL a connection to URL a socket connection, datagram packet.

Applet Packages

- Java.applet contains the applet classes
- It includes audioclip interface that provides a very high level abstraction of audio

Abstract Window Toolkit packages

- 3 packages are there

  1)java.awt - it provides GUI elements used to get i/p from and display information to the user such as windows, buttons, scrollbars, textitems

  2)java.awt.image - it contains classes and interface for managing image data such as setting the color model, cropping, color filtering setting pixel values, grabbing, snapshots of the screen

  3)java.awt.peer - it contains classes and interfaces that connect platform independent AWT components to their platform-dependent implementations.

3.1.8 Access Protection

- One of the most useful features of java packages is the ability to grant access to classes, interfaces, methods or fields of the same packages

Access Levels for classes

- Classes must be either declared with the keyword public or have no access specifier.

Access Levels for class Members

- 3 access modifiers 1)private 2)public 3)protected
- Private access denoted by the keyword private accessible only to the class which defines it.
- Package Access is denoted by lack of any access modifier keywords. It is accessible to any type in the same package.
- Protected Access denoted by protected. Its field or method is accessible to any type in the same package and to subclasses in any package.
- Public Access denoted by keyword public. Its field or method is accessible to any type in any package.

3.2.1 Exception Handling

An exception is an abnormal condition that arises in a code sequence at run time. A Java exception is an object that describes an exceptional condition that has occurred in a piece of code. When an exceptional condition arises, an object representing that exception is created and thrown in the method that caused the error. An exception can be caught to handle it or pass it on .Exceptions can be generated by the Java run-time system, or they can be manually generated by your code.

3.2.1 Fundamentals of Exception Handling

Java exception handling is managed by via five keywords: try, catch, throw, throws, and finally. Program statements to monitor are contained within a try block if an exception occurs within the try block, it is thrown. Code within catch block catch the exception and handle it. System generated exceptions are automatically thrown by the Java run-time system. To manually throw an exception, use the keyword throw. Any exception that is thrown out of a method must be specified as such by a throws clause. Any code that absolutely must be executed before a method returns is put in a finally block. General form of an exception-handling block

```
try{    // block of code to monitor for errors
}catch (ExceptionType1 exOb){
        // exception handler for ExceptionType1}
catch (ExceptionType2 exOb){
        // exception handler for ExceptionType2
}//...
Finally   // block of code to be executed before try block ends
}
```

3.2.2 Hierarchy of the Exception Class

The runtime has a base set of exceptions deriving from SystemException that it throws when executing individual instructions. The following table hierarchically lists the standard exceptions provided by the runtime and the conditions under which you should create a derived class.

Exception type	Base type	Description	Example
Exception	Object	Base class for all exceptions.	None (use a derived class of this exception).
SystemException	Exception	Base class for all runtime-generated errors.	None (use a derived class of this exception).
IndexOutOfRangeException	SystemException	Thrown by the runtime only when	Indexing an array outside its valid range:

		an array is indexed improperly.	arr[arr.Length+1]
NullReferenceException	SystemException	Thrown by the runtime only when a null object is referenced.	object o = null; o.ToString();
AccessViolationException	SystemException	Thrown by the runtime only when invalid memory is accessed.	Occurs when interoperating with unmanaged code or unsafe managed code, and an invalid pointer is used.
InvalidOperationException	SystemException	Thrown by methods when in an invalid state.	Calling Enumerator.GetNext() after removing an Item from the underlying collection.
ArgumentException	SystemException	Base class for all argument exceptions.	None (use a derived class of this exception).

3.2.3 Types of exception

An exception is a problem that arises during the execution of a program. An exception can occur for many different reasons, including the following:

- A user has entered invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications, or the JVM has run out of memory.

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

To understand how exception handling works in Java, you need to understand the three categories of exceptions:

- **Checked exceptions:** A checked exception is an exception that is typically a user error or a problem that cannot be foreseen by the programmer. For example, if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions cannot simply be ignored at the time of compilation.
- **Runtime exceptions:** A runtime exception is an exception that occurs that probably could have been avoided by the programmer. As opposed to checked exceptions, runtime exceptions are ignored at the time of compilation.
- **Errors:** These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation

3.2.4 Exception Class

The Exception Class is the super class of the exception objects that should be caught by the Java Program. It provide 2 constructors

public Exception()

public Exception(String message)

First is the default one and the second constructor takes a string as argument.

The Exception class does not define any methods of its own. It inherits methods provided by Throwable.

String getMessage()

It returns the String object associated with the invoking exception object.

Void printStackTrace()

It displays the stack trace by sending information to System.err

String toString()

It returns a string containing the name of the invoking object and its associated messages.

Throwable fillInStackTrace()

It fills in the execution stack trace and can be used to obtain additional information about the invoking error or exception.

String getLocalizedMessage()

It is used to provide a localized message different from that passed to the objects constructor.

3.2.5 Uncaught Exceptions

```
class Error1
{
Public static void main(String args[])
{
int a=10;
int b=5;
int c=5;
int x=a/(b-c);
System.out.println("X="+x);
int y=a/(b+c);
System.out.println("Y="+y);
}
}
```

When Java runtime detect the attempt to divide by zero, it constructs a new exception object and then throws this exception. It is caught by the default handler. The errors are printed by Stack Trace. The Stack Trace will always show the sequence of method invocations that led to the error.

3.2.6 Handling Exceptions

An exception is generated when the program does some illegal operation.

```
Public class Example1
{
Public static void main(String args[])
{
int i=1, j=0,k;
k=i/j;
System.out.println("Hello world");
}}
```

Compiling the above program will give the following result.

Java.lang.ArithmeticException:/ by zero

At Example1.main(Example1.java:6)

The above exception can be handled properly using try-catch blocks

The try-catch construct

The general form of Exception handling block is

Try

```
{
// block of code to monitor for errors
}
Catch (Exceptiontype1 exob)
{
//code to handle exception of Exceptiontype1
}
Catch (Exceptiontype2 exob)
{
//code to handle exception of Exceptiontype2
}
```

Exceptions thrown during the execution of the try block can be caught and handled in catch block. If an exception occurs which matches the argument of catch, the statements in the curly braces that follows the catch keyword will be executed.

```
public class Example2
{
public static void main(String args[])
{
int i=1,j=0,k;
try
{
k=i/j;
}
catch (ArithmeticException ae)
{
System.out.println("Division by zero");
}
System.out.println("Hello World");
}
}
```

Output

```
Division by zero
Hello World
```

```
public class Example2
{
public static void main(String args[])
{
int i=1,j=0,k;
try
{
k=i/j;
System.out.println("Hello World");
}
catch (ArithmeticException ae)
{
System.out.println("Division by zero");
}
}
}
```

Output

```
Division by zero
```

```
public class Example3
{
```

```
public static void main(String args[])
{
int i=1,j=0,k;
try
{
k=i/j;
}
catch (ArithmeticException ae)
{
System.out.println("Division by zero");
}
}
}
```

Output

```
Java.lang.ArithmeticException:/ by zero
At Example1.main(Example1.java:6)
```

Nested Try Block

It is a try block nested inside another try block. If inner try does not have a catch handler then the stack is unwound and the next try block catch handlers are inspected for a match.

```
class NestTry
{
public static void main(String args[])
{
try
{
int a=args.length;
int b=42/a;
System.out.println("a="+a);
        try
        {
        if(a==1)
                a=9/(a-a);
        if(a==2)
        {
                int c[]={1};
                c[10]=99;
        }
        }
        catch(ArrayIndexOutOfBoundException e)
        {
        System.out.println("Array index out of bound" +e);
        }
        }
catch(ArithmeticException e)
{
System.out.println("Divide by 0"+e);
}
}
}
```

Finally Block

If a try block has a finally block attached, the code block associated with finally is always executed regardless of how the try block exits. A try block exits because of normal termination by falling through the end brace, return or break statement or an exception that was thrown. Both catch and finally blocks are optional. At least one of the catch or finally blocks must exist with a try.

```
class finallytest
{
public static void main(String args[])
{
int a[]={1,2};
char b[]={'x'};
try
{
System.out.println("in outer try");
try
{
System.out.println("in inner try");
a[5]=10;
}
finally
{
System.out.println("in inner finally");
}
}
catch(ArithmeticException ae)
{
System.out.println("in outer catch");
}
Finally
{
System.out.println("in outer finally");
}
}}
```

Output

```
in outer try
in inner try
in inner finally
in outer finally
```

throw Clause

It is possible for the program to throw an exception explicitly rather than being thrown by the Java runtime system.

Syntax:

```
        Throw expression;
```

Here expression is an object of type Throwable or subclass of Throwable. There are 2 ways to create Throwable object

1. Using a parameter into catch clause
2. creating one with the new operator.

Eg.

```
throw new NullPointerException("demo");
throw new ArtithmeticException();
```

The flow of execution stops immediately after throw statement.

```
class throwtest
{
static void xyz()
{
try
{
```

```
Throw new NullPointerException("Exception thrown explicitly");
}
catch (NullPointerException e)
{
System.out.println("Caught inside xyz");
throw e;
}
}
public static void main(String args[])
{
try
{
xyz();
}
catch(NullPointerException e)
{
System.out.println("Caught inside  main"+ e);
}
}
}
```

Output

```
Caught inside xyz
Caught inside main: java.lang.NullPointerException: Exception thrown
explicitly
```

Throws Clause

It lists the type of exception that a method might throw.

Syntax:

```
return_type method_name(parameter_list) throws exceptionlist
{
body of the method
}
```

Eg

```
class throwtest
{
static void methodone() throws IllegalAcessException
{
System.out.println("inside method one");
throw new IllegalAccessException("Exception raised");
}
public static void main(String args[])
{
try
{
methodone();
}
catch(IllegalAccessException e)
{
System.out.println("Exception handled:"+e);
}
}
}
```

Output

```
inside method one
Exception handled: java.lang.IllegalAccessException:Exception raised
```

3.2.7 User defined Exception

It is done by defining a subclass of exception.

```
class myexception extends Exception
{
myexception(String message)
{
super(message);
}
}
class TestMyException
{
public static void main(String args[])
{
int x=5,y=1000;
try
{
float z=(float)x/(float)y;
if(z<0.01)
{
throw new myException("Number is too small");
}
}
catch(MyException e)
{
System.out.println("Caught my exception");
System.out.println("e.getMessage());
}
finally
{
System.out.println("I am always here");
}}}
```

Output

```
Caught my exception
Number is too small
I am always here
```

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF COMMERCE (CA)

SUBJECT : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 **CLASS : II M.COM CA**

Unit – IV

Multithread programming: Thread model –Life cycle of thread – Creating thread – Multiple threads – Thread priorities – Synchronization – Inter thread Communication – Suspending, Resuming and Stopping threads – I/O Applets, and other topics.

UNIT IV

4.1 Multithreaded Programming

Multithreading allows multiple tasks to execute concurrently within a single program. The advantage of multiple threads in a program is that it utilizes system resources better because other threads can grab CPU time when one line of execution is blocked.

4.1.1 The Java Thread model

A thread goes through various stages in its life cycle. For example, a thread is born, started, runs, and then dies. The stages of the thread are:

New: A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.

Runnable: After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.

Waiting: Sometimes a thread transitions to the waiting state while the thread waits for another thread to perform a task. A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.

Timed waiting: A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state, transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.

Terminated: A runnable thread enters the terminated state when it completes its task or otherwise terminates.

4.1.2 Thread Priorities

Every Java thread has a priority that helps the operating system determine the order in which threads are scheduled. Java priorities are in the range between MIN_PRIORITY (a constant of 1) and MAX_PRIORITY (a constant of 10). By default, every thread is given priority NORM_PRIORITY (a constant of 5). Threads with higher priority are more important to a program and should be allocated processor time before lower-priority threads. However, thread priorities cannot guarantee the order in which threads execute and very much platform dependent.

4.1.3 Creating a Thread

Java defines two ways in which this can be accomplished:

You can implement the Runnable interface.

You can extend the Thread class, itself.

Create Thread by Implementing Runnable:

The easiest way to create a thread is to create a class that implements the Runnable interface.

To implement Runnable, a class need only implement a single method called run( ), which is declared like this:

```
public void run( )
```

You will define the code that constitutes the new thread inside run() method. It is important to understand that run () can call other methods, use other classes, and declare variables, just like the main thread can.

After you create a class that implements Runnable, you will instantiate an object of type Thread from within that class. Thread defines several constructors. The one that we will use is shown here:

```
Thread(Runnable threadOb, String threadName);
```

Here threadOb is an instance of a class that implements the Runnable interface and the name of the new thread is specified by threadName.
After the new thread is created, it will not start running until you call its start( ) method, which is declared within Thread. The start( ) method is shown here:

```
void start( );
```

Example:
Here is an example that creates a new thread and starts it running:

```
class NewThread implements Runnable {
Thread t;
NewThread() {
 t = new Thread(this, "Demo Thread");
 System.out.println("Child thread: " + t);
 t.start(); // Start the thread
 }
.
   public void run() {
   try {
   for(int i = 5; i > 0; i--) {
   System.out.println("Child Thread: " + i);
   Thread.sleep(500);
    }
    }
    catch (InterruptedException e) {
    System.out.println("Child interrupted.");
   }
   System.out.println("Exiting child thread.");
   }
   }
   class ThreadDemo {
   public static void main(String args[]) {
```

```
 new NewThread(); // create a new thread
 try {
 for(int i = 5; i > 0; i--) {
 System.out.println("Main Thread: " + i);
 Thread.sleep(1000);
 }
 } catch (InterruptedException e) {
 System.out.println("Main thread interrupted.");
 }
System.out.println("Main thread exiting.");
}
}
```

4.1.4 Runnable interface

The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. The class must define a method of no arguments called run. This interface is designed to provide a common protocol for objects that wish to execute code while they are active. For example, Runnable is implemented by class Thread. Being active simply means that a thread has been started and has not yet been stopped.

In addition, Runnable provides the means for a class to be active while not subclassing Thread. A class that implements Runnable can run without subclassing Thread by instantiating a Thread instance and passing itself in as the target. In most cases, the Runnable interface should be used if you are only planning to override the run() method and no other Thread methods. This is important because classes should not be subclassed unless the programmer intends on modifying or enhancing the fundamental behavior of the class.

public void **run**()

When an object implementing interface Runnable is used to create a thread, starting the thread causes the object's run method to be called in that separately executing thread.

4.1.5 Thread class

A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently. Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority. Each thread may or may not also be marked as a daemon. When code running in some thread creates a new Thread object, the new thread has its priority initially set equal to the priority of the creating thread, and is a daemon thread if and only if the creating thread is a daemon.

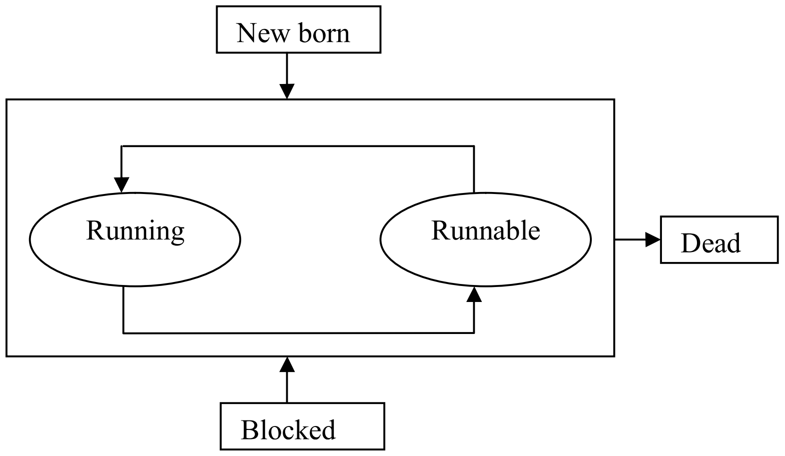
When a Java Virtual Machine starts up, there is usually a single non-daemon thread (which typically calls the method named main of some designated class). The Java Virtual Machine continues to execute threads until either of the following occurs:

- The exit method of class Runtime has been called and the security manager has permitted the exit operation to take place.
- All threads that are not daemon threads have died, either by returning from the call to the run method or by throwing an exception that propagate beyond the run method.

There are two ways to create a new thread of execution. One is to declare a class to be a subclass of Thread. This subclass should override the run method of class Thread. An instance of the subclass can then be allocated and started.

4.1.6 Thread Life Cycle

In the life cycle of the thread, only one thread can be executed and uses in the CPU at a time. All the other thread should be in some other states. The are mainly 5 states in the life cycle of thread.



States of a thread

1. newborn – New Thread that is created
2. runnable – Thread which is ready for execution and waiting for CPU
3. running – CPU gives the time and thread executes using run() method
4. blocked – Thread which is suspended, sleeping or waiting
4. dead – Thread completes execution

Methods

Start() – starts the thread

Stop() – kills the thread

Resume() – suspended thread resumes execution

Wait – used to run again using notify or notify all

4.1.7 Thread Scheduling

Thread scheduling is the process of determining execution order of threads. At a time only one thread can be executed by the CPU. Schedulers follow different algorithm to decide which thread gets CPU. The scheduler is platform independent.

There are two types of scheduling method

a) Pre-emptive Scheduling – priority based

If a thread with highest priority exists then current thread is moved to Runnable state. It is used when the CPU encounters an important thread.

4.1.8 Thread priorities

Each thread is assigned a priority that affects the order of execution of the thread. The scheduler lets the high priority thread to be executed first. A thread can inherit the priority of its parent thread during execution. Priorities are integer values from 1 to 10.

Methods

setPriority – it is used to set priority of the thread
getPriority – it reads the priority of the thread

MIN_PRIORITY – 1
MAX_PRIORITY – 10
NORM_PRIORITY – 5

If no priority is provided default priority is set as 5 that is NORM_PRIORITY.

Example

```
class A extends Thread
{
public void run()
{
System.out.println("Thread A started");
for(int i=0;i<=4;i++)
```

```
System.out.println("From Thread A: i="+i);
System.out.println("Exit from A");
}
}
class B extends Thread
{
public void run()
{
System.out.println("Thread B started");
for(int j=0;j<=4;j++)
System.out.println("From Thread B: j="+j);
System.out.println("Exit from B");
}
}
class C extends Thread
{
public void run()
{
System.out.println("Thread C started");
for(int k=0;k<=4;k++)
System.out.println("From Thread C: k="+k);
System.out.println("Exit from C");
}
}
class ThreadPriority
{
public static void main(String args[])
{
A threadA=new A();
B threadB=new B();
```

```
C threadC=new C();
threadC.setPriority(Thread.MAX_PRIORITY);
threadB.setPriority(Thread.getPriority()+1);
threadA.setPriority(Thread.MIN_PRIORITY);
System.out.println("Start ThreadA");
threadA.start();
System.out.println("Start ThreadB");
threadB.start();
System.out.println("Start ThreadC");
threadC.start();
System.out.println("End of Main");
}
}
```

b) Time-Sliced Scheduling

It is a non-priority based scheduling. Every thread executes for fixed length of time. If the allotted time is over then the current thread moves to Runnable state. The low or high priority threads gets the same time slots. The advantage of this scheduling method is no thread starves for CPU.

4.1.9 Synchronization and Deadlock

Generally, one thread accesses one source of data (or to say, its own source) like your thread accesses your bank account. Sometimes, it may be needed multiple threads to access the same source of data like all the partners of a business may be acting on the joint bank account at the same time. When multiple threads access or share, there may be **data corruption** or **inconsistency**.

To avoid data inconsistency, a concept called **synchronization** comes. The literature meaning of synchronization is **putting the events in an order**. **Synchronized** is a keyword of Java used as **access modifier**. A block of code or a method can be synchronized. The advantage of synchronization is only one thread is allowed to access the source and other threads are made to wait until the first thread completes its task (of accessing the source). This is known as a **thread safe** operation. Synchronization is a way of **inter-thread communication** (another is **join ()**). In the following snippet of code, whole method is synchronized.

4.1.10 Starvation and Deadlock

These are the two problems that may creep into the programming code if the programmer is not aware of these concepts.

1. Your style of programming should allow fairly each thread to get microprocessor time; this is known as **fair system**. A **fair system** does not allow a thread to get **starved** of processor time. A **starved thread** is the one that does not get processor for a long time. A starved thread cannot give the output fast.

After **starvation**, another problem with threads is **deadlock**. When one thread depends on another for their execution, a deadlock occurs. When two threads are holding locks on two different resources, one thread would like to have other's source, a deadlock occurs. Deadlock is the result of poor programming code and is not shown by a compiler or execution environment as an exception. When exists, the program simply hangs and programmer should take care of himself.

4.1.11 Suspending, Resuming, or Stopping threads

Java provides complete control over multithreaded program. You can develop a multithreaded program which can be suspended, resumed, or stopped completely based on your requirements. There are various static methods which you can use on thread objects to control their behavior. Following table lists down those methods –

Sr.No.	Method & Description
	public void suspend() This method puts a thread in the suspended state and can be resumed using resume() method.
2	**public void stop()** This method stops a thread completely.
3	**public void resume()** This method resumes a thread, which was suspended using suspend() method.
4	**public void wait()** Causes the current thread to wait until another thread invokes the notify().
5	**public void notify()** Wakes up a single thread that is waiting on this object's monitor.

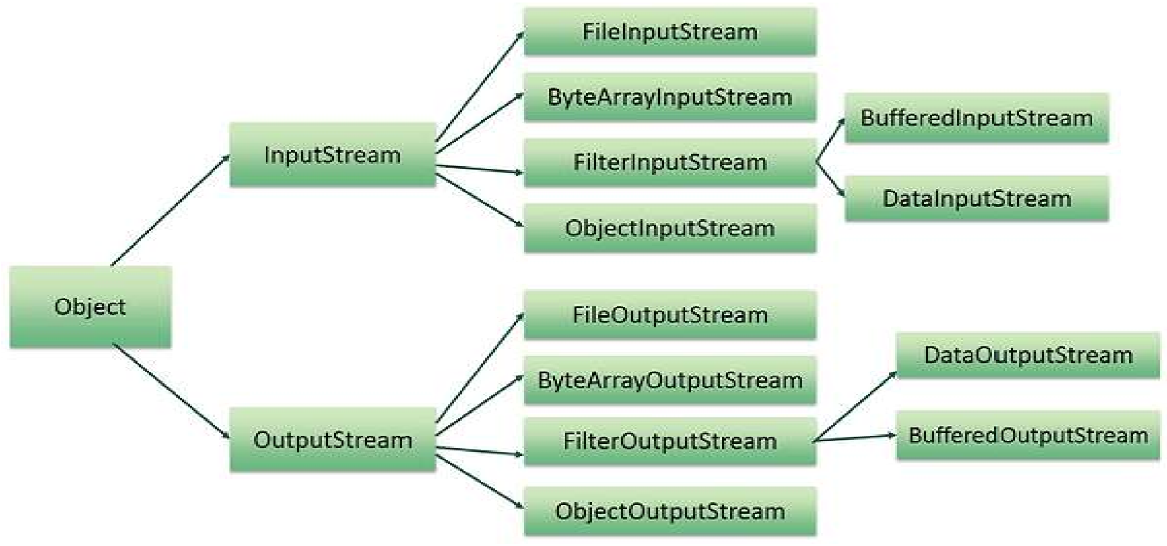
4.2 I/O Applets

Streams

Reading and Writing Files

A stream can be defined as a sequence of data. The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination. Here is a hierarchy of classes to deal with Input and Output

streams.



Input and Output Operations

- Java.io package – used for input and output operations
- Data retrieved from an input source and results sent to an output destination
- All devices handled by streams (a logical entity that produces or consumes information)
- Stream is linked to physical device by Java I/O system.
- Streams – array of bytes or characters, a file, a pipe or a network connection.

- Two kinds of streams
    - Byte stream – 8-bit I/O operation. 2 classes are InputStream and OutputStream.
    - Character stream – 16-bit Unicode character I/O. 2 classes are Reader and Writer.

4.2.1Hierarchy of Classes in java.io package

- InputStream
    - FilterInputStream
        - BufferedInputStream
        - DataInputStream
        - LineNumberInputStream
        - PushbackInputStream
    - ByteArrayInputStream
    - FileInputStream
    - ObjectInputStream
    - PipedInputStream
    - StringBufferInputStream
- OutputStream
    - FilterOutputStream
        - BufferedOutputStream
        - DataOutputStream
        - PrintStream
    - ByteArrayOutputStream
    - FileOutputStream

    - ObjectOutputStream
    - PipedOutputStream
- Reader
    - BufferedReader
        - LineNumberReader
    - CharArrayReader
    - PipedReader
    - StringReader
    - FilterReader
        - PushbackReader
    - InputStreamReader
        - FileReader
- Writer
    - BufferedWriter
    - CharArrayWriter
    - FileWriter
    - PipedWriter
    - PrintWriter
    - StringWriter
    - OutputStreamWriter
        - FileWriter
- File
- RandomAccessFile
- FileDescriptor

{

- StreamTokenizer

4.2.2 InputStream and OutputStream Classes

- **InputStream**

  InputStream is an abstract class that defines Java's model of streaming byte input. All of the methods in this class will throw an IOException on error conditions.

- **OutputStream**

  OutputStream is an abstract class that defines streaming byte output. All of the methods in this class return a void value and throw an IOException in the case of errors.

- Example:

```
import java.io.*;
class filecopy
{
public static void main(String s[])
{
FileInputStream f1;
FileOutputStream f2;
try
{
f1=new FileInputStream(s[0]);
f2=new FileOutputStream(s[1]);
int i=f1.read();
while(i!=-1)
{
```

```
f2.write(i);
i=f1.read();
while(i!=-1)
{
f2.write(i);
i=f1.read();
}
f1.close();
f2.close();
}
catch(FileNotFoundException e)
{
}
catch(IOException e)
{
}
}}
```

4.2.3Reading Console Input

There are few ways to read input string from your console/keyboard. The following smaple code shows how to read a string from the console/keyboard by using Java.

Note: Java Console Example to read password

```
import java.io.*;
class ReadPasswordTest{
 public static void main(String args[]){
 Console c=System.console();
 System.out.println("Enter password: ");
 char[] ch=c.readPassword();
 String pass=String.valueOf(ch);//converting char array into string.
 }
 public class ConsoleReadingDemo {

 public static void main(String[] args) {
BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
System.out.print("Please enter user name : ");
String username = null;
try {
   username = reader.readLine();
} catch (IOException e) {
   e.printStackTrace();
}
System.out.println("You entered : " + username);
Scanner in = new Scanner(System.in);
System.out.print("Please enter user name : ");
username = in.nextLine();
System.out.println("You entered : " + username);
 Console console = System.console();
username = console.readLine("Please enter user name : ");
System.out.println("You entered : " + username);
  }}
```

The last part of code used java.io.Console class. We cannot get Console instance from System.Console() when running the demo code through Eclipse. Because eclipse runs your application as a background process and not as a top-level process with a system console.

4.2.4 Writing Console Output

```
File file = new File("test.txt");
FileOutputStream fis = new FileOutputStream(file);
PrintStream out = new PrintStream(fis);
System.setOut(out);
System.out.println("First Line");
System.out.println("Second Line");
System.out.println("Third Line");
System.out.println("Fourth Line");
```

4.2.5 Applet Fundamentals

- **public void init():** is used to initialized the Applet. It is invoked only once.
- **public void start():** is invoked after the init() method or browser is maximized. It is used to start the Applet.
- **public void stop():** is used to stop the Applet. ...
- **public void destroy():** is used to destroy the Applet.

Transient and Volatile Modifiers

The **volatile** and **transient modifiers** apply to fields of classes. The **transient modifier** tells the **Java** object serialization subsystem to exclude the field when serializing an instance of the class. ... **Volatile** means other threads can edit that particular variable. So the compiler allows access to them.

When serializable interface is declared, the compiler knows that the object has to be handled so as to be able to serialize it. However, if you declare a variable in an object as transient, then it doesn't get serialized.

Volatile

Specifying a variable as volatile tells the JVM that any threads using that variable are not allowed to cache that value at all. Volatile modifier tells the compiler that the variable modified by volatile can be changed unexpectedly by other parts of the program.

What are transient and volatile modifiers?

Volatile is a access modifier that informs to the compiler that the variable with this modifier can be changed unexpectedly by other elements of the program. In multithreading environment, one or more threads can share the same instance variables. Each thread can have its own copy of volatile variable. The real copy or the master copy of the variable is updated at timesThe transient access modifier is used at the time of object serialization in order not to serialize the instance variable value.

Instance of

The **java instanceof operator** is used to test whether the object is an instance of the specified type (class or subclass or interface).

The instanceof in java is also known as type*comparison operator* because it compares the instance with type. It returns either true or false. If we apply the instanceof operator with any variable that has null value, it returns false.

Example

```
class Simple1{
```

```
public static void main(String args[]){
Simple1 s=new Simple1();
System.out.println(s instanceof Simple1);//true
}
}
```

strictfp

strictfp is a keyword in the Java programming language that restricts floating-point calculations to ensure portability. The strictfp command was introduced into Java with the Java virtual machine (JVM) version 1.2 and is available for use on all currently updated Java VMs.

Native Method

The Java native method is a great way to gain and merge the power of C or C++ programming into Java. To use Java as a scientific and high performance language, when efficient native Java compilers are not fully implemented, use native method can boost the performance to at least the speed of C compiled code

Assertion:

Assertion is a statement in java. It can be used to test your assumptions about the program.

While executing assertion, it is believed to be true. If it fails, JVM will throw an error named AssertionError. It is mainly used for testing purpose.

Advantage of Assertion:

It provides an effective way to detect and correct programming errors.

Syntax of using Assertion:

There are two ways to use assertion. First way is:

```
    assert expression;
```

and second way is:

```
    assert expression1 : expression2;
```

Simple Example of Assertion in java:

```
import java.util.Scanner;
 class AssertionExample{
 public static void main( String args[] ){

  Scanner scanner = new Scanner( System.in );
  System.out.print("Enter ur age ");

  int value = scanner.nextInt();
  assert value>=18:" Not valid";

  System.out.println("value is "+value);
}  }
```

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF COMMERCE (CA)

SUBJECT : JAVA
SEMESTER : III
SUBJECT CODE: 16CCP304 CLASS : II M.COM CA

Unit – V

The Applet Class: Basics – Building applet code – Applet life cycle – Creating an executable applet – Designing a web page – Running the applet – Getting input from the user – Graphics programming: The graphic class – Lines and rectangles – Circles and ellipses – Using control loops in applets – Drawing bar charts.

UNIT V

5.1. Definition

An Applet is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal.

There are some important **differences between an Applet and a standalone Java application**, including the following:

- An applet is a Java class that extends the java.applet.Applet class.
- A main() method is not invoked on an applet, and an applet class will not define main().
- Applets are designed to be embedded within an HTML page.
- When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine.
- A JVM is required to view an applet. The JVM can be either a plug-in of the Web browser or a separate runtime environment.
- The JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's lifetime.
- Applets have strict security rules that are enforced by the Web browser. The security of an applet is often referred to as sandbox security, comparing the applet to a child playing in a sandbox with various rules that must be followed.
- Other classes that the applet needs can be downloaded in a single Java Archive (JAR) file.

5.1.1 Local Applets

Local applets are applet types that are developed and stored in local system. The web page will search the local system directories, find the local applet and execute it. Execution of local applet does not require internet connection.

Specifying a Local Applet

<applet codebase=**"path**" **code=**"**NewApplet.class**" **width=120 height=120 >**
</apple>

Remote Applets

Local applets are applet types that are developed and stored in local system. The web page will search the local system directories, find the local applet and execute it. Execution of local applet does not require internet connection.

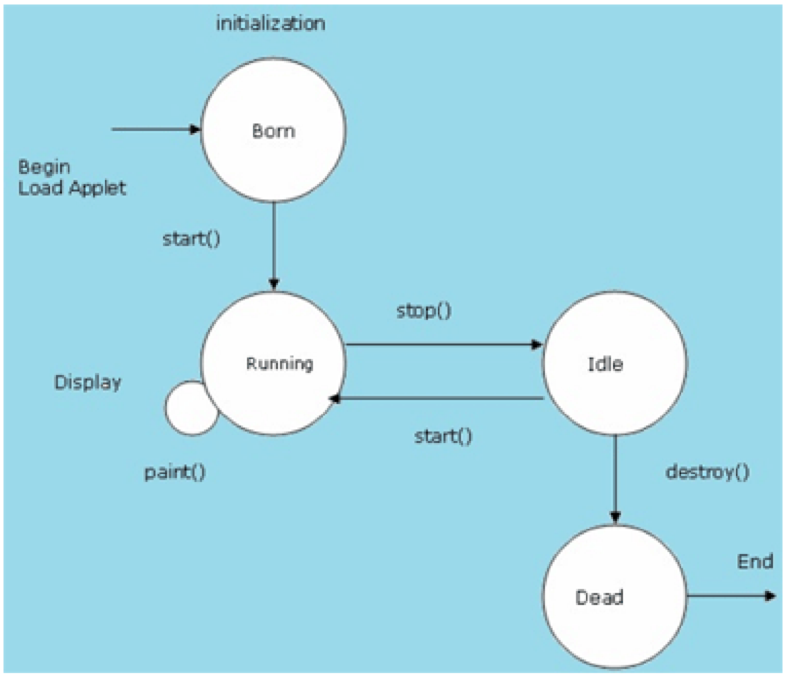
Specifying a Remote Applet
<applet
codebase="http://www.myconnect.com/applets/"
code="NewApplet.class"
width=120
height=120 >
</applet>

Note: The only difference between Local Applet and Remote Applet is the **value of the codebase attribute.** In the first case, codebase specifies a local folder, and in the second case, it specifies the URL at which the applet is located.

5.1.2 Life Cycle of an Applet

Various states, an Applet, undergo between its object creation and object removal (when the job is over) is known as **life cycle**. Each state is represented by a method. There exist 5 states represented by 5 methods. That is, in its life of execution, the applet exists in one of these 5 states.

These methods are known as "**callback methods**" as they are called automatically by the browser whenever required for the smooth execution of the applet. Programmers just write the methods with some code but never call.



Four methods in the Applet class :

- **init:** This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.
- **start:** This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.
- **stop:** This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.

- **destroy:** This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.
- **paint:** Invoked immediately after the start() method, and also any time the applet needs to repaint itself in the browser. The paint() method is actually inherited from the java.awt.

Applet Life Cycle Example

```
/*
        Applet Life Cycle Example
        This java example explains the life cycle of Java applet.
*/
import java.applet.Applet;
import java.awt.Graphics;

/*
 * Applet can either run by browser or appletviewer application.
 * Define <applet> tag within comments as given below to speed up
 * the testing.
 */

/*
<applet code="AppletLifeCycleExample" width=100 height=100>
</applet>
*/

public class AppletLifeCycleExample extends Applet{
```

```
/*
 * init method is called first.
 * It is used to initialize variables and called only once.
 */
public void init() {
        super.init();
}

/*
 * start method is the second method to be called. start method is
 * called every time the applet has been stopped.
 */
public void start() {
        super.start();
}

/*
 * stop method is called when the the user navigates away from
 * html page containing the applet.
 */
public void stop() {
        super.stop();
}

/* paint method is called every time applet has to redraw its
 * output.
 */
public void paint(Graphics g) {
        super.paint(g);
}
```

```
        /*
         * destroy method is called when browser completely removes
         * the applet from memeory. It should free any resources initialized
         * during the init method.
         */
        public void destroy() {
                super.destroy();
        }

}
```

5.1.3 Building Applet Code

To building the applet code **two classes of java library are essential namely Applet and Graphics.**

- The Applet class is contained in java.applet package provides life and beehaviour to the applet through its methods such as int(), start() and paint().
- Unlike with applications, where java calls the main() method directly to initiate the execution of the program, when an applet is loaded java automatically calls a series of Applet class methods for starting running and stopping the applet code.
- The Applet class therefore maintains the life cycle of an applet.
- To display the result of the applet code, the paint() method of the Applet class is called up.
- The output may be test, graphics, or sound.
- The **syntax of paint() method** which requires a Graphic object as an argument, is defined as follows

  public void paint(Graphics g)

- This requires that the applet code imports the java.awt package that contain the Graphic class.
- All output operations of an applet are performed using the methods defined in the graphics class.

The general format of applet code is as following:

```
import java.awt.*;
import java.applet.*;
........................
........................
public class applet classname extends Applet
{
................................
............................... //statements
...............................
public void  paint(Graphics g)
{
..........................
........................//Applet operations code
.........................
}
......................
.....................
}
```

Building Applet Code - Example

```
//HelloApplet.java
import java.applet.Applet;
import java.awt.*;
```

```
public class HelloApplet extends Applet
{
  public void paint(Graphics g)
  {
    g.drawString ("Welcome to Applet Tutorial !",100, 100);
  }
}
```

Embedding Applet in HTML (Web Page) - Example

```
<HTML>
<HEAD>
<TITLE>
Hello World Applet
</TITLE>
</HEAD>
<body>
<h1>Hi, This is My First Java Applet on the Web!</h1>
<APPLET CODE="HelloApplet.class" width=500 height=400>
</APPLET>
</body>
</HTML>
```

5.1.4 Creating an Executable Applet

Executable applet is nothing but the .class file of applet, which is obtained by compiling the source code of the applet. Compiling the applet is exactly the smae as compiling an application using following command.

```
javac appletname.java
```

The compiled output file called appletname.class should be placed in the same directory as the source file.

5.1.5 Designing a Web Page Applet

- ➢ Java applet are programs that reside on web page. A web page is basically made up of text and HTML tags that can be interpreted by a web browser or an applet viewer.
- ➢ Java source code, it can be prepared using any ASCII text editor.
- ➢ A web page is also called HTML page or HTML document.
- ➢ A Web pages are stored using a file extension .html such as my Applet.html. Such files are referred to as HTML files.
- ➢ HTML files should be stored in the same directory as the compiled code of the applets.
- ➢ A web page is marked by an opening HTML tag <HTML> and closing HTML tag </HTML> and is **divided into the following three major parts:**
  - ✓ ***Comment Section***
  - ✓ ***Head Section***
  - ✓ ***Body Section***

Comment Section

This is very first section of any web page containing the comments about the web page functionality. It is important to include comments that tell us what is going on in the web page. A comment line begins with <! And ends with a > and the web browsers will ignore the text enclosed between them. The comments are optional and can be included anywhere in the web page.

Head Section

This section contains title, heading and sub heading of the web page. The head section is defined with a starting <Head> tag and closing </Head> tag.

<Head>

<Title>Hello World Applet</Title>

</Head>

Body Section

After the Head Section the body section comes. This is called as body section because the entire information and behavior of the web page is contained in it, It defines what the data placed and where on to the screen. It describes the color, location, sound etc. of the data or information that is going to be placed in the web page.

<body>

<h1>Hi, This is My First Java Applet on the Web!</h1>

</body>

Applet Tag

- The <Applet...> tag supplies the name of the applet to be loaded and tells the browser how much space the applet requires.
- The ellipsis in the tag <Applet...> indicates that it contains certain attributes that must specified.
- The <Applet> tag given below specifies the minimum requirements to place the Hellojava applet on a web page.

```
<Applet
code = "Hellojava.class"
width = 400
Height = 200 >
</Applet>
```

- This HTML code tells the browser to load the compiled java applet Hellojava.class, which is in the same directory as this HTML file.

- It also specifies the display area for the applet output as 400 pixels width and 200 pixels height.
- We can make this display area appear in the center of screen by using the CENTER tags as showsn below:

```
<CENTER>
<Applet>
------------
------------
------------
</Applet>
</CENTER>
```

- **The applet tag discussed above specified the three things:**

1. Name of the applet
2. Width of the applet (in pixels)
3. Height of the applet (in pixels)

Applet Adding HTML to html file

To execute an applet in a web browser, you need to write a short HTML text file that contains the appropriate APPLET tag. Here is the HTML file that executes SimpleApplet:

```
<Applet code = "Hellojava.class" width = 400 Height = 200 >
</Applet>
```

The width and height attributes specify the dimensions of the display area used by the applet. After you create this file, you can execute the HTML file called RunApp.html (say) on the command line.

c:\ appletviewer RunApp.html

Running the Applet

- To execute an applet with an applet viewer, you may also execute the HTML file in which it is enclosed, eg.

  c:\ appletviewer RunApp.html
- Execute the applet the applet viewer, specifying the name of your applet's source file. The applet viewer will encounter the applet tag within the comment and execute your applet.

Example: To Display Numerical Values in Java Applet (Getting input from the user)

In applets, we can display numerical values by first converting them into strings and then using the drawstring() method of Graphics class we can do this easily by calling the valueOf() Method of String class.

Example

```
import java.awt.*;
import java.applet.*;
public class DisplayNumericalValues extends Applet
{
        public void paint(Graphics g)
        {
                int val1 = 10;
                int val2 = 20;
                int sum = val1 + val2;
        String str_sum = "Sum="+String.valueOf(sum);
        g.drawString(str_sum,100,200);
        }
}
```

This applet runs using the following HTML file:

```
<HTML>
<HEAD>
        <TITLE>Display Numerical Values</TITLE>
</HEAD>
<BODY>
        <APPLET Code="DisplayNumericalValues.class" Width=400 Height=300>
        &/APPLET>
<BODY>
</HTML>
```

Output: Sum = 30

GRAPHICS PROGRAMMING

Graphics in any language gives a wonderful look and feel to the users as well as programmers. Programmers draw figures, strings etc with the help of graphics. Without graphics the windows programming is incomplete. Java is not behind. Java provides Abstract Window Toolkit.

5.2. AWT

The Abstract Window Toolkit (AWT) is Java's original platform-independent windowing, graphics, and user-interface widget toolkit. The AWT classes are contained in the java.awt package. It is one of Java's largest packages. Because it is logically organized in a top-down, hierarchical fashion, it is easier to understand and use.

- **Graphics Class**
- **Drawing Strings**
- **Drawing Lines**
- **Drawing Rectangle**

- **Drawing Ellipses and Circles and Ovals**
- **Drawing Arcs**
- **Drawing Polygons**

<u>Note:</u> The java.awt.Graphics is an abstract class, as the actual act of drawing is system-dependent and device-dependent. Each operating platform will provide a subclass of Graphics to perform the actual drawing under the platform, but conform to the specification defined in Graphics.

5.2.1Graphics Class

- Browser or appletviewer sends a Graphics object to the paint method.
- The Graphics object represents the applet window, current font, and current color and provides methods to draw shapes (rectangles, triangles, circles, etc.) and text on the window.

<u>The Graphics Class Coordinate System</u>

- Never use 0, 0 because the applet itself uses it for the title bar.

<u>Graphics class methods</u>

- No return type so they are all void
- ***<u>Draw ()?</u>*** methods draw an outlined share
- ***<u>Fill ()?</u>*** methods draw a solid shape

Displaying text use

- g.drawString ("string",   x, y);
- Need to draw the strings.

Draw a line

- g.drawLine(xStart, yStart, xEnd, yEnd);

Draw a Square

- g.drawRect(int x,int y, int width, int height);

- Make multiple lines that attach at the points.
- For square, width and height value should be same.

Drawing a Rectangle

- g.drawRect(int x,int y, int width, int height);
- Use pixels for height and width and an anchoring corner.

Drawing an oval

- g.drawOval (int x, int y , int width, int height)
- drawn inside an invisible rectangle. Give rectangle coordinates.
- Graphics methods, Use offsets to make your figure easier to move resize.

Using the Color class

- Found in the Java.awt
- All drawing is done in the current color
- Default color is black
- On the Graphics object method called setColor. This takes a Color object.

Example: setColor(Color.RED);

- Many different constants in the color class defining many preset colors
- Colors consist of red, green and blue components (RGB)
- You may create custom colors in the Color class by passing in three integers for red, green and blue.

Example: Color green = new Color(0,255,0);

- Static color Constants defined in the Color class

Color.BLACK

Color.WHITE

Color.GREEN

Color.BLUE

Color.CYAN

Color.LIGHT_GRAY

Color.DARK_GRAY

Color.GRAY

Color.ORANGE

Color.RED

Color.PINK

Color.YELLOW

Color.MAGENTA

Using the Font class

- Also found in Java.awt
- Works just like color, in that color, in that the drawing of strings is done in the current font.

Example:

```
 Font myFont = new Font(?TimesRoman? Font.ITALIC, 28);
 g.Font.setFont(myFont);
 Font(string fontName, int fontStyle, int fontSize)
```

Example

```
import java.awt.*;
class FontsExample extends JApplet
{
public void paint(Graphics g)
{
g.setFont(new Font("TimesRoman", Font.ITALIC, 28));
g.drawString("Some Examples of Fonts", 20, 45);
g.setFont(new Font("Helvetica", Font.PLAIN, 12));
g.drawString("This is an example of plain 12pt Helvetica font", 20, 70);
g.setFont(new Font("TimesRoman", Font.PLAIN, 12));
g.drawString("This is an example of plain 12pt TimesRoman font", 20, 90);
```

```
      g.drawOval(10,10,50,50);
   g.fillOval(10,10,50,50);//filling colors in Oval0
  }
}
```

Drawing Strings

These methods let you draw text strings on the screen. The coordinates refer to the left end of the text's baseline.

public abstract void drawString (String text, int x, int y) :- This method draws text which is specified as the method's argument on the screen in the current font and color, starting at position (x, y).

Drawing Lines

public abstract void drawLine (int x1, int y1, int x2, int y2):- The drawLine() method draws a line on the graphics context in the current color that begins at startX,startY and ends at endX,endY. If (x1, y1) and (x2, y2) are the same point, it will draw a point. There is no method specific to drawing a point.

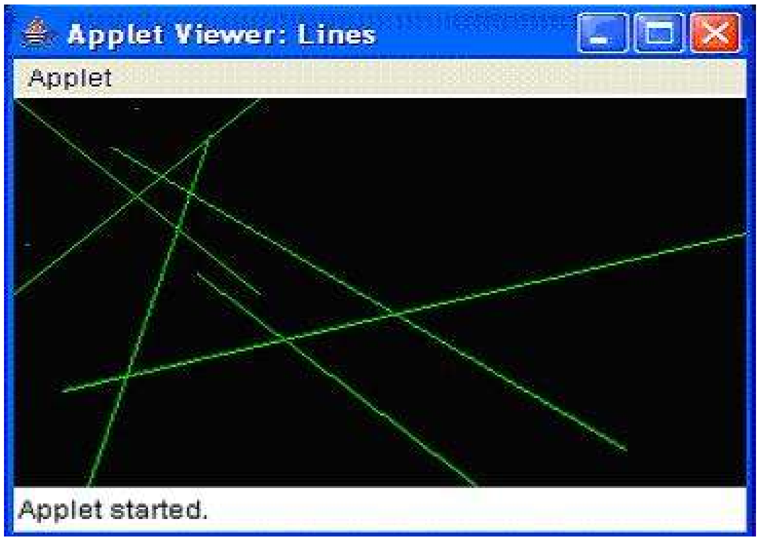
Example:

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Rectangles" width=300 height=200>
</applet>
*/

public class Lines extends Applet
{
        public void init()
        {
                setBackground (Color.black);
                setForeground(Color.green);
        }

        public void paint(Graphics g)
        {
                g.drawLine(0, 0, 100, 100);
                g.drawLine(0, 100, 100, 0);
                g.drawLine(40, 25, 250, 180);
                g.drawLine(75, 90, 400, 400);
                g.drawLine(20, 150, 400, 40); //line
                g.drawLine(5, 290, 80, 19); //line
                g.drawLine (5, 75, 5, 75); // point
                g.drawLine (50, 5, 50, 5); // point
        }
}
```

Output:



Drawing Rectangle

- **public void drawRect (int x, int y, int width, int height):-** The drawRect() method draws a rectangle on the drawing area in the current color from (x, y) to (x+width, y+height). If width or height is negative, nothing is drawn.

- **public abstract void fillRect (int x, int y, int width, int height):-** The fillRect() method draws a filled rectangle on the drawing area in the current color from (x, y) to (x+width-1, y+height-1). The filled rectangle is one pixel smaller to the right and bottom than requested. If width or height is negative, nothing is drawn.

- **public abstract void drawRoundRect (int x, int y, int width, int height, int arcWidth, int arcHeight):-** The drawRoundRect() method draws a rectangle on the drawing area in the current color from (x, y) to (x+width, y+height). Instead of perpendicular corners, the corners are rounded with a horizontal diameter of arcWidth and a vertical diameter of arcHeight. If width or height is a negative number, nothing is drawn. If width, height, arcWidth, and arcHeight are all equal, you get a circle.

- **public abstract void fillRoundRect (int x, int y, int width, int height, int arcWidth, int arcHeight):-** The fillRoundRect() method is similar to drawRoundRect() method except that it draws a filled rectangle on the drawing area in the current color from (x, y) to (x+width-1, y+height-1). If width, height, arcWidth, and arcHeight are all equal, you get a filled circle

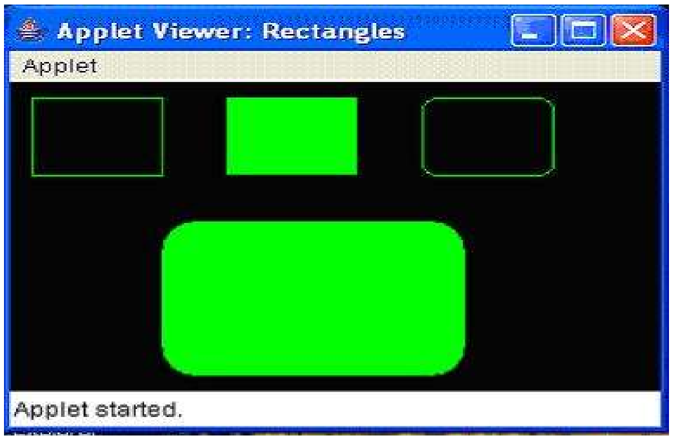
Example:

```
import java.awt.*;
import java.applet.*;
```

```
/*
<applet code="Rectangles" width=300 height=200>
</applet>
*/
public class Rectangles extends Applet
{
        public void init()
        {
                setBackground(Color.black);
                setForeground(Color.green);
        }

        public void paint(Graphics g)
        {
                g.drawRect(10, 10, 60, 50);
                g.fillRect(100, 10, 60, 50);
                g.drawRoundRect(190, 10, 60, 50, 15, 15);
                g.fillRoundRect(70, 90, 140, 100, 30, 40);
        }
}
```

Output:



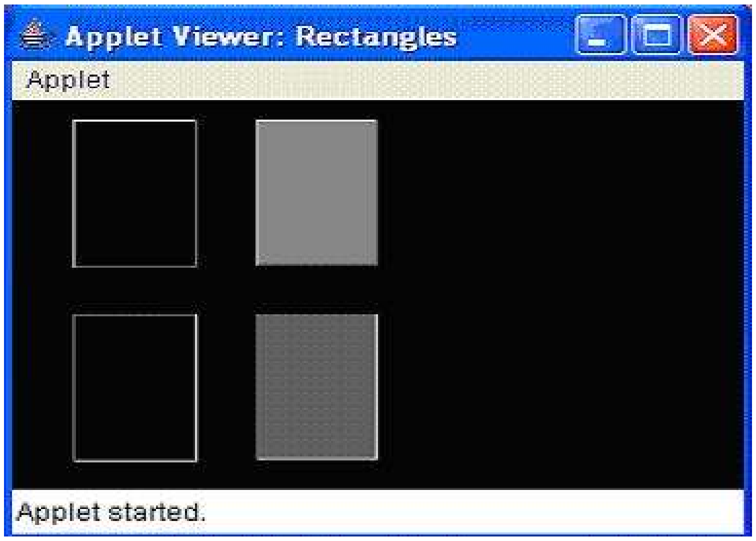
```
Applet started.
```

- **public void fill3DRect (int x, int y, int width, int height, boolean raised):-** It is similar to the draw3DRect ( ) method except that it draws a filled rectangle in the current color from (x, y) to (x+width,y+height).. If width or height is negative, the shadow appears from another direction, and the rectangle isn't filled.

Example

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Rectangles3D" width=300 height=200>
</applet>
*/
public class Rectangles3D extends Applet
{
  public void init()
  {
            setBackground(Color.black);
  }
  public void paint(Graphics g)
  {
            g.setColor (Color.gray);
            g.draw3DRect (25, 10, 50, 75, true);
            g.draw3DRect (25, 110, 50, 75, false);
            g.fill3DRect (100, 10, 50, 75, true);
            g.fill3DRect (100, 110, 50, 75, false);
  }
}
```

Output:



Drawing Ellipses and Circles and Ovals

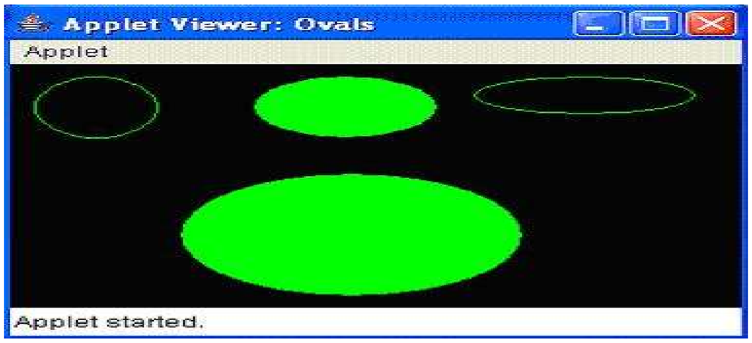
- **public abstract void drawOval (int x, int y, int width, int height):-** The drawOval() method draws an oval in the current color within an invisible bounding rectangle from (x, y) to (x+width, y+height). You cannot specify the oval's center point and radii. If width and height are equal, you get a circle. If width or height is negative, nothing is drawn.
- **public abstract void fillOval (int x, int y, int width, int height):-** The fillOval() method draws a filled oval in the current color within an invisible bounding rectangle from (x, y) to (x+width-1, y+height-1). You cannot specify the oval's center point and radii. Notice that the filled oval is one pixel smaller to the right and bottom than requested. If width or height is negative, nothing is drawn.

Example:

```
import java.awt.*;
import java.applet.*;
/*
```

```
<applet code="Ovals" width=300 height=200>
</applet>
*/
public class Ovals extends Applet
{
    public void init()
    {
        setBackground(Color.black);
        setForeground(Color.green);
    }
    public void paint(Graphics g)
    {
        g.drawOval(10, 10, 50, 50);
        g.fillOval(100, 10, 75, 50);
        g.drawOval(190, 10, 90, 30);
        g.fillOval(70, 90, 140, 100);
    }
    }
```

Output:



Drawing Arcs

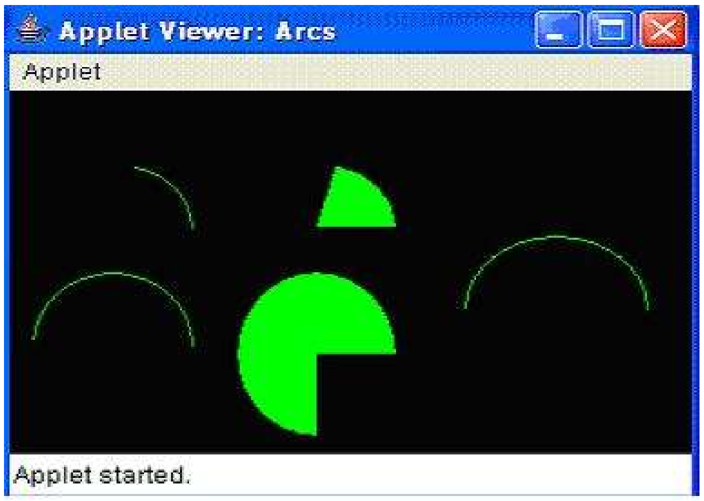
- **The drawArc() method draws an arc in the current color within an invisible bounding rectangle from (x,y) to (x+width, y+height).** The arc starts at startAngle degrees and goes to startAngle + arcAngle degrees. An angle of 0 degrees is at the 3 o'clock position; angles increase counter-clockwise. If arcAngle is negative, drawing is in a clockwise direction. If width and height are equal and arcAngle is 360 degrees, drawArc() draws a circle. If width or height is negative, nothing is drawn.

- **public abstract void fillArc (int x, int y, int width, int height, int startAngle, int arcAngle):-** The fillArc() method is similar to the drawArc() method except that it draws a filled arc in the current color within an invisible bounding rectangle from (x, y) to (x+width-1, y+height-1. If width and height are equal and arcAngle is 360 degrees, fillArc() draws a filled circle. import java.awt.*;

Example

```
import java.applet.*;
/*
<applet code="Arcs" width=300 height=200>
</applet>
*/
public class Arcs extends Applet
{
        public void init()
        {
                setBackground(Color.black);
                setForeground(Color.green);
        }
        public void paint(Graphics g)
        {
                g.drawArc(10, 40, 70, 70, 0, 75);
                g.fillArc(100, 40, 70, 70, 0, 75) ;
                g.drawArc(10, 100, 70, 80, 0, 175);
                g.fillArc(100, 100, 70, 90, 0, 270);
                g.drawArc(200, 80, 80, 80, 0, 180);
        }
}
```

Output:



Drawing Polygons

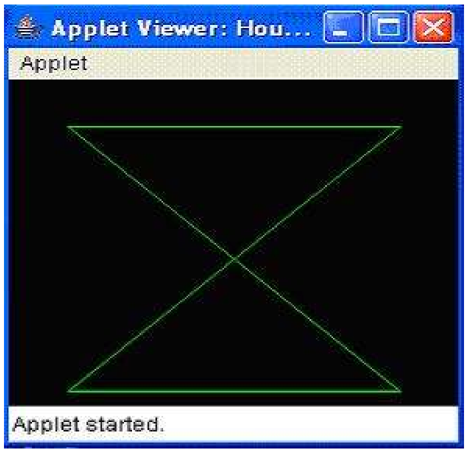
- **public abstract void drawPolygon (int xPoints[], int yPoints[], int numPoints):-** The drawPolygon() method draws a path of numPoints nodes by taking one element at a time out of xPoints and yPoints to make each point. The path is drawn in the current color. If either xPoints or yPoints does not have numPoints elements, drawPolygon() throws a run-time exception

Example:

```
import java.awt.*;
import java.applet.*;
/*
<applet code="HourGlass" width=230 height=210>
</applet>
*/
public class HourGlass extends Applet
{
        public void init()
        {
                setBackground(Color.black);
                setForeground(Color.green);
        }

        public void paint(Graphics g)
        {
                int xpoints[] = {30, 200, 30, 200, 30};
                int ypoints[] = {30, 30, 200, 200, 30};
                int num = 5;
                g.drawPolygon(xpoints, ypoints, num);
        }
}
```

Output:



An entire Java graphics program:

```
import java.awt.*;

class GraphicsProgram extends Canvas{

    public GraphicsProgram(){
        setSize(200, 200);
        setBackground(Color.white);
    }

    public static void main(String[] argS){

         //GraphicsProgram class is now a type of canvas
          //since it extends the Canvas class
          //lets instantiate it
        GraphicsProgram GP = new GraphicsProgram();

        //create a new frame to which we will add a canvas
        Frame aFrame = new Frame();
        aFrame.setSize(300, 300);

        //add the canvas
        aFrame.add(GP);

        aFrame.setVisible(true);
    }

    public void paint(Graphics g){

        g.setColor(Color.blue);
        g.drawLine(30, 30, 80, 80);
        g.drawRect(20, 150, 100, 100);
        g.fillRect(20, 150, 100, 100);
        g.fillOval(150, 20, 100, 100);
        Image img1 = Toolkit.getDefaultToolkit().getImage("sky.jpg");
        g.drawImage(img1, 140, 140, this);
    }
}
```

Output: