JAVA PROGRAMMING

LAB MANUAL

Year	**: 2021**
Subject Code	**: BCSPC3130**
Class	**: 3rd sem**
Branch	**: CSE/IT/MECH/EE**

GANDHI INSTITUTE OF ENGINNERING AND TECHNOLOGY UNIVERSITY), GUNUPUR
RAYAGADA-565022
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OBJECTIVES:

- To teach the students basics of JAVA programs and its execution.
- To teach the students the differences between C++ and Java programming.
- To make the students learn concepts like packages and interfaces.
- To make the students understand life cycle of the applets and its functionality.
- To make the students understand the usage utile package.
- To teach the student, to develop java programs using interfaces.

RECOMMENDED SYSTEM/SOFTWARE REQUIREMENTS:

- Intel based desktop PC with minimum of 2.6GHZ or faster processor with at least 256 MB RAM and 40GB free disk space.
- Operating system: Flavor of any WINDOWS.
- Software : j2sdk1.7.
- Linux and MySQL.
- Eclipse or Net beam.

INTRODUCTION TO OOP

- Object-oriented programming (OOP) is a computer science term used to characterize a Programming language that began development in the 1960's. The term 'object-oriented Programming' was originally coined by Xerox PARC to designate a computer application that describes the methodology of using objects as the foundation for computation. By the 1980's, OOP rose to prominence as the programming language of choice, exemplified by the success of C++. Currently, OOPs such as Java, J2EE, C++,C=, Visual Basic.NET, Python and java Script are popular OOP programming languages that any career-oriented Software Engineer or developer should be familiar with.
- OOP is widely accepted as being far more flexible than other computer programming languages. OOPs use three basic concepts as the fundamentals for the Abstraction, Polymorphism, Event Handling and Encapsulation are also significant concepts within object-oriented programming languages that are explained in online tutorial describing the functionality of each concept in detail.
- The java platform is undoubtedly fast moving and comprehensive. Its many application programming interfaces (APIs) provide a wealth of functionality for all aspects of application and system-level programming. Real-world developers never use one or two APIs to solve a problem, but bring together key functionality spanning a number of APIs, Knowing which APIs you need, which parts of which APIs you need, and how the APIs work together to create the best solution can be a daunting Program.

1. GUIDELINES TO STUDENTS

- ❖ Equipment in the lab for the use of student community. Students need to maintain a proper decorum in the computer lab. Students must use the equipment with care. Any damage is caused is punishable.
- ❖ Students are instructed to come to lab in formal dresses only.
- ❖ Students are supposed to occupy the systems allotted to them and are not supposed to talk or make noise in the lab.
- ❖ Students are required to carry their observation book and lab records with completed exercises while entering the lab.
- ❖ Lab records need to be submitted every week.
- ❖ Students are not supposed to use pen drives in the lab.

2. LAB OBJECTIVE

- ❖ To introduce Java compiler and eclipse platform.
- ❖ To make the student learn an object oriented way of solving problems using java.
- ❖ To make the students to write programs using multithreading concepts and handle exceptions.
- ❖ To make the students to write programs that connects to a database and be able to perform various operations.
- ❖ To make the students to create the Graphical User Interface using Applets, AWT Components & Swing Components.

3. LAB OUTCOME

- ❖ Able to use Java compiler and eclipse platform to write and execute java program.
- ❖ Understand and Apply Object oriented features and Java concepts.
- ❖ Able to apply the concept of multithreading and implement exception handling.
- ❖ Able to access data from a Database with java program.
- ❖ Develop applications using Console I/O and File I/O,GUI applications

4. INTRODUCTION ABOUT LAB

There are 70 systems (DELL) installed in this Lab. Their configurations are as follows:

Processor	: Pentium(R) Dual-Core CPU E5700 @ 3.00GHz
RAM	: 1 GB
Hard Disk	: 320 GB
Mouse	: Optical Mouse

5. LIST OF ASSIGNMENTS AS PER THE UNIVERSITY CURRICULUM

Sl. No	LAB ASSIGNMENTS	Page No.
1	**ASSIGNMENT-1** ➢ Program1: Process for Installation of java software, set the path settings, Programs compilations and executions. ➢ Program2: Write a program to display (“Hello”, “Welcome to Java World”, “A very good morning”) and addition of two numbers by using function. ➢ Program3: Write a program to Convert seconds to hour, minute and seconds ➢ Program4: Write a program to find largest of 3 numbers. ➢ Program5: Write a program to print Fibonacci Series using loop.	
2	**ASSIGNMENT-2** ➢ Program1: Write a Java program that prints the following pattern 1 2 3 4 5 1 2 3 4 1 2 3 1 2 1 ➢ Program2: Write a Java program that calculate mathematical constant ‘e’ using the formula e=1+1/2!+1/3!+........ up to 5 . ➢ Program3: Write a Java program to sort the elements using bubble sort. ➢ Program4: Write a Java program to search an element using binary search. ➢ Program5: Write a Java program to create and display unique three-digit number using 1, 2, 3, 4. Also count how many three-digit numbers are there.	
3	**ASSIGNMENT-3** ➢ Program1: Write a Java program to add two matrices. ➢ Program2: Write a Java program by using a Command Line Argument (CMD). ➢ Program3: Write a Java program to define Scanner class which reads the int, string and double value as an input. ➢ Program4: Write a Java program to convert a binary number to decimal number.	

	➢ Program5: Write a Java program that prints all real solutions to the quadratic equation ax2 + bx +c=0. Read in a, b, c and use the quadratic formula. If the discriminate b2-4ac is negative, display a message stating that there are no real solutions (Using Buffered Reader concept).	
4	**ASSIGNMENT-4**	
	➢ Program1: Write a Java program using classes and object. ➢ Program2: Write a Java program to implement Inheritance ➢ Program3: Write a Java program to implement Polymorphism.(Note: Consider a scenario, Bank is a class that provides method to get the rate of interest. But, rate of interest may differ according to banks. For example, SBI, ICICI and AXIS banks are providing 8.4%, 7.3% and 9.7% rate of interest. ➢ Program4: Write a Java program to implement Method Overriding ➢ Program5: Write a Java program to implement Method Overloading	
5	**ASSIGNMENT-5**	
	➢ Program1: Write a Java program to demonstrate multiple inheritance through default method.(Using Interface) ➢ Program2: Define Abstract class. In this program, Shape is the abstract class; its implementation is provided by the Rectangle and Circle classes. Mostly, we don't know about the implementation class (i.e. hidden to the end user) and object of the implementation class is returns the instance of the class (i, e. Factory Method). ➢ **Wrapper Classes in Java** Program3: Java program to demonstrate Autoboxing Program4: Java program to demonstrate Unboxing Program5: Java program to demonstrate Wrapping and UnWrapping	
6	**ASSIGNMENT-6**	

	➢ **Packages in Java**
	Program1: Introduction of Packages.
	Program2:
	• Simple example of java package
	• How to compile java package
	• How to run java package program
	• How to send the class file to another directory or drive?
	• How to access package from another package?
	
	Program3: Write a Java program to demonstrate Accessing of members when a corresponding class is imported and not imported.
	Program4: Java program to Illustration of user-defined packages:
	➢ **Exception Handling**
	Program5: Write a Java program to demonstrate exception is thrown
	// how the runtime system searches the call stack
	// to find appropriate exception handler.	

7	**ASSIGNMENT-7**	
	➢ **Thread in Java** Program1: Main thread in Java. Program2**:** Java Concurrency – yield(), sleep() and join() methods • Write a Java program to illustrate yield() method • Write a Java program to illustrate sleep() method • Write a Java program to illustrate join() method Program3: Write a Java program to implement Multithreading Program4: Write a Java program to implement Multithreading with synchronized.	
8	**ASSIGNMENT-8**	
	➢ **IO Streams (java.io package) in Java** Program1: Write a program to makes use of two classes (FileInputStream and FileOutputStream) to copy an input file into an output file.{ Byte Stream } Program2**:** Write a program to makes the use of two classes (FileReader and FileWriter) to copy an input file (having Unicode characters) into an output file. { Character Streams } Program3: Write a java program to shows how to read and write Files Using a RandomAccessFile Object. Program4: Write a Java program to implement Serializing an Object Program5: Java program to show why collection framework was needed.	
9	**ASSIGNMENT-9**	
	➢ **Util Package interfaces (List, Set, and Map)** Program1: Write a program to declaration for java.util.ArrayList class. Program2**:** Write a program to declaration for java.util.BitSet class. Program3: Write a program to declaration for java.util.HashMap class. ➢ **Applet in java** Program4: Write a Java program to showing Simple example of Applet by html file. Program5: Write a Java program to implement EventHandling in Applet.	
10	**ASSIGNMENT-10**	
	Program1: Write a Java program to display digital clock by using Applet. Program2**:** Write a Java program of event handling by implementing Action Listener. Program3: Write a Java program to implement window Adapter. Program4: Write a Java program to implement Mouse Motion Adapter. Program5: Write a Java program to implement Key Adapter.	

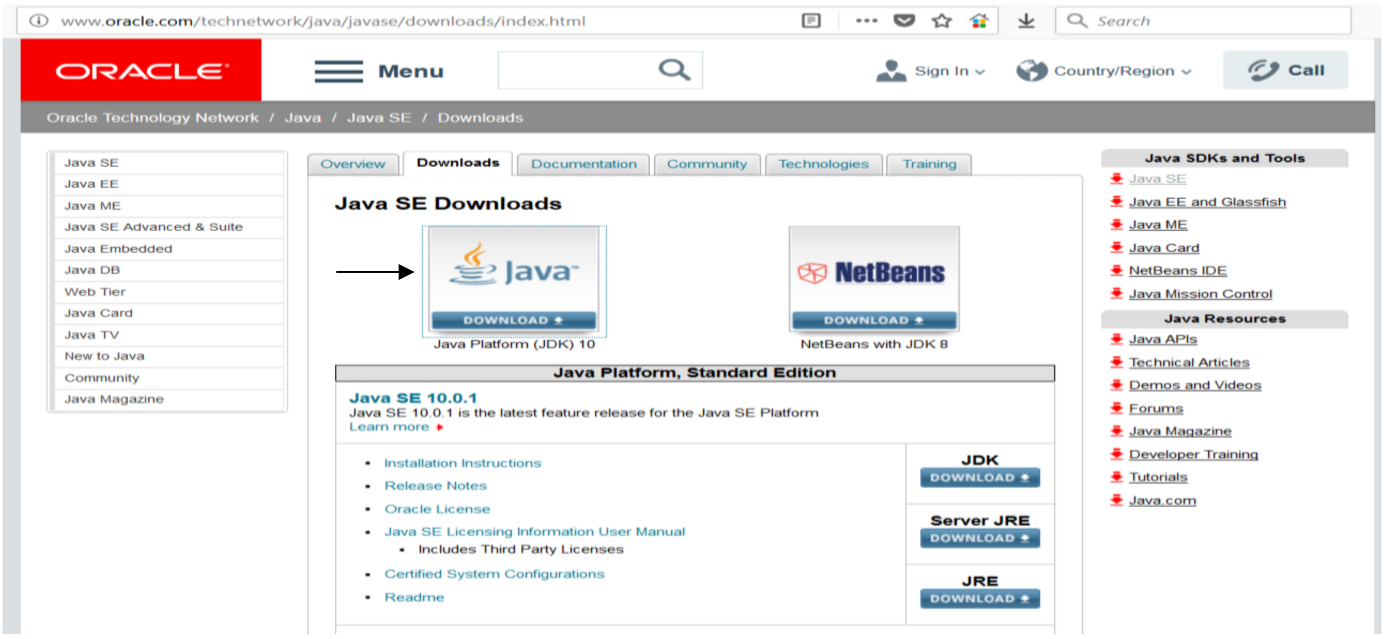
ASSIGNMENT-1

Program1:
Process for Installation of java software; set the path settings, Programs compilations and executions (JDK, JRE and JVM).

Solution

1) Installation of java software :

Step-1: Go to the given link
http://www.oracle.com/technetwork/java/javase/downloads/index.html


www.oracle.com/technetwork/java/javase/downloads/index.html

ORACLE · Menu · Sign In · Country/Region · Call

Oracle Technology Network / Java / Java SE / Downloads

Java SE
Java EE
Java ME
Java SE Advanced & Suite
Java Embedded
Java DB
Web Tier
Java Card
Java TV
New to Java
Community
Java Magazine

Overview | Downloads | Documentation | Community | Technologies | Training

Java SE Downloads

Java DOWNLOAD — Java Platform (JDK) 10
NetBeans DOWNLOAD — NetBeans with JDK 8

Java Platform, Standard Edition

Java SE 10.0.1
Java SE 10.0.1 is the latest feature release for the Java SE Platform
Learn more

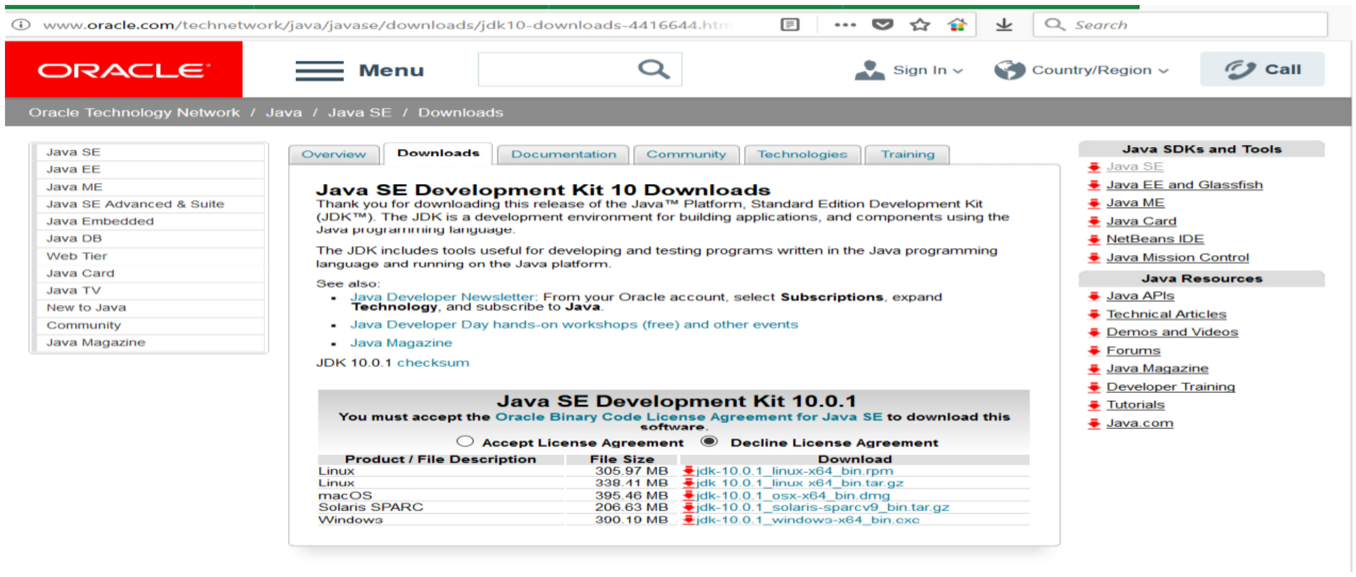
- Installation Instructions
- Release Notes
- Oracle License
- Java SE Licensing Information User Manual
  - Includes Third Party Licenses
- Certified System Configurations
- Readme

JDK DOWNLOAD
Server JRE DOWNLOAD
JRE DOWNLOAD

Java SDKs and Tools
Java SE
Java EE and Glassfish
Java ME
Java Card
NetBeans IDE
Java Mission Control

Java Resources
Java APIs
Technical Articles
Demos and Videos
Forums
Java Magazine
Developer Training
Tutorials
Java.com

Step-2: Click on java platform (JDK), select Accept License Agreement then selects the software download link.


www.oracle.com/technetwork/java/javase/downloads/jdk10-downloads-4416644.htm

ORACLE · Menu · Sign In · Country/Region · Call

Oracle Technology Network / Java / Java SE / Downloads

Java SE
Java EE
Java ME
Java SE Advanced & Suite
Java Embedded
Java DB
Web Tier
Java Card
Java TV
New to Java
Community
Java Magazine

Overview | Downloads | Documentation | Community | Technologies | Training

Java SE Development Kit 10 Downloads
Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:
- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

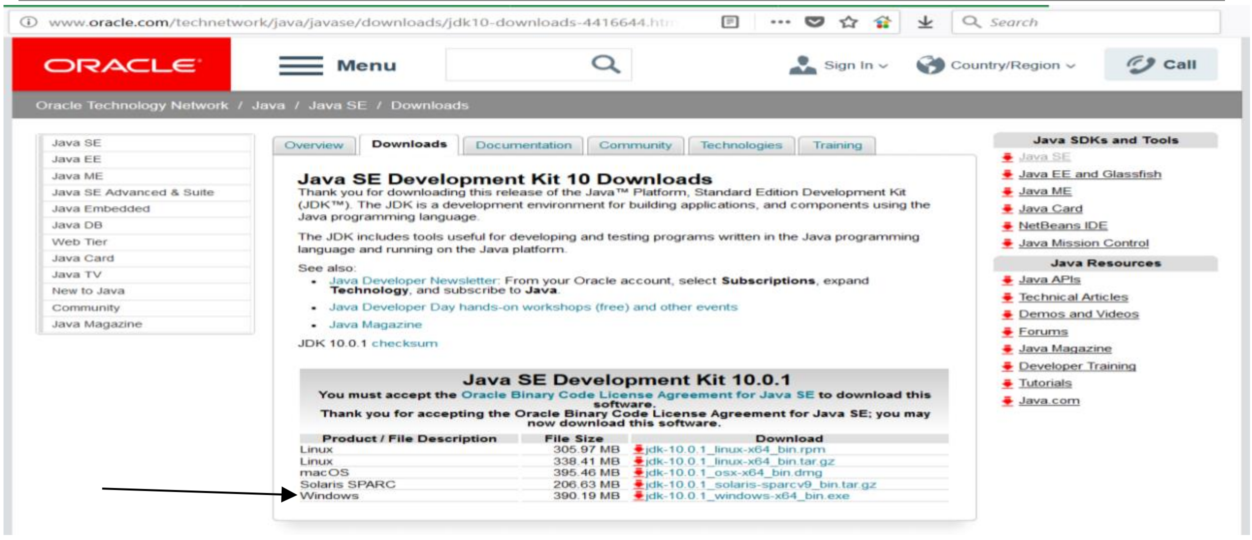
JDK 10.0.1 checksum

Java SE Development Kit 10.0.1
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Accept License Agreement [✓ Decline License Agreement]

Product / File Description	File Size	Download
Linux	305.97 MB	jdk-10.0.1_linux-x64_bin.rpm
Linux	338.41 MB	jdk 10.0.1_linux x64_bin.tar.gz
macOS	395.46 MB	jdk-10.0.1_osx-x64_bin.dmg
Solaris SPARC	206.63 MB	jdk-10.0.1_solaris-sparcv9_bin.tar.gz
Windows	300.10 MB	jdk-10.0.1_windows-x64_bin.exe

Java SDKs and Tools
Java SE
Java EE and Glassfish
Java ME
Java Card
NetBeans IDE
Java Mission Control

Java Resources
Java APIs
Technical Articles
Demos and Videos
Forums
Java Magazine
Developer Training
Tutorials
Java.com



www.oracle.com/technetwork/java/javase/downloads/jdk10-downloads-4416644.htm

Oracle Technology Network / Java / Java SE / Downloads

- Java SE
- Java EE
- Java ME
- Java SE Advanced & Suite
- Java Embedded
- Java DB
- Web Tier
- Java Card
- Java TV
- New to Java
- Community
- Java Magazine

Overview | Downloads | Documentation | Community | Technologies | Training

Java SE Development Kit 10 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK 10.0.1 checksum

Java SE Development Kit 10.0.1

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux	305.97 MB	jdk-10.0.1_linux-x64_bin.rpm
Linux	338.41 MB	jdk-10.0.1_linux-x64_bin.tar.gz
macOS	395.46 MB	jdk-10.0.1_osx-x64_bin.dmg
Solaris SPARC	206.63 MB	jdk-10.0.1_solaris-sparcv9_bin.tar.gz
Windows	390.19 MB	jdk-10.0.1_windows-x64_bin.exe

Java SDKs and Tools

- Java SE
- Java EE and Glassfish
- Java ME
- Java Card
- NetBeans IDE
- Java Mission Control

Java Resources

- Java APIs
- Technical Articles
- Demos and Videos
- Forums
- Java Magazine
- Developer Training
- Tutorials
- Java.com

Step-3: Download the software and install it in your computer.

2) Set the path in Java :

The path is required to be set for using tools such as javac, java etc.

If you are saving the java source file inside the jdk/bin directory, path is not required to be set because all the tools will be available in the current directory.

But If you are having your java file outside the jdk/bin folder, it is necessary to set path of JDK.

There are 2 ways to set java path:

1. temporary
2. permanent

1) How to set Temporary Path of JDK in Windows

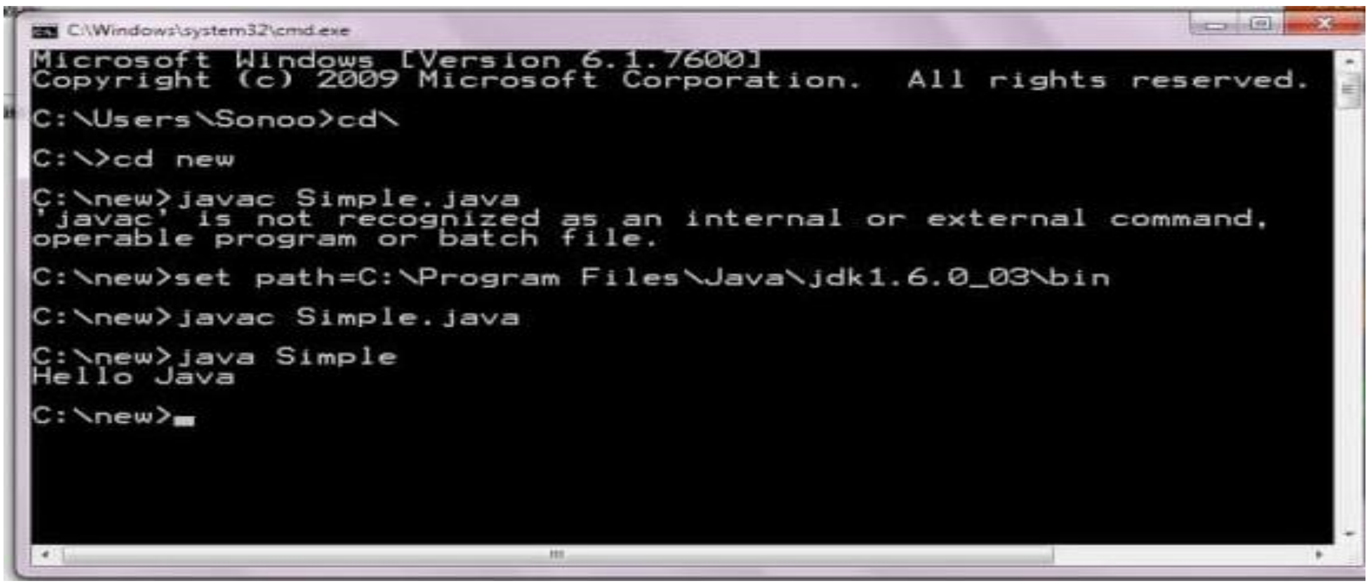
To set the temporary path of JDK, you need to follow following steps:

- Open command prompt
- Copy the path of jdk/bin directory
- Write in command prompt: set path=copied_path

For Example:

```
set path=C: \Program Files\Java\jdk1.6.0_23\bin
```

Let's see it in the figure given below:


C:\Windows\system32\cmd.exe

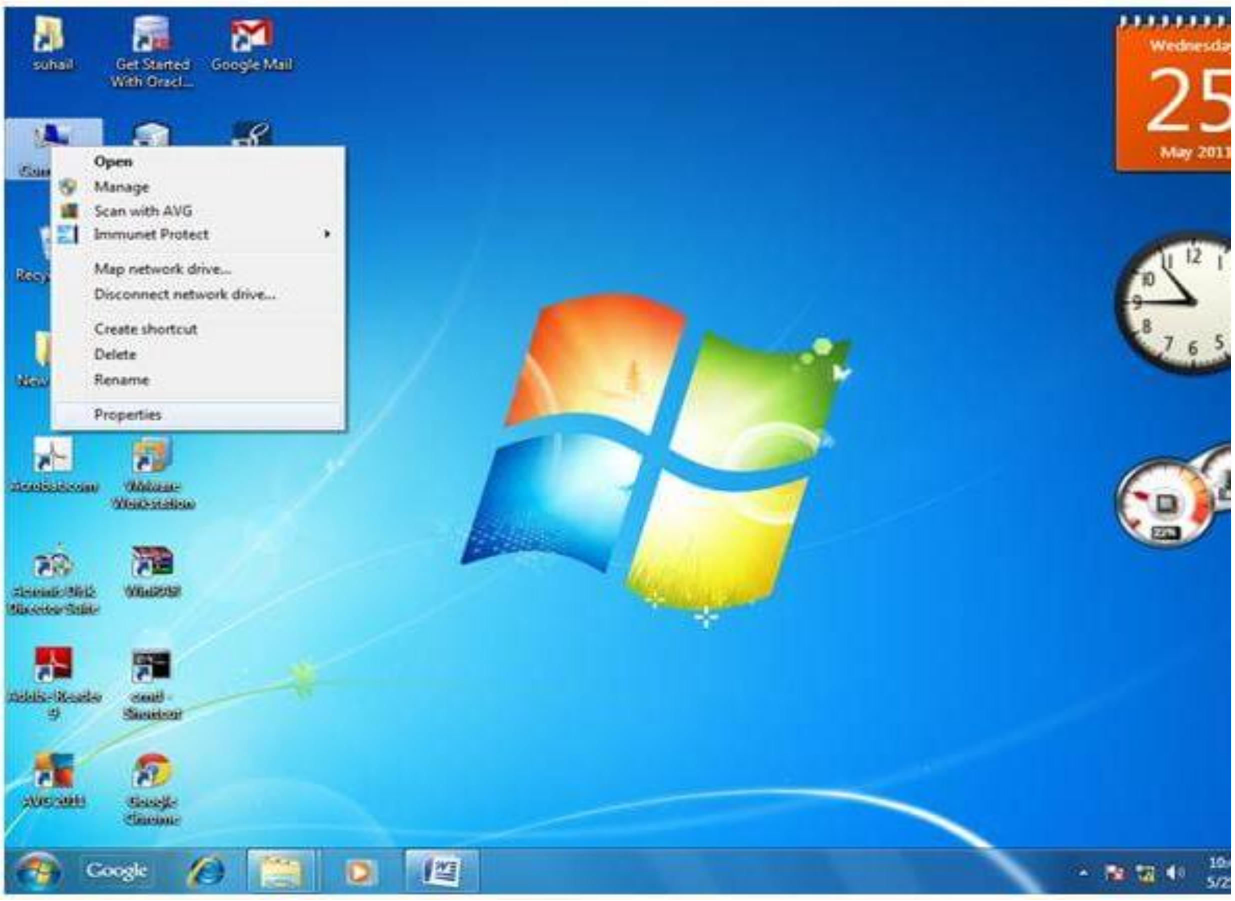
```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Sonoo>cd\

C:\>cd new

C:\new>javac Simple.java
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\new>set path=C:\Program Files\Java\jdk1.6.0_03\bin

C:\new>javac Simple.java

C:\new>java Simple
Hello Java

C:\new>
```

2) How to set Permanent Path of JDK in Windows

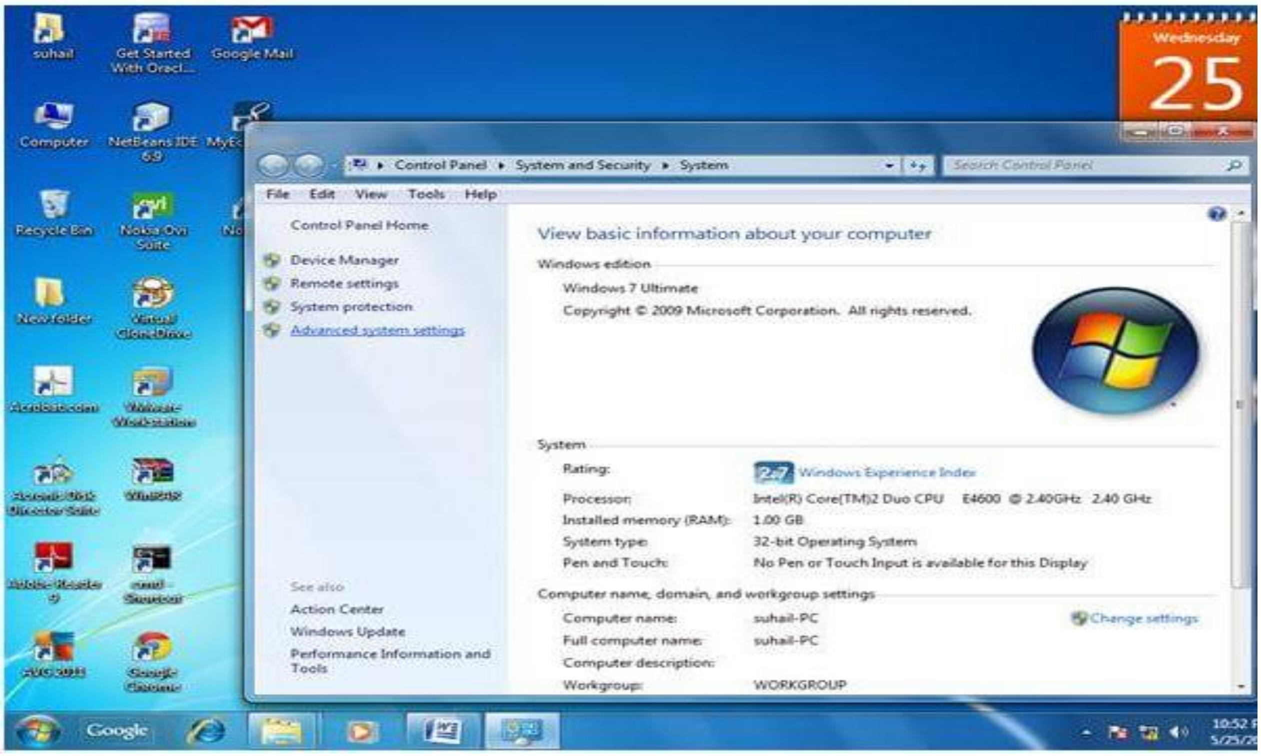
For setting the permanent path of JDK, you need to follow these steps:

Go to MyComputer properties -> advanced tab -> environment variables -> new tab of user variable -> write path in variable name -> write path of bin folder in variable value -> ok -> ok -> ok. For Example:

1)Go to MyComputer properties



2)click on advanced tab



View basic information about your computer

Windows edition

Windows 7 Ultimate

Copyright © 2009 Microsoft Corporation. All rights reserved.

System

Rating: 2.7 Windows Experience Index

Processor: Intel(R) Core(TM)2 Duo CPU E4600 @ 2.40GHz 2.40 GHz

Installed memory (RAM): 1.00 GB

System type: 32-bit Operating System

Pen and Touch: No Pen or Touch Input is available for this Display

Computer name, domain, and workgroup settings

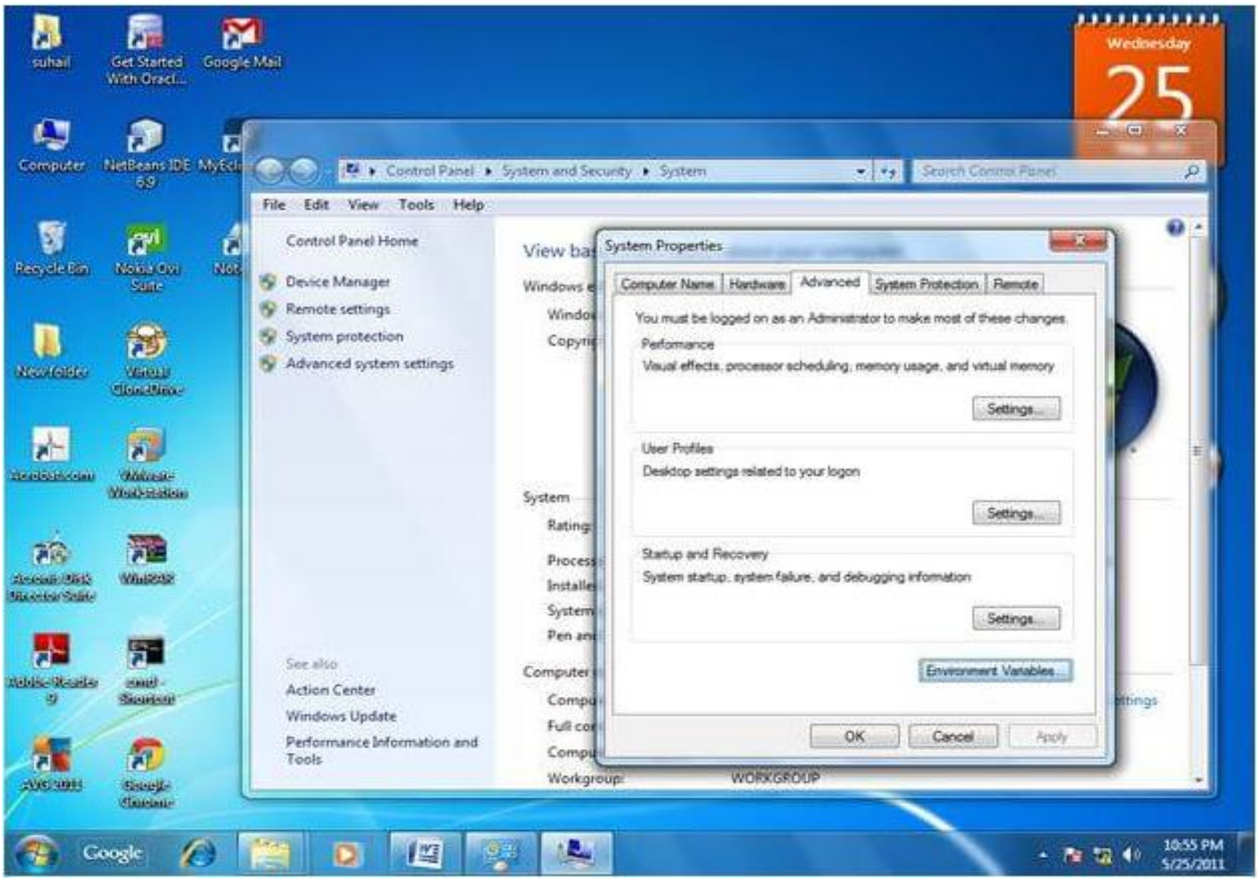
Computer name: suhail-PC

Full computer name: suhail-PC

Computer description:

Workgroup: WORKGROUP

3)click on environment variables



You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

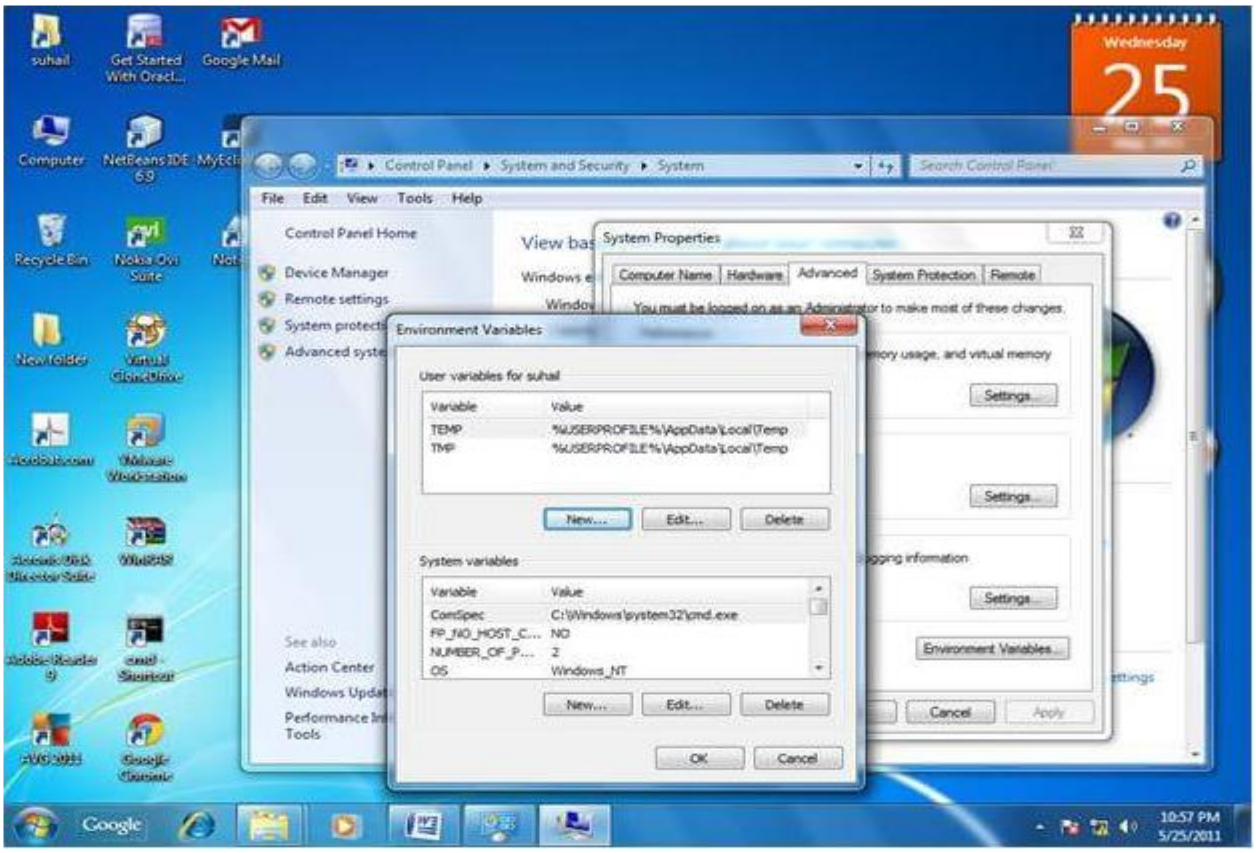
User Profiles

Desktop settings related to your logon

Startup and Recovery

System startup, system failure, and debugging information

4)click on new tab of user variables



You must be logged on as an Administrator to make most of these changes.

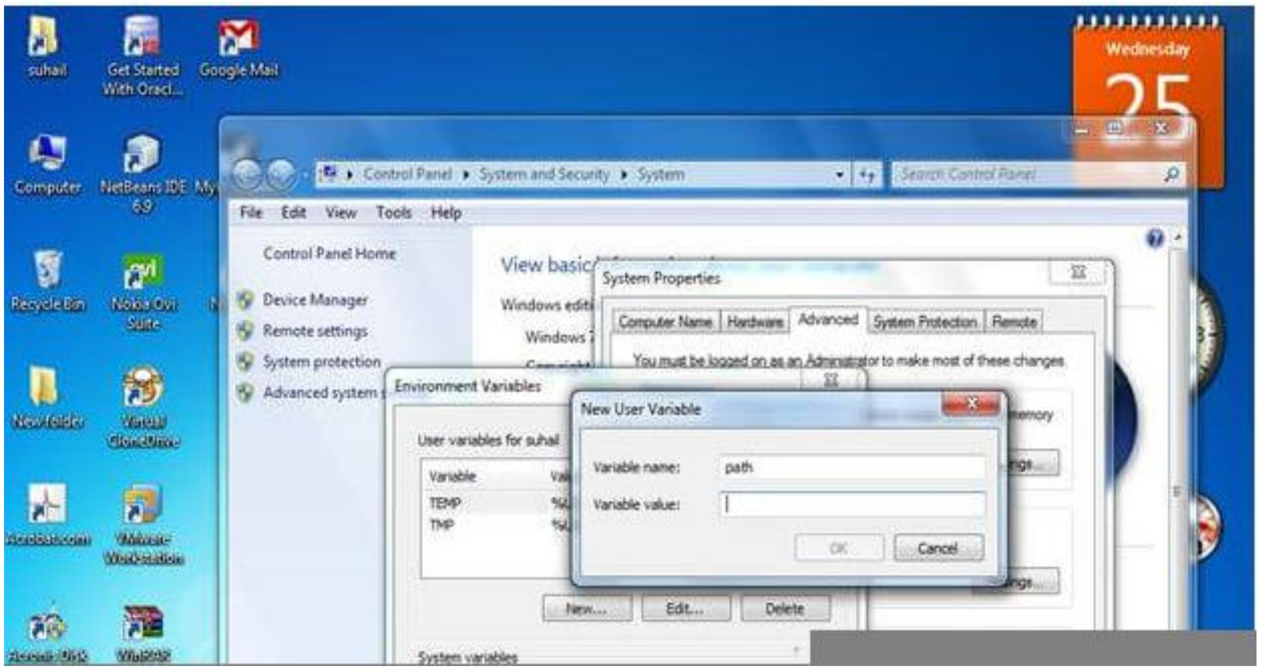
User variables for suhail

Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_P...	2
OS	Windows_NT

5)write path in variable name

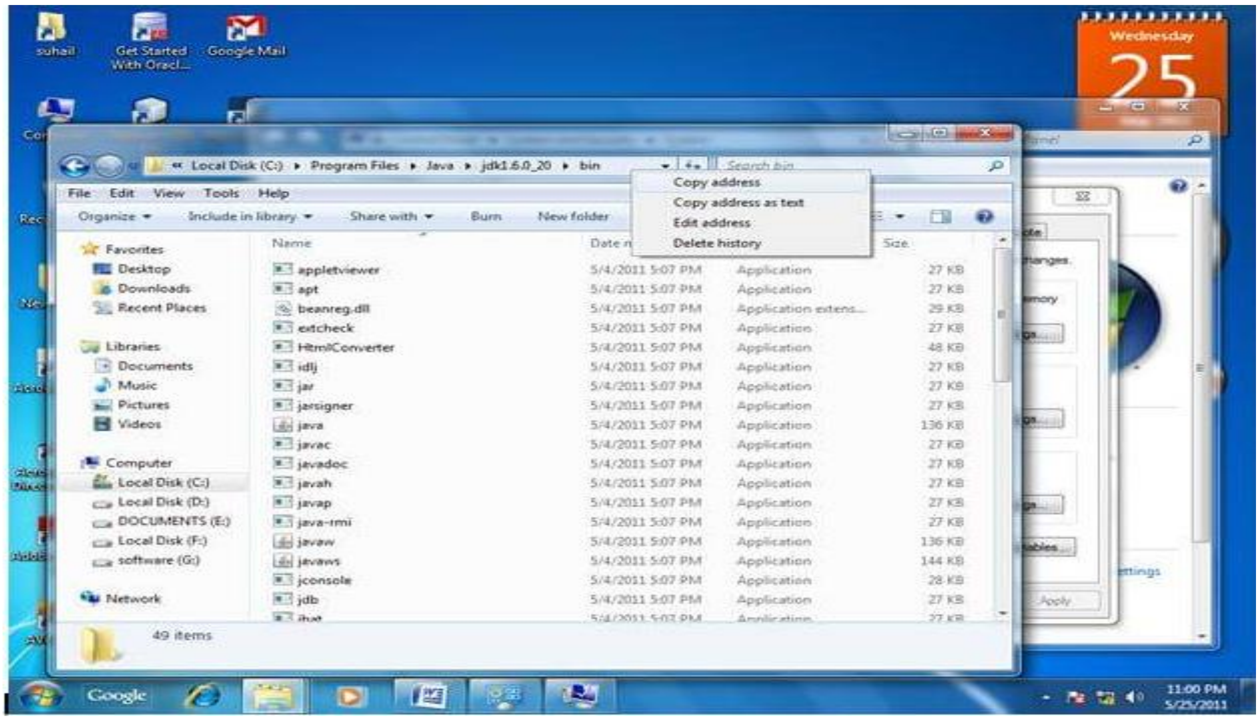


You must be logged on as an Administrator to make most of these changes.

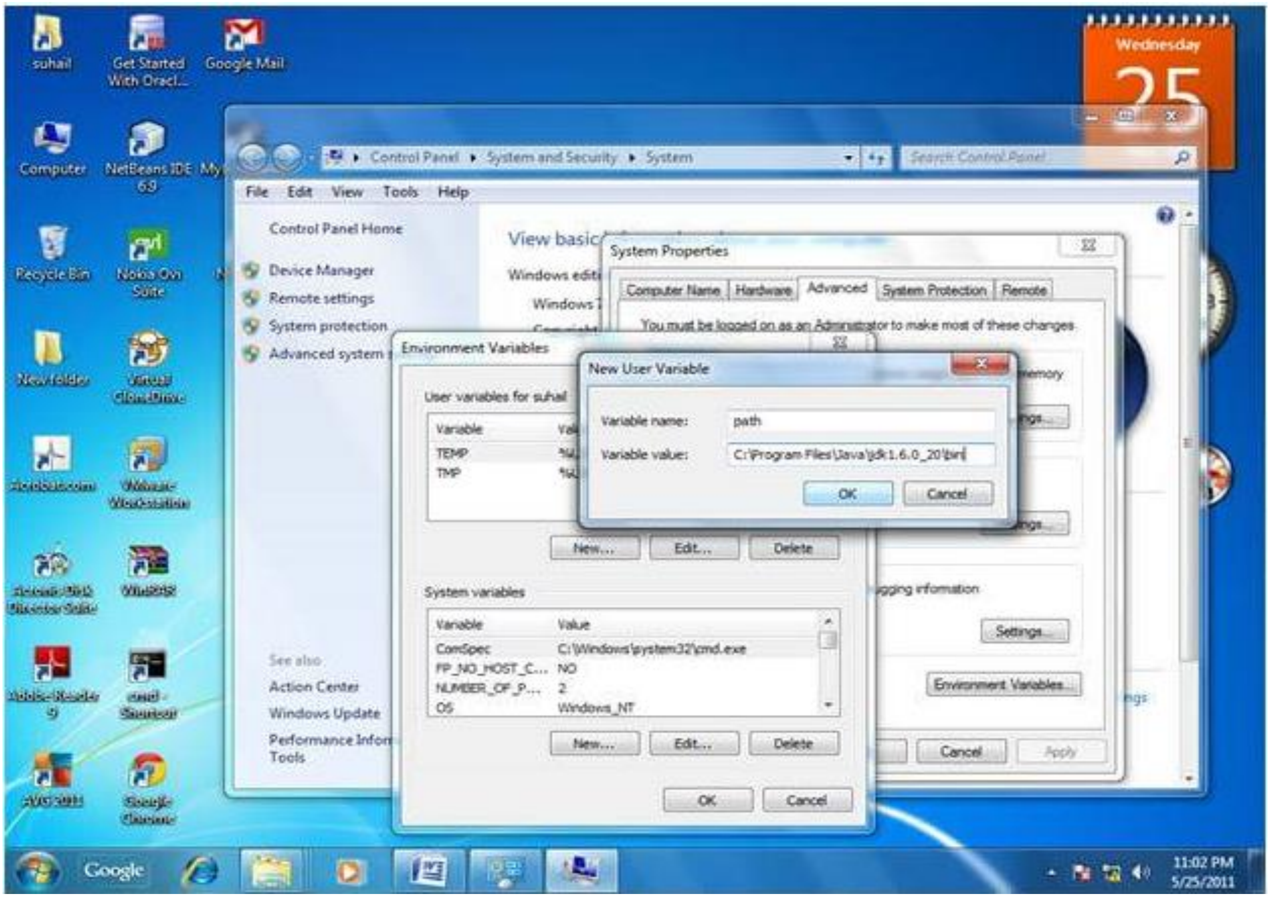
Variable name: path

Variable value:

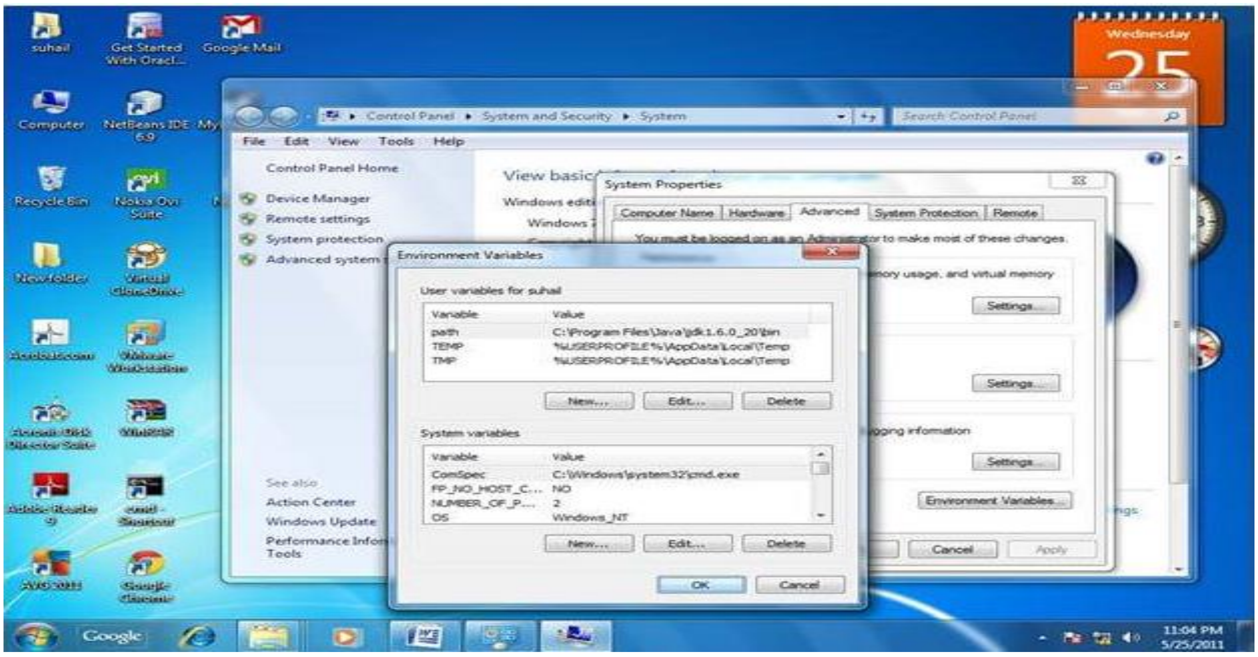
6)Copy the path of bin folder



7)paste path of bin folder in variable value



8)click on ok button



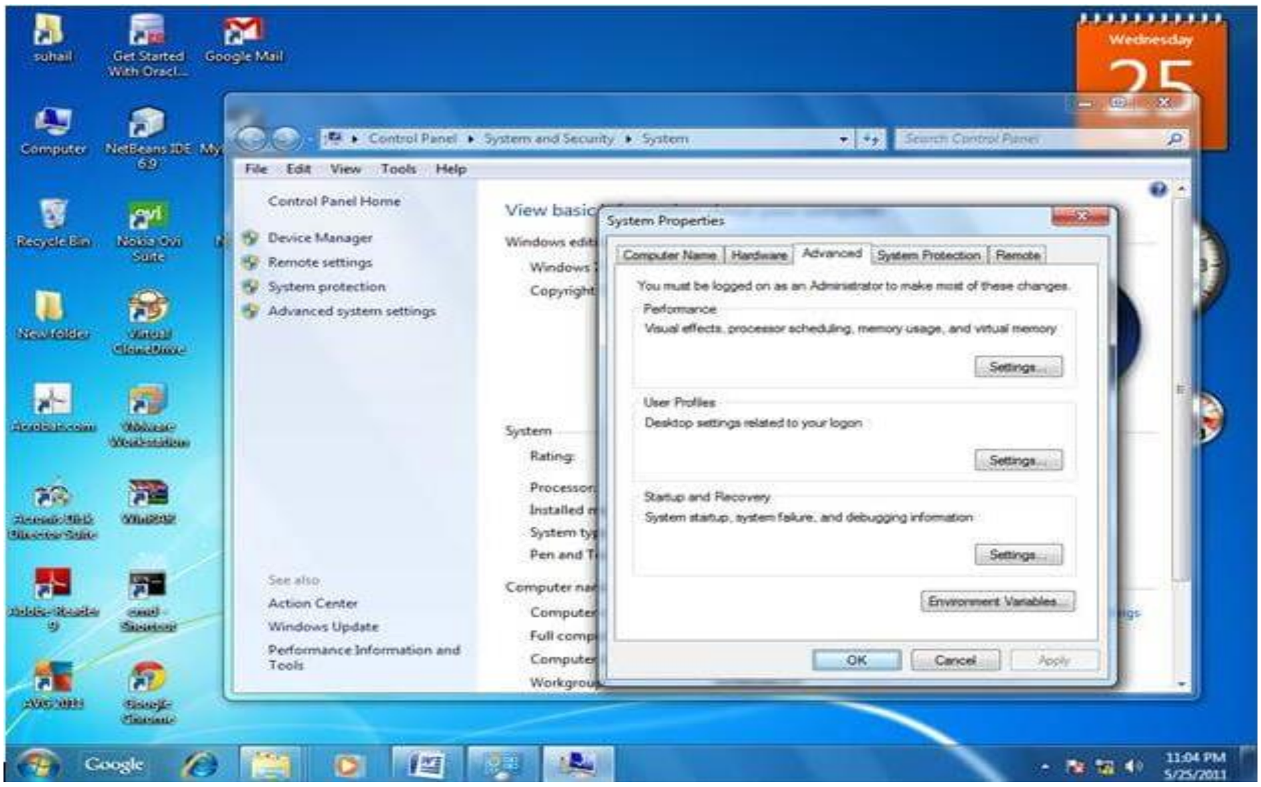
User variables for suhail

Variable	Value
path	C:\Program Files\Java\jdk1.6.0_20\bin
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_P...	2
OS	Windows_NT

9)click on ok button


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your logon
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

Now your permanent path is set. You can now execute any program of java from any drive.

3) Programs compilations and executions (JDK, JRE and JVM)

JVM

JVM (Java Virtual Machine) is an abstract machine. It is called virtual machine because it doesn't physically exist. It is a specification that provides runtime environment in which java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java byte code.

JVMs are available for many hardware and software platforms. JVM, JRE and JDK are platform dependent because configuration of each OS are different from each other. But, Java is platform independent. There are three notions of the JVM: *specification*, *implementation*, and *instance*.

The JVM performs following main Programs:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

What is JVM

It is:

1. A specification where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.
2. An implementation Its implementation is known as JRE (Java Runtime Environment).
3. Runtime Instance Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

What it does

The JVM performs following operation:

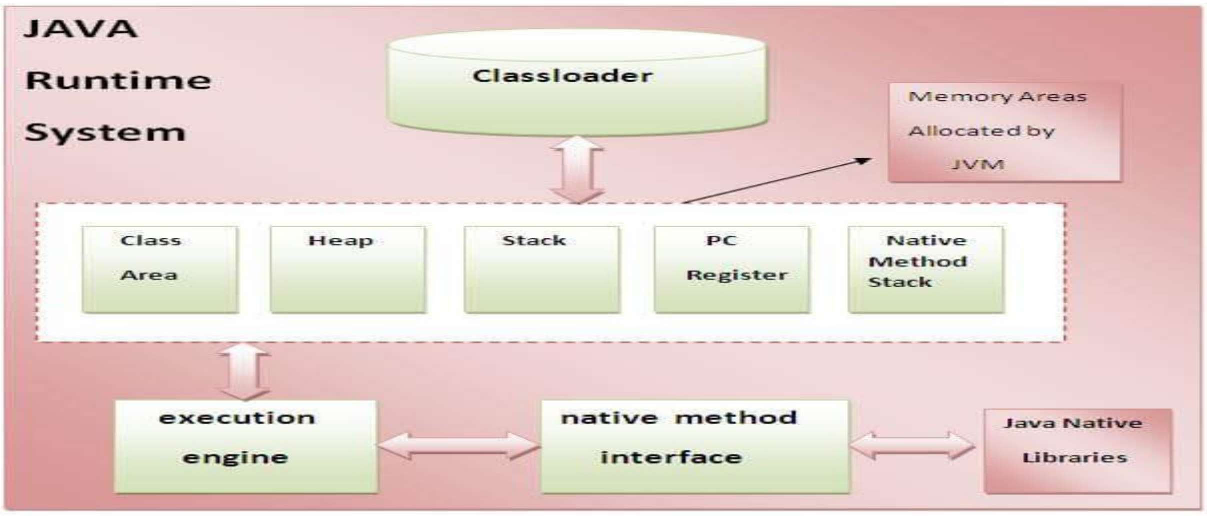
- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap
- Fatal error reporting etc.

Internal Architecture of JVM

Let's understand the internal architecture of JVM. It contains classloader, memory area, execution engine etc.



1) Classloader

Classloader is a subsystem of JVM that is used to load class files.

2) Class (Method) Area

Class (Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap

It is the runtime data area in which objects are allocated.

4) Stack

Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) **Program Counter Register**
PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

6) **Native Method Stack:**
It contains all the native methods used in the application.

7) **Execution Engine**
It contains:

1) A virtual processor

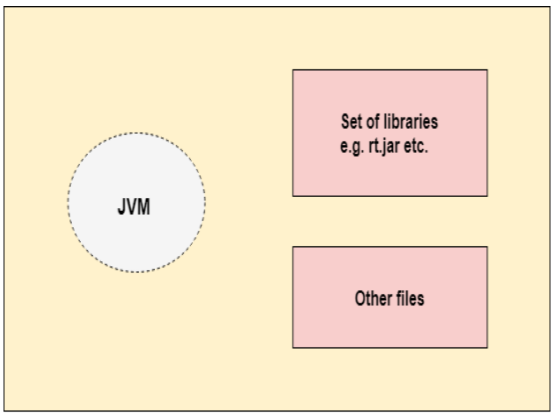
2) Interpreter: Read bytecode stream then execute the instructions.

3) Just-In-Time (JIT) compiler: It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here, the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

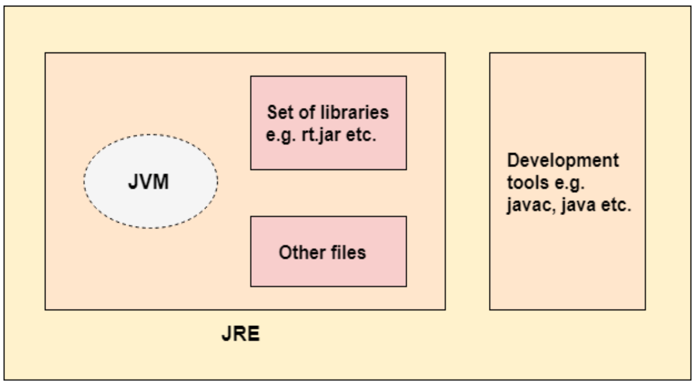
JRE

JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing java applications. It is used to provide runtime environment. It is the implementation of JVM. It physically exists. It contains set of libraries + other files that JVM uses at runtime.

Implementation of JVMs is also actively released by other companies besides Sun Micro Systems.



JRE



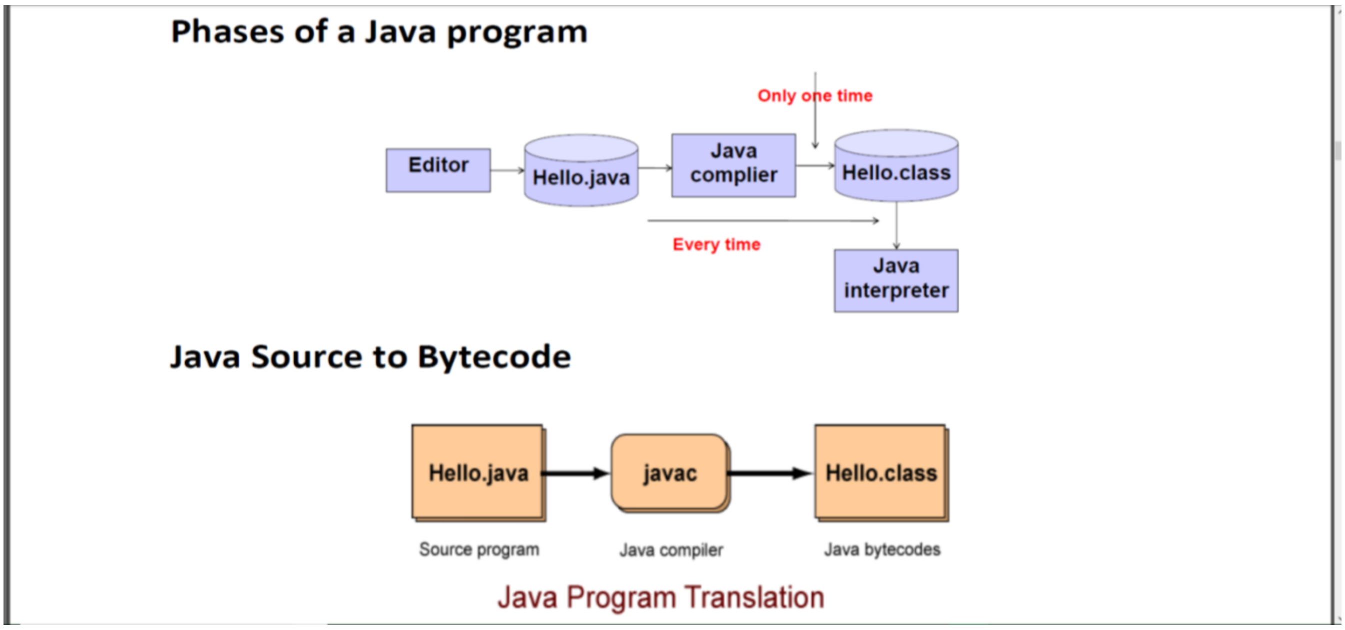
JDK

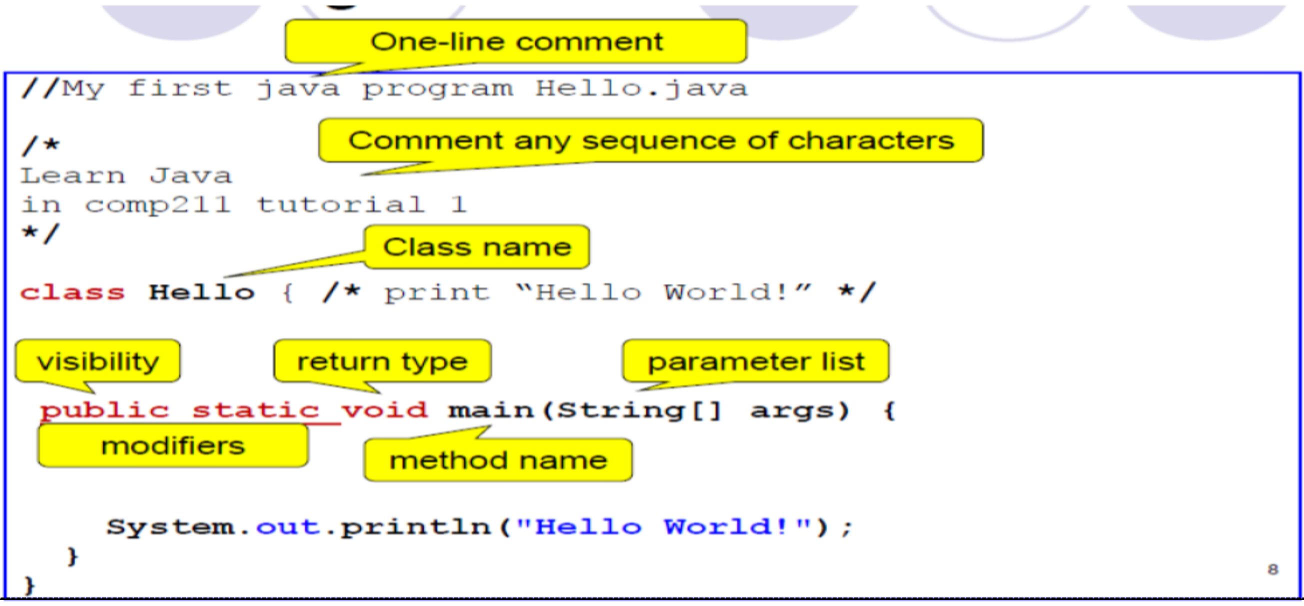
JDK

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop java applications and applets. It physically exists. It contains JRE + development tools.
JDK is an implementation of any one of the below given Java Platforms released by Oracle corporation:

- Standard Edition Java Platform
- Enterprise Edition Java Platform
- Micro Edition Java Platform

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) etc. to complete the development of a Java Application.





```
//My first java program Hello.java

/*
Learn Java
in comp211 tutorial 1
*/

class Hello { /* print "Hello World!" */

 public static void main(String[] args) {

    System.out.println("Hello World!");
  }
}
```

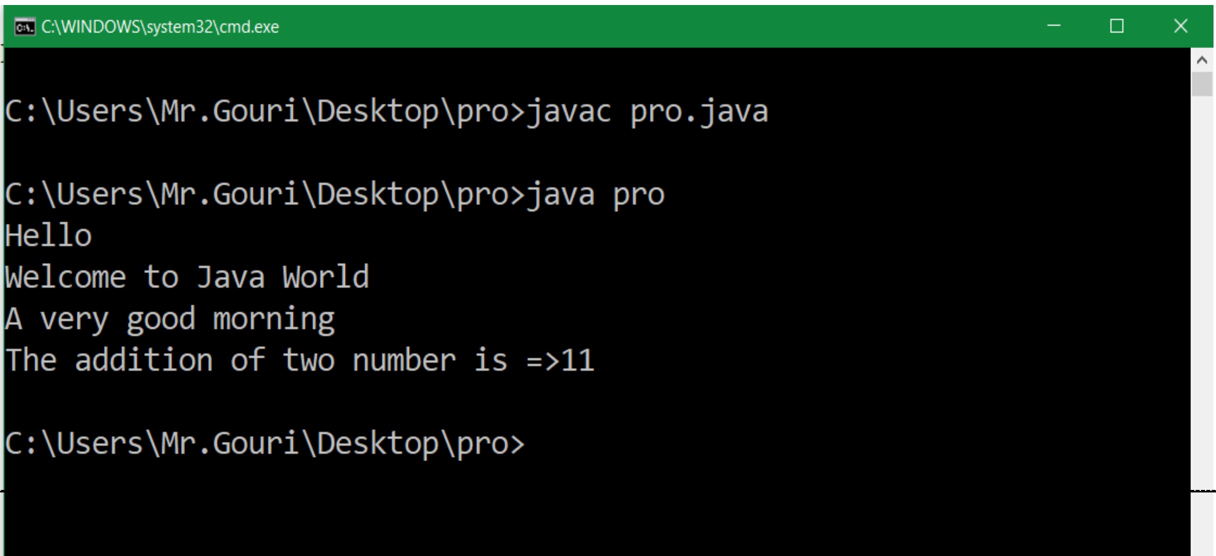
Program2:

Write a program to display ("Hello", "Welcome to Java World", "A very good morning") and addition of two numbers by using function.

Solution:

```
Class Demo
{
    public static void main(String arg[])
   {
        System.out.println("Hello");
        System.out.println("Welcome to Java World");
        System.out.println("A very good morning");
        Demo obj = new obj();
        obj.add(2,7);

     }
     int result;
     void add ( int x, int y )
     {
          result = x + y;
         System.out.println("The addition of two number is =>"+result);
      }
  }
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac pro.java

C:\Users\Mr.Gouri\Desktop\pro>java pro
Hello
Welcome to Java World
A very good morning
The addition of two number is =>11

C:\Users\Mr.Gouri\Desktop\pro>
```

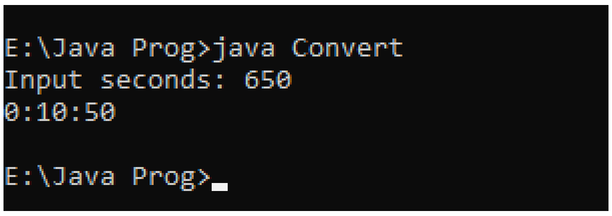
Program3:

Write a program Convert seconds to hour, minute and seconds

Solution

```
import java.util.*;
 public class Convert {
 public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        System.out.print("Input seconds: ");
         int seconds = in.nextInt();
        int p1 = seconds % 60;
        int p2 = seconds / 60;
        int p3 = p2 % 60;
        p2 = p2 / 60;
        System.out.print( p2 + ":" + p3 + ":" + p1);
       System.out.print("\n");
    }
 }
```

Output:



```
E:\Java Prog>java Convert
Input seconds: 650
0:10:50

E:\Java Prog>_
```

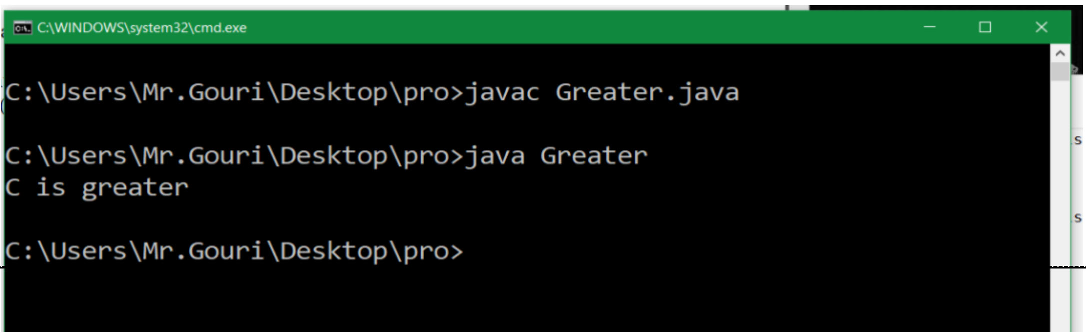
Program4:

Write a program to find largest of 3 numbers.

Solution:

```
class Greater
{
    public static void main(String arg[])
   {
            int a = 98,b = 87,c = 99;
            if(a>b)
            {
                    if(a>c)
                    {
                            System.out.println("A is greater");
                    }
                    else
                    {
                            System.out.println("C is greater");
                    }
            }
            else
            {
                    if(b>c)
                    {
                            System.out.println("B is greater");
                    }
                    else
                    {
                            System.out.println("C is greater");
                    }
             }

      }
 }
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac Greater.java

C:\Users\Mr.Gouri\Desktop\pro>java Greater
C is greater

C:\Users\Mr.Gouri\Desktop\pro>
```

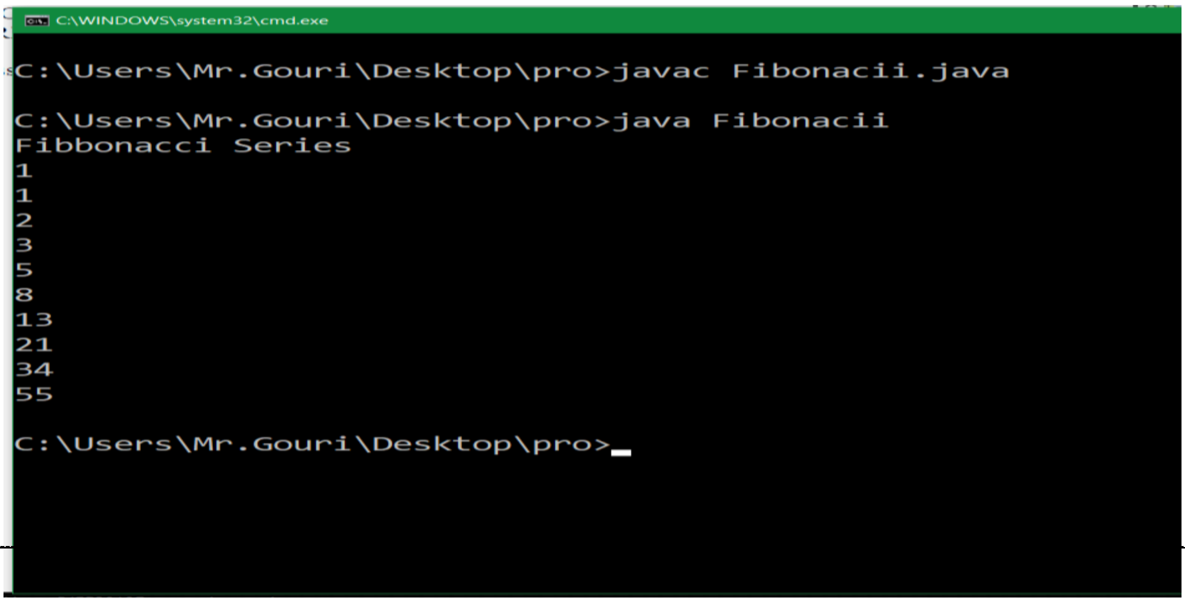
Program5:

Write a program to print Fibonacci Series using loop.

Solution:

```
class Fibonacii
{
    public static void main(String arg[])
    {
                int i = 1,j = 1,k = 0;
                System.out.println("Fibbonacci Series");
                while(i<=10)
                {
                        System.out.println(+j);
                        j = j+k;
                        k = j-k;
                        i++;
                }

     }
}
```

Output:



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Mr.Gouri\Desktop\pro>javac Fibonacii.java

C:\Users\Mr.Gouri\Desktop\pro>java Fibonacii
Fibbonacci Series
1
1
2
3
5
8
13
21
34
55

C:\Users\Mr.Gouri\Desktop\pro>_
```

ASSIGNMENT-2

Program1:

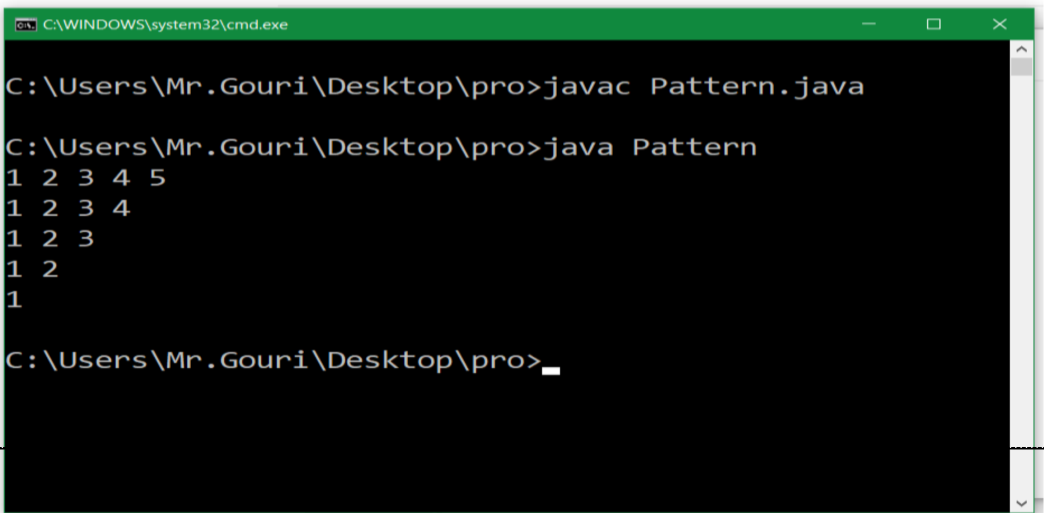
Write a Java program that prints the following pattern

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Solution:

```
public class Pattern
{
    public static void main(String[] args)
    {
            int rows = 5;

            for(int i = rows; i >= 1; --i)
           {
                for(int j = 1; j <= i; ++j)
                {
                    System.out.print(j + " ");
                }
                System.out.println();
           }
    }
}
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac Pattern.java

C:\Users\Mr.Gouri\Desktop\pro>java Pattern
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

C:\Users\Mr.Gouri\Desktop\pro>
```

Program2:

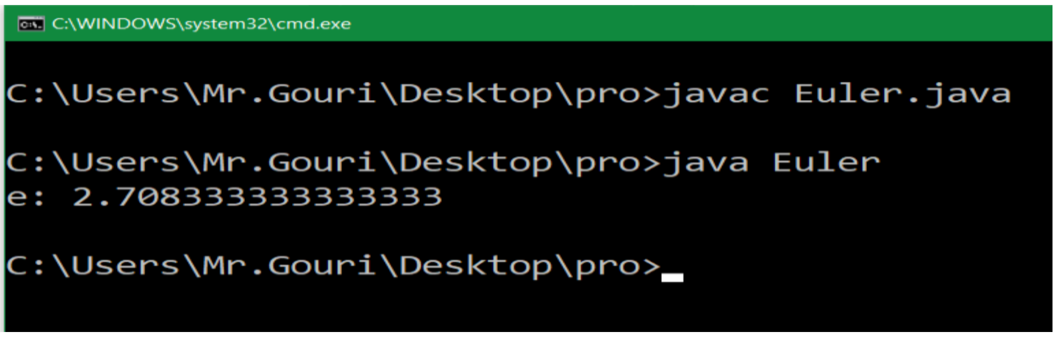
Write a Java program that calculate mathematical constant ‘e’ using the formula e=1+1/2! +1/3! +........ up to 5.

Solution:

```
public class Euler
{
    public static void main(String args[])
    {
             double e = 1;

             for(int x = 1; x < 5; x++)
             e = e + 1/(double)fictorial(x);
             System.out.println("e: "+ e);

    }
    public static double fictorial(int n)
    {
              int x;
              if(n < 1)
              {
                     return 0;
               }
              else
             {
                     if(n == 1)
                     {
                              return 1;
                     }
                     else if(n > 1)
                     {
                              return n*fictorial(n-1);
                     }
              }
              return 0;
```

```
        }
    }
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac Euler.java

C:\Users\Mr.Gouri\Desktop\pro>java Euler
e: 2.708333333333333

C:\Users\Mr.Gouri\Desktop\pro>_
```

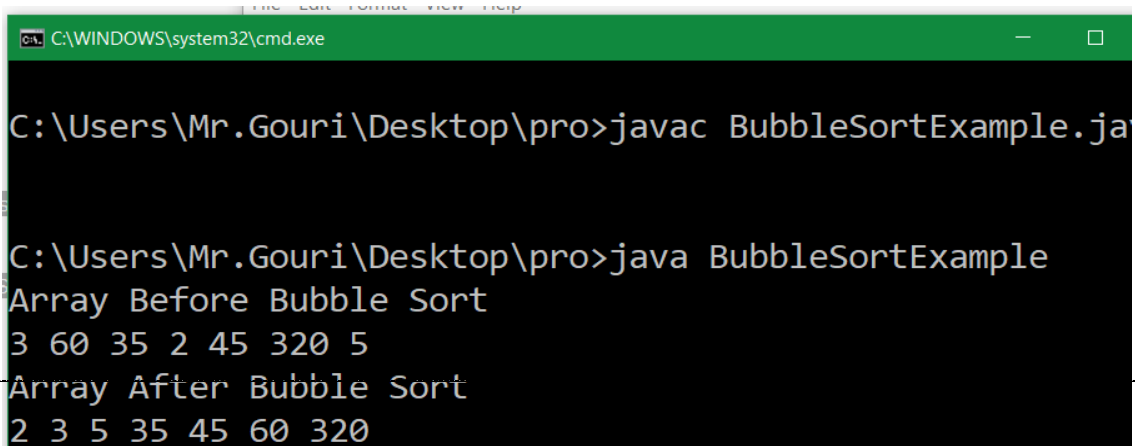
Program3:

Write a Java program to sort the elements using bubble sort.

Solution:

```
public class BubbleSortExample
{
   static void bubbleSort(int[] arr)
   {
      int n = arr.length;
      int temp = 0;
       for(int i=0; i < n; i++)
         {
               for(int j=1; j < (n-i); j++)
                   {
                     if(arr[j-1] > arr[j])

                      {
                           //swap elements
                           temp = arr[j-1];
                           arr[j-1] = arr[j];
                           arr[j] = temp;
                      }

                   }
           }

     }

public static void main(String[] args)
{
   int arr[] ={3,60,35,2,45,320,5};
   System.out.println("Array Before Bubble Sort");
   for(int i=0; i < arr.length; i++)
    {
        System.out.print(arr[i] + " ");
    }
        System.out.println();

        bubbleSort(arr);//sorting array elements
                                     using bubble sort

     System.out.println("Array After Bubble Sort");
     for(int i=0; i < arr.length; i++)
          {
              System.out.print(arr[i] + " ");
          }
       }

}
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac BubbleSortExample.ja


C:\Users\Mr.Gouri\Desktop\pro>java BubbleSortExample
Array Before Bubble Sort
3 60 35 2 45 320 5
Array After Bubble Sort
2 3 5 35 45 60 320
```

Program4:

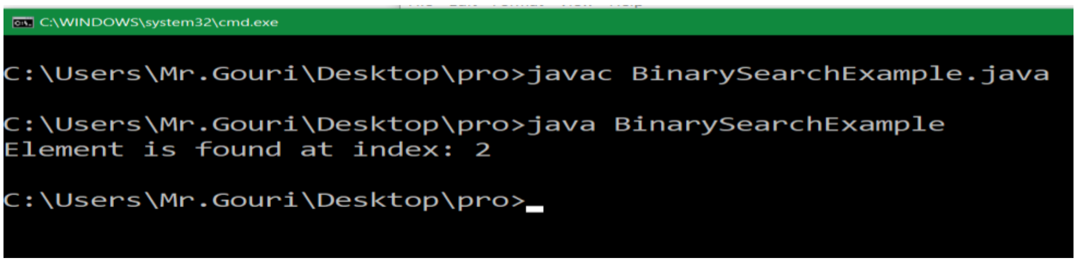
Write a Java program to search an element using binary search.

Solution:

```
class BinarySearchExample
{
    public static void binarySearch(int arr[], int first, int last, int key)
    {
            int mid = (first + last)/2;
            while( first <= last )
            {
                    if ( arr[mid] < key )
                    {
                            first = mid + 1;
                    }
                    else if ( arr[mid] == key )
                    {
                            System.out.println("Element is found at index: " + mid);
                            break;
                    }
                    else
                    {
                            last = mid - 1;
                    }
                    mid = (first + last)/2;
             }
             if ( first > last )
            {
                    System.out.println("Element is not found!");
            }
     }
     public static void main(String args[])
    {
             int arr[] = {10,20,30,40,50};
             int key = 30;
```

```
                int last=arr.length-1;
                 binarySearch(arr,0,last,key);
    }
  }
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac BinarySearchExample.java

C:\Users\Mr.Gouri\Desktop\pro>java BinarySearchExample
Element is found at index: 2

C:\Users\Mr.Gouri\Desktop\pro>
```

Program5:

Write a Java program to create and display unique three-digit number using 1, 2, 3, 4. Also count how many three-digit numbers are there.

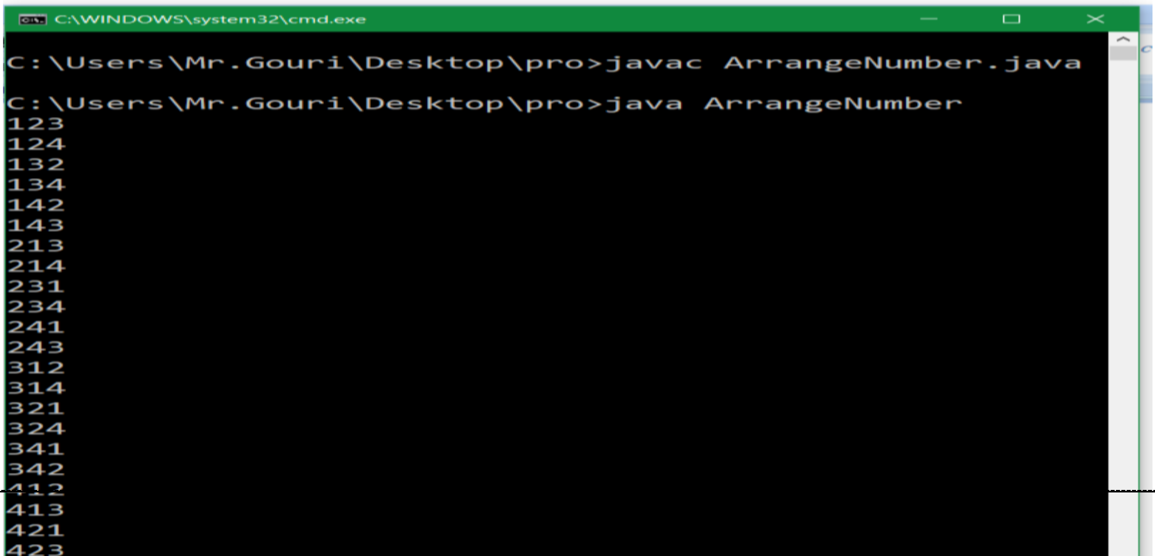
Solution:

Total number of the three-digit-number is 24

123	124	132	134
142	234	321	413
143	241	324	421
213	243	341	423
214	312	342	431
231	314	412	432

```
public class ArrangeNumber
{
      public static void main(String[] args)
      {
            int amount = 0;
            for(int i = 1; i <= 4; i++)
            {
                  for(int j = 1; j <= 4; j++)
                  {
                        for(int k = 1; k <= 4; k++)
                        {
                           if(k != i && k != j && i != j)
                           {
                                 amount++;
                                 System.out.println(i + "" + j + "" + k);
                           }
                        }
                  }
            }
      System.out.println("Total number of the three-digit-number is " + amount);
      }
}
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac ArrangeNumber.java

C:\Users\Mr.Gouri\Desktop\pro>java ArrangeNumber
123
124
132
134
142
143
213
214
231
234
241
243
312
314
321
324
341
342
412
413
421
423
```

ASSIGNMENT-3

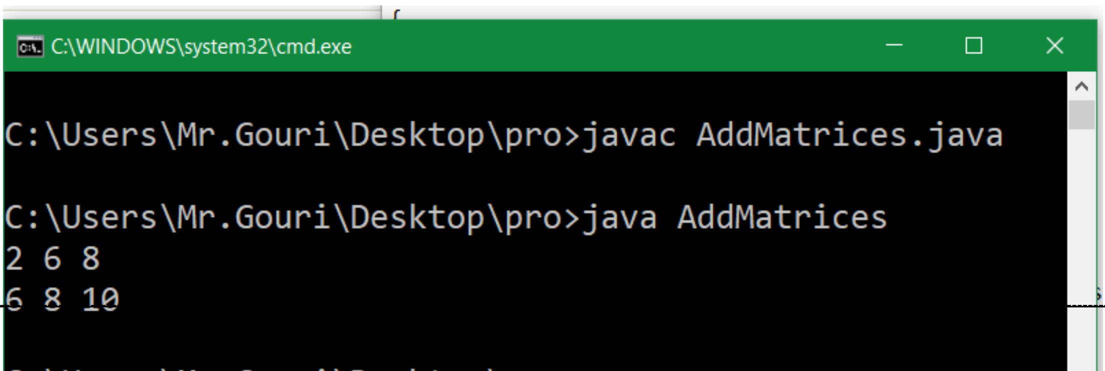
Program1:

Write a Java program to add two matrices.

Solution:

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

       //creating two matrices

       int a[][]={{1,3,4},{3,4,5}};
       int b[][]={{1,3,4},{3,4,5}};

       //creating another matrix to store the sum of two matrices

       int c[][]=new int[2][3];

       //adding and printing addition of 2 matrices

       for(int i=0;i<2;i++)
       {
            for(int j=0;j<3;j++)
            {
                    c[i][j]=a[i][j]+b[i][j];
                    System.out.print(c[i][j]+" ");
            }
            System.out.println();//new line
        }
    }
}
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac AddMatrices.java

C:\Users\Mr.Gouri\Desktop\pro>java AddMatrices
2 6 8
6 8 10
```

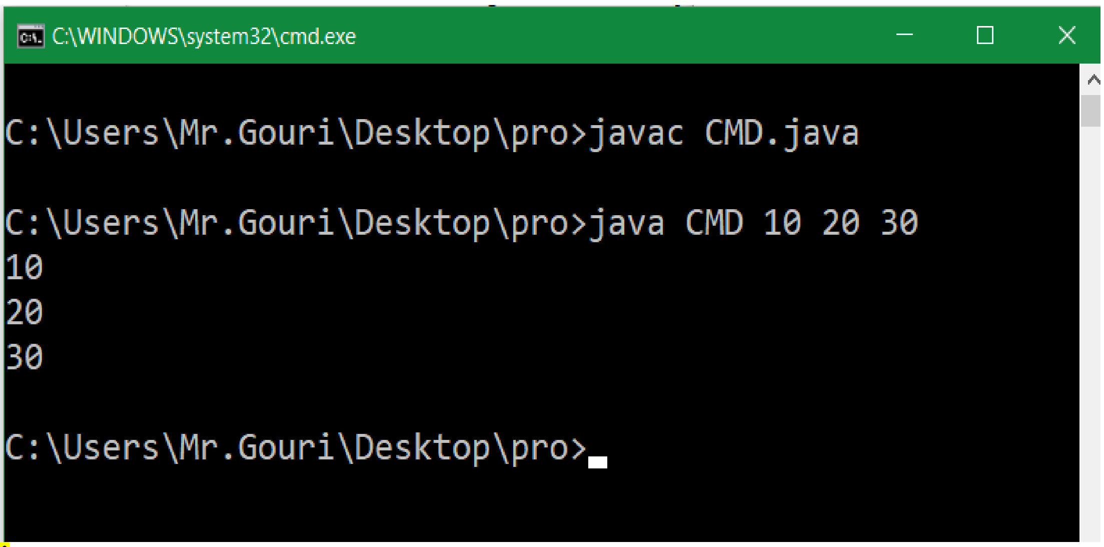
Program2:

Write a Java program by using a Command Line Argument (CMD).

Solution:

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

     int num1=Integer.parseInt(args[0]);
     int num2=Integer.parseInt(args[1]);
     int sum=num1+num2;
     System.out.println(" "+sum);

     }
}
```

Output:



C:\WINDOWS\system32\cmd.exe

```
C:\Users\Mr.Gouri\Desktop\pro>javac CMD.java

C:\Users\Mr.Gouri\Desktop\pro>java CMD 10 20 30
10
20
30

C:\Users\Mr.Gouri\Desktop\pro>
```

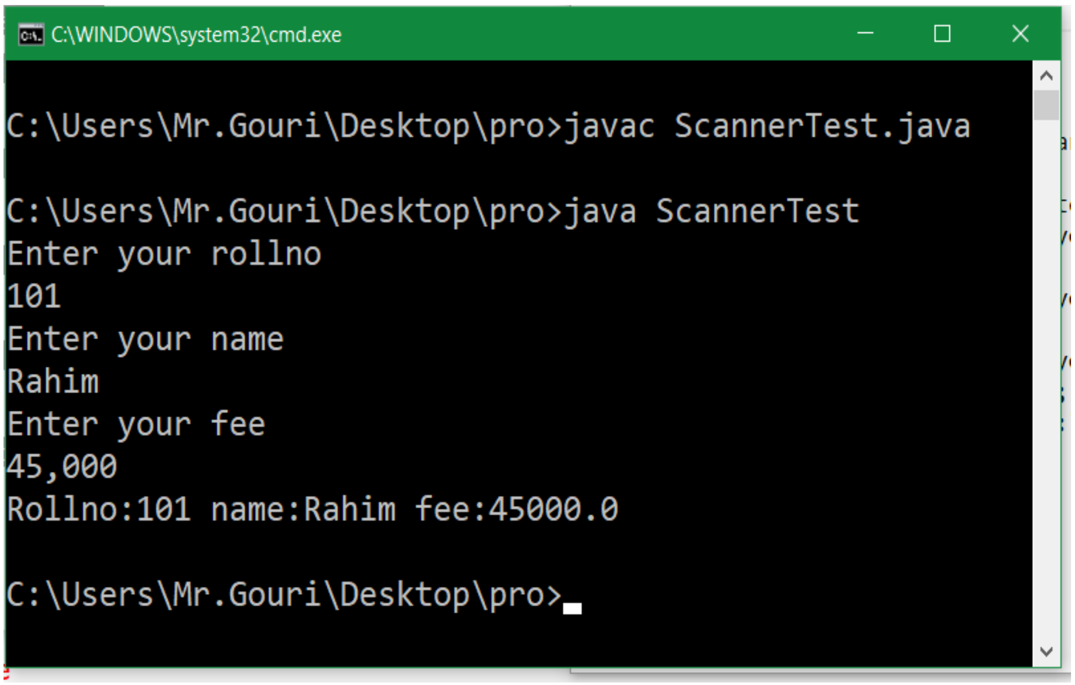
Program3:

Write a Java program to define Scanner class which reads the int, string and double value as an input.

Solution:

```
import java.util.Scanner;
class ScannerTest
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner (System.in);

        System.out.println("Enter your rollno");
        int rollno=sc.nextInt();

        System.out.println("Enter your name");
        String name=sc.next();

        System.out.println("Enter your fee");
        double fee=sc.nextDouble();

        System.out.println("Rollno:"+rollno+" name:"+name+" Fee:"+fee);
        sc.close();
    }
}
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac ScannerTest.java

C:\Users\Mr.Gouri\Desktop\pro>java ScannerTest
Enter your rollno
101
Enter your name
Rahim
Enter your fee
45,000
Rollno:101 name:Rahim fee:45000.0

C:\Users\Mr.Gouri\Desktop\pro>
```

Program4:

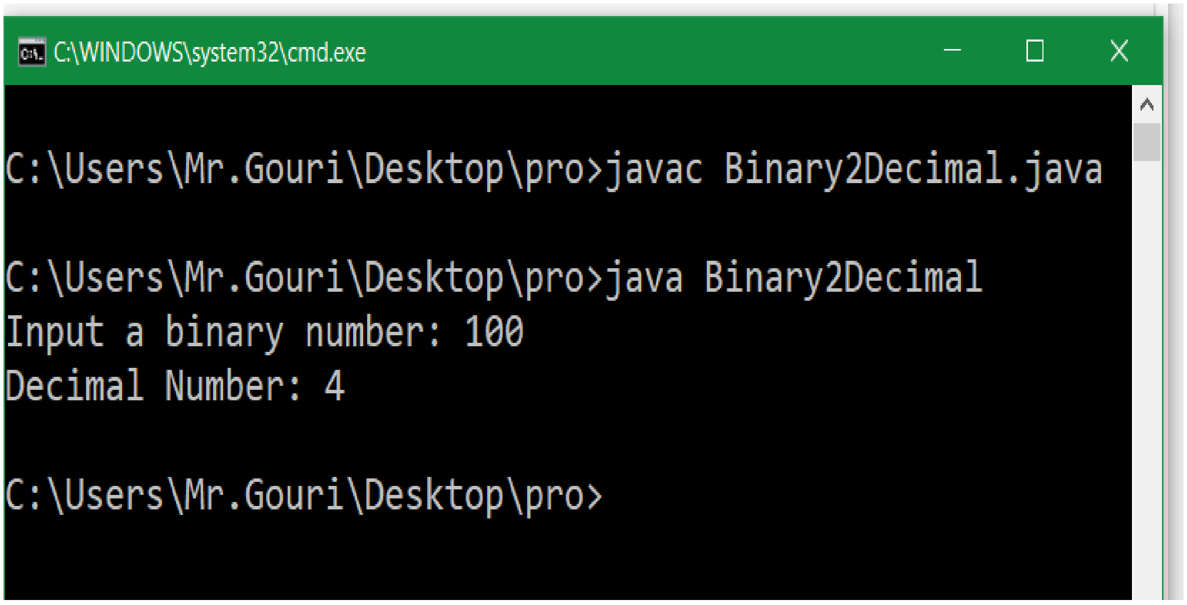
Write a Java program to convert a binary number to decimal number.

Solution:

```
import java.util.Scanner;
```

```
public class Binary2Decimal
{
      public static void main(String[] args)
      {
               Scanner sc = new Scanner(System.in);
               long binaryNumber, decimalNumber = 0, j = 1, remainder;
               System.out.print("Input a binary number: ");
               binaryNumber = sc.nextLong();

            while (binaryNumber != 0)
               {
                          remainder = binaryNumber % 10;
                          decimalNumber = decimalNumber + remainder * j;
                          j = j * 2;
                          binaryNumber = binaryNumber / 10;
               }
               System.out.println("Decimal Number: " + decimalNumber);
      }
}
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac Binary2Decimal.java

C:\Users\Mr.Gouri\Desktop\pro>java Binary2Decimal
Input a binary number: 100
Decimal Number: 4

C:\Users\Mr.Gouri\Desktop\pro>
```

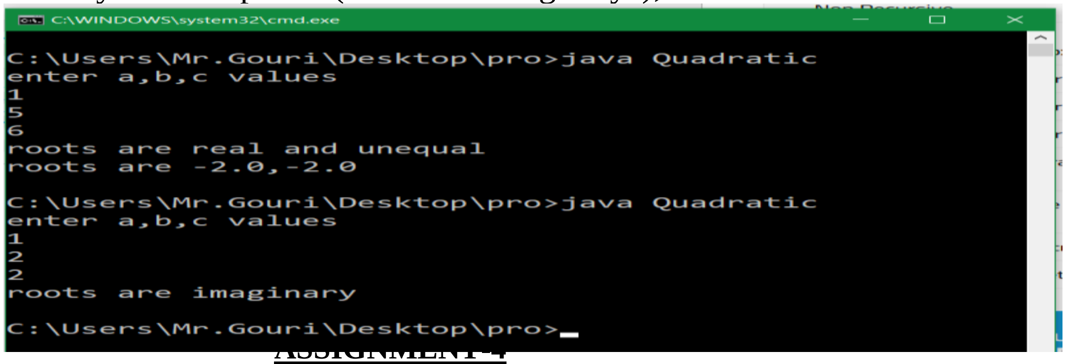
Program5:

Write a Java program that prints all real solutions to the quadratic equation ax2 + bx +c=0. Read in a, b, c and use the quadratic formula. If the discriminate b2-4ac is negative, display a message stating that there are no real solutions.(Using Buffered Reader concept)

Solution:

```
import java.io.*;
class Quadratic
{
    public static void main(String args[])throws IOException
    {
            double x1,x2,disc,a,b,c;
            InputStreamReader obj=new InputStreamReader(System.in);
            BufferedReader br=new BufferedReader(obj);
            System.out.println("enter a,b,c values");
            a=Double.parseDouble(br.readLine());
            b=Double.parseDouble(br.readLine());
            c=Double.parseDouble(br.readLine());
            disc=(b*b)-(4*a*c);
            if(disc==0)
            {
                    System.out.println("roots are real and equal ");
                    x1=x2=-b/(2*a);
                    System.out.println("roots are "+x1+","+x2);
            }
            else if(disc>0)
            {
                    System.out.println("roots are real and unequal");
                    x1=(-b+Math.sqrt(disc))/(2*a);
                    x2=(-b+Math.sqrt(disc))/(2*a);
                    System.out.println("roots are "+x1+","+x2);
            }
            else
            {
                    System.out.println("roots are imaginary");
            }
        }
    }
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>java Quadratic
enter a,b,c values
1
5
6
roots are real and unequal
roots are -2.0,-2.0

C:\Users\Mr.Gouri\Desktop\pro>java Quadratic
enter a,b,c values
1
2
2
roots are imaginary

C:\Users\Mr.Gouri\Desktop\pro>_
```

ASSIGNMENT-4

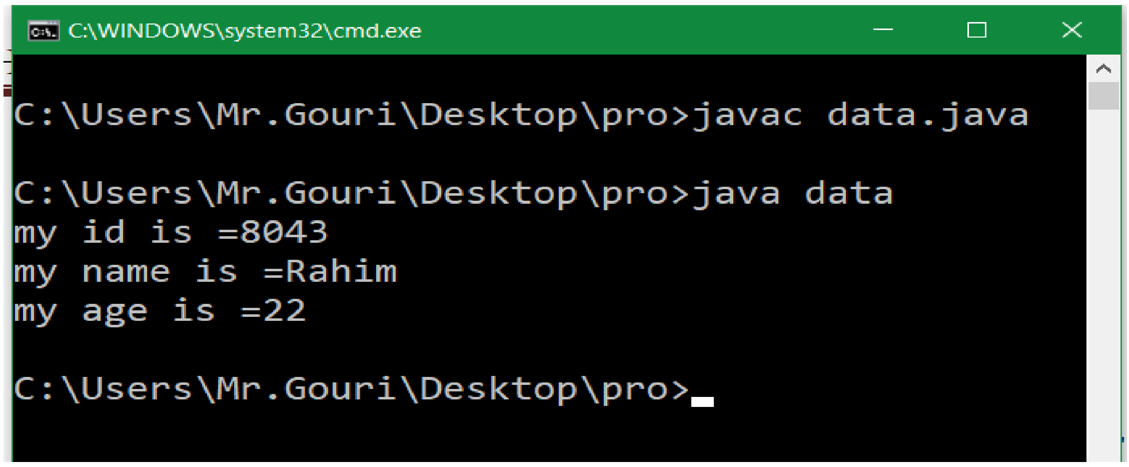
Program1:

Write a Java program using classes and object.

Solution:

```
public class data
{
    String id;
    String name;
    String age;

    public data()
    {
            id="8043";
            name="Rahim";
            age="22";
    }

    public void displaydata()
    {
            System.out.println("my id is ="+id);
            System.out.println("my name is ="+name);
            System.out.println("my age is ="+age);
    }

    public static void main(String args[])
    {
            data obj=new data();       //Create an object "obj" of class "data"
            obj.displaydata();
    }
}
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac data.java

C:\Users\Mr.Gouri\Desktop\pro>java data
my id is =8043
my name is =Rahim
my age is =22

C:\Users\Mr.Gouri\Desktop\pro>_
```

Program2:

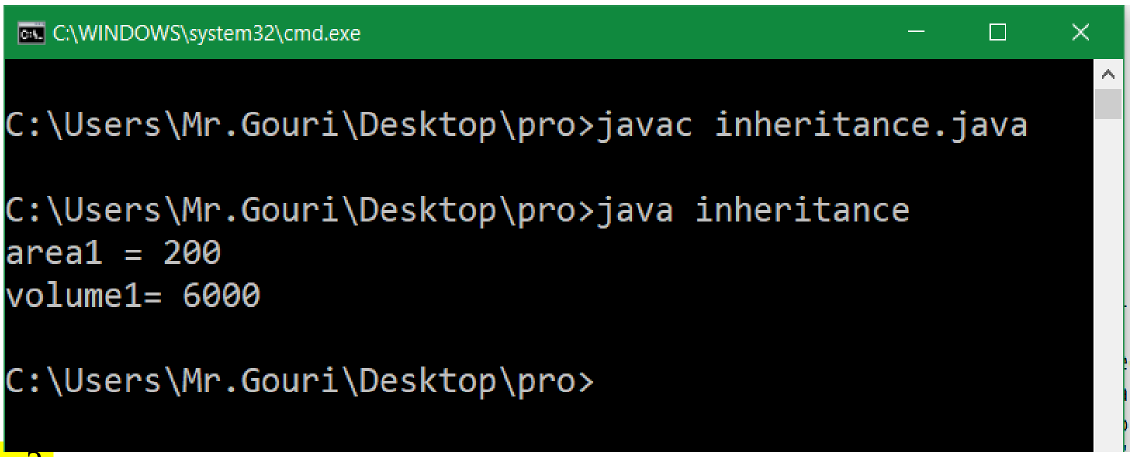
Write a Java program to implement Inheritance

Solution:

```
class room
{
    int l;
    int b;
    room(int x,int y)
    {
            l=x;
            b=y;

    }
    int area()
    {

            return(l*b);
    }
}
class bedroom extends room
{
    int h;
    bedroom(int x,int y,int z)
    {
            super(x,y);
            h=z;
    }
    int volume()
    {
            return(l*b*h);
    }
}
```

```
class inheritance
{
        public static void main(String args[])
        {
        bedroom room1=new bedroom(10,20,30);
        int area1=room1.area();
        int volume1=room1.volume();

        System.out.println("area1 = " +area1);
        System.out.puintln("volume1= "+volume1);
        }
}
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac inheritance.java

C:\Users\Mr.Gouri\Desktop\pro>java inheritance
area1 = 200
volume1= 6000

C:\Users\Mr.Gouri\Desktop\pro>
```

Program3:

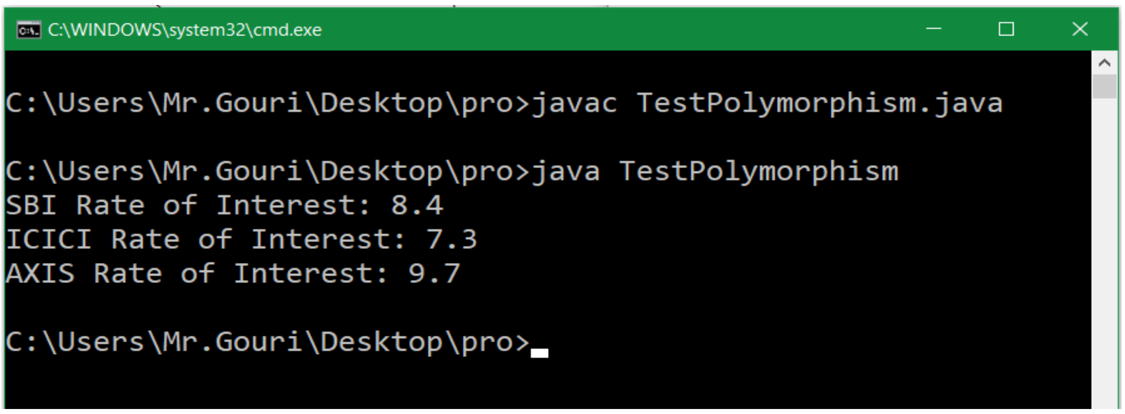
Write a Java program to implement **Polymorphism**. (Note: Consider a scenario; Bank is a class that provides method to get the rate of interest. But, rate of interest may differ according to banks. For example, SBI, ICICI and AXIS banks are providing 8.4%, 7.3% and 9.7% rate of interest.

Solution:

```
class Bank
{
   float getRateOfInterest()
   {
    return 0;
   }
}
class SBI extends Bank
{
   float getRateOfInterest()
   {
    return 8.4f;
   }
}
class ICICI extends Bank
{
   float getRateOfInterest()
   {
    return 7.3f;
   }
}
class AXIS extends Bank
{
   float getRateOfInterest()
   {
    return 9.7f;
   }
}
```

```
class TestPolymorphism
{
        public static void main(String args[])
        {
                Bank b;
                b=new SBI();
                System.out.println
    ("SBI Rate of Interest: "+b.getRateOfInterest());

                b=new ICICI();
                System.out.println
 ("ICICI Rate of Interest: "+b.getRateOfInterest());


                b=new AXIS();
                System.out.println
 ("AXIS Rate of Interest: "+b.getRateOfInterest());
          }
   }
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac TestPolymorphism.java

C:\Users\Mr.Gouri\Desktop\pro>java TestPolymorphism
SBI Rate of Interest: 8.4
ICICI Rate of Interest: 7.3
AXIS Rate of Interest: 9.7

C:\Users\Mr.Gouri\Desktop\pro>
```

Program4:

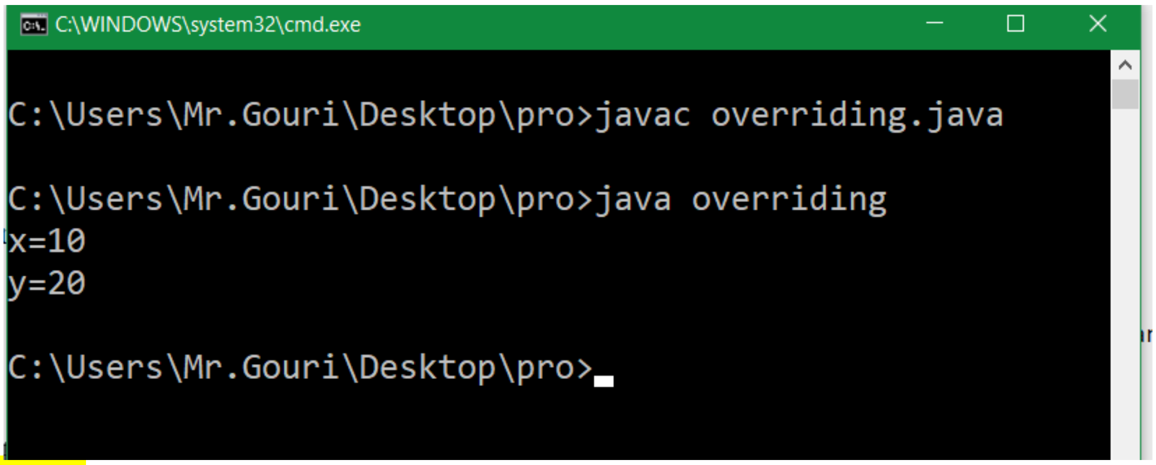
Write a Java program to implement Method Overriding

Solution:

```
class sup
{
     int x;
     sup(int x)
     {
     this.x = x;
     }
     void display()
     {
     System.out.println("x="+x);
     }
  }
class sub extends sup
{
     int y;
     sup(int x,int y)
     {
             super(x);
             this.y = y;
     }
     void display()
     {
     System.out.println("x="+x);
     System.out.println("y="+y);
     }
  }
```

```
class overriding
{
       public static void main(String args[])
       {
               sub s = new sub(10,20);
               s.display();

       }
  }
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac overriding.java

C:\Users\Mr.Gouri\Desktop\pro>java overriding
x=10
y=20

C:\Users\Mr.Gouri\Desktop\pro>
```

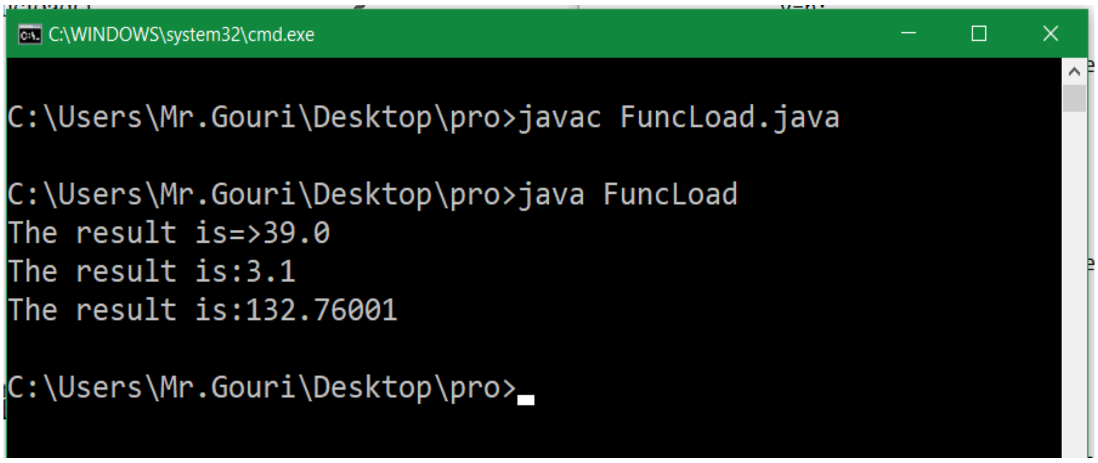
Program5:

Write a Java program to implement Method Overloading

Solution:

```
class FuncLoad
{
    public static void main(String args[])
    {
    FuncLoad obj=new FuncLoad();
    obj.add(15,24);
    obj.add(2.3f,0.8f);
    obj.add(56,76.76f);
    }
    int x,y;
    float p,q,result;
    void add(int a,int b)
    {
            x=a;
            y=b;
            result=x+y;
            System.out.println("The result is=>"+result);
    }
    void add(float a,float b)
    {
            p= a;
            q= b;
            result= p+q;
            System.out.println("The result is:"+result);
    }
void add(int a,float b)
{
        x= a;
        p= b;
        result= x+p;
        System.out.println
        ("The result is:"+result);

}
}
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac FuncLoad.java

C:\Users\Mr.Gouri\Desktop\pro>java FuncLoad
The result is=>39.0
The result is:3.1
The result is:132.76001

C:\Users\Mr.Gouri\Desktop\pro>_
```

ASSIGNMENT-5

Program1:

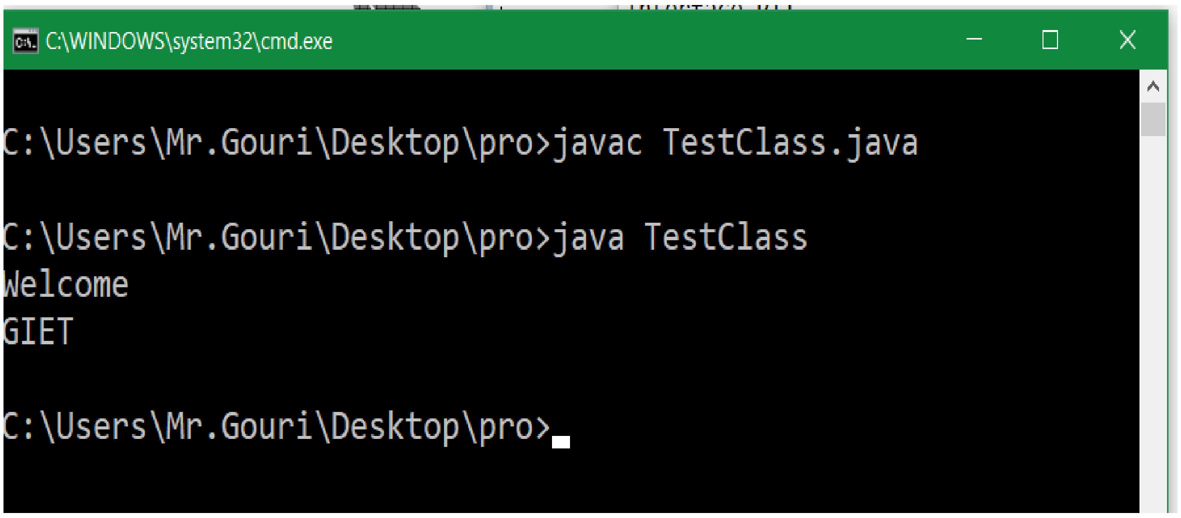
Write a Java program to demonstrate multiple inheritances through default method. (Using Interface)

Solution:

```
// A simple Java program to demonstrate multiple
// inheritance through default methods.

interface PI1
{
    // default method
    default void show()
    {
            System.out.println("Welcome");
    }
}
interface PI2
{
    // Default method

    default void show()
    {
            System.out.println("GIET");
    }
}
    // Implementation class code

class TestClass implements PI1, PI2
{
    // Overriding default show method

    public void show()
    {
            // use super keyword to call the show
            // method of PI1 interface

                    PI1.super.show();
            // use super keyword to call the show
            // method of PI2 interface

                    PI2.super.show();
    }
```

```
        public static void main(String args[])
        {
                TestClass d = new TestClass();
                d.show();
        }
}
```

Output:



C:\WINDOWS\system32\cmd.exe

```
C:\Users\Mr.Gouri\Desktop\pro>javac TestClass.java

C:\Users\Mr.Gouri\Desktop\pro>java TestClass
Welcome
GIET

C:\Users\Mr.Gouri\Desktop\pro>_
```

Program2:

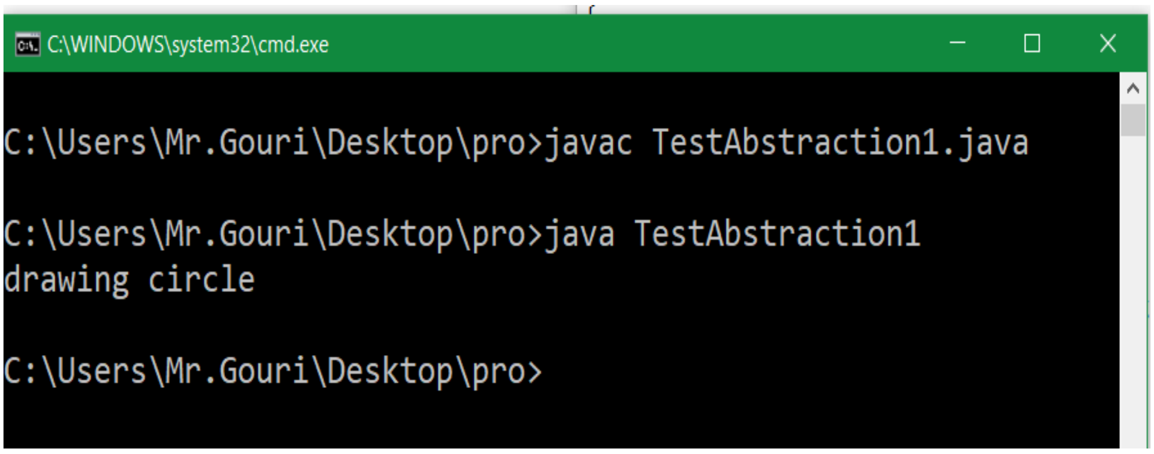
Define **Abstract class**. In this program, Shape is the abstract class; its implementation is provided by the Rectangle and Circle classes. Mostly, we don't know about the implementation class (i.e. hidden to the end user) and object of the implementation class is returns the instance of the class (i,e. Factory Method).

Solution:

```
abstract class Shape
{
   abstract void draw ();
}

//In real scenario, implementation is provided by others i.e. unknown by end user

class Rectangle extends Shape
{
   void draw ()
```

```
    {
     System.out.println ("drawing rectangle");
    }
}
class Circle1 extends Shape
{
    void draw ()
    {
     System.out.println ("drawing circle");
    }
}

//In real scenario, method is called by programmer or user

class TestAbstraction1
{
    public static void main(String args[])
    {
        Shape s=new Circle1 (); //In real scenario, object is provided through
                        method e.g. getShape () method
        s.draw ();
    }
}
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac TestAbstraction1.java

C:\Users\Mr.Gouri\Desktop\pro>java TestAbstraction1
drawing circle

C:\Users\Mr.Gouri\Desktop\pro>
```

Program3: Wrapper Classes in Java

A Wrapper class is a class whose object wraps or contains a primitive data types. When we create an object to a wrapper class, it contains a field and in this field, we can store a primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

Need of Wrapper Classes

1. They convert primitive data types into objects. Objects are needed if we wish to modify the arguments passed into a method (because primitive types are passed by value).
2. The classes in java.util package handle only objects and hence wrapper classes help in this case also.
3. Data structures in the Collection framework, such as Array List and Vector, store only objects (reference types) and not primitive types.
4. An object is needed to support synchronization in multithreading.

Primitive Data types and their Corresponding Wrapper class

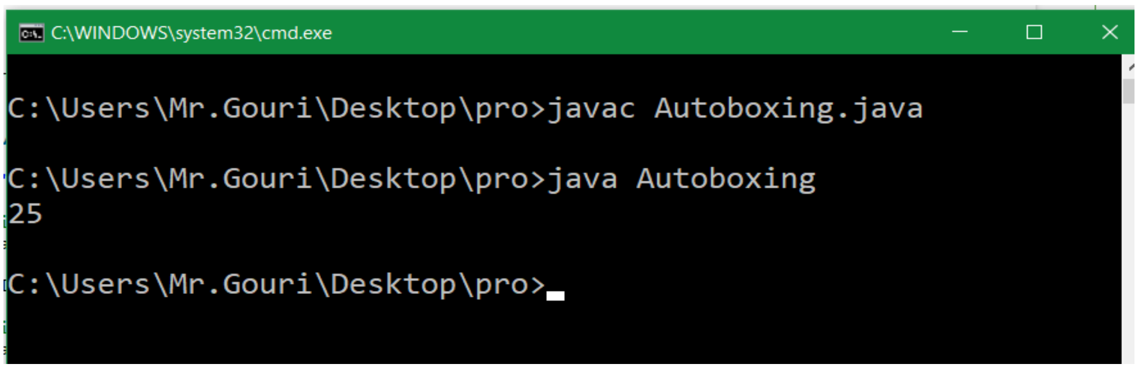
Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
long	Integer
float	Float
double	Double
boolean	Boolean

Autoboxing: Automatic conversion of primitive types to the object of their corresponding wrapper classes is known as autoboxing. For example – conversion of int to Integer, long to Long, double to Double etc.
Example:

```
// Java program to demonstrate Autoboxing

import java.util.ArrayList;

class Autoboxing
{
   public static void main(String[] args)
   {
      char ch = 'a';;

      // Autoboxing- primitive to Character object conversion

       Character a = ch;
       ArrayList<Integer> arrayList = new ArrayList<Integer>();
```

```
        // Autoboxing because ArrayList stores only objects

         arrayList.add(25);

        // printing the values from object

         System.out.println(arrayList.get(0));
    }
}
```

Output:

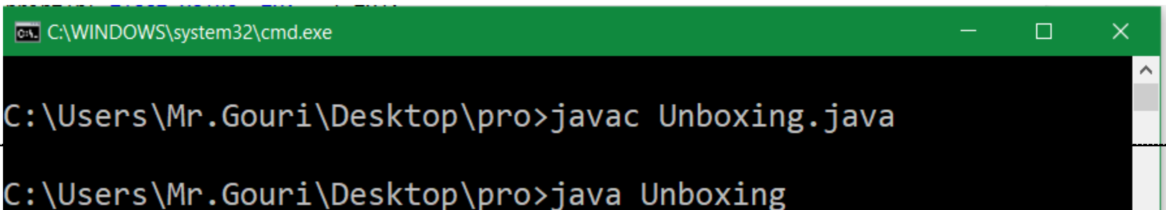


```
C:\Users\Mr.Gouri\Desktop\pro>javac Autoboxing.java

C:\Users\Mr.Gouri\Desktop\pro>java Autoboxing
25

C:\Users\Mr.Gouri\Desktop\pro>_
```

Unboxing: It is just the reverse process of autoboxing. Automatically converting an object of a wrapper class to its corresponding primitive type is known as unboxing. For example – conversion of Integer to int, Long to long, Double to double etc.

// Java program to demonstrate Unboxing

```
import java.util.ArrayList;
class Unboxing
{
    public static void main(String[] args)
    {
        Character ch = 'a';

        // unboxing - Character object to primitive conversion
        char a = ch;

        ArrayList<Integer> arrayList = new ArrayList<Integer>();
        arrayList.add(24);

        // unboxing because get method returns an Integer object
        int num = arrayList.get(0);

        // printing the values from primitive data types
        System.out.println(num);
    }
```



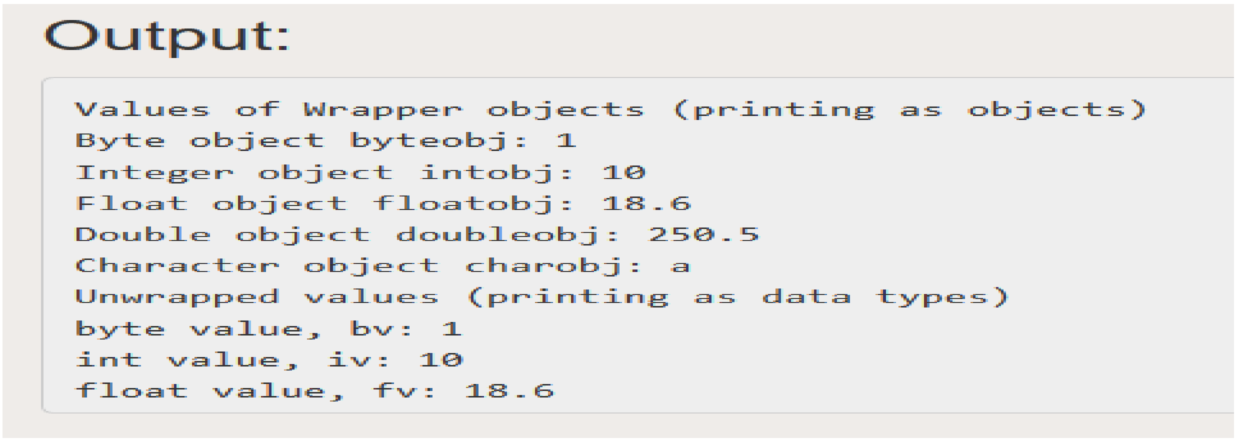
```
C:\Users\Mr.Gouri\Desktop\pro>javac Unboxing.java

C:\Users\Mr.Gouri\Desktop\pro>java Unboxing
```

}

Output:

Implementation :

```
// Java program to demonstrate Wrapping and UnWrapping

class WrappingUnwrapping
{
    public static void main(String args[])
    {
        //  byte data type
        byte a = 1;

        // wrapping around Byte object
        Byte byteobj = new Byte(a);

        // int data type
        int b = 10;

        //wrapping around Integer object
        Integer intobj = new Integer(b);

        // float data type
        float c = 18.6f;

        // wrapping around Float object
        Float floatobj = new Float(c);

        // double data type
        double d = 250.5;

        // Wrapping around Double object
        Double doubleobj = new Double(d);

        // char data type
        char e='a';

        // wrapping around Character object
        Character charobj=e;
        //  printing the values from objects
        System.out.println("Values of Wrapper objects (printing as objects)");
        System.out.println("Byte object byteobj:  " + byteobj);
        System.out.println("Integer object intobj:  " + intobj);
        System.out.println("Float object floatobj:  " + floatobj);
        System.out.println("Double object doubleobj:  " + doubleobj);
        System.out.println("Character object charobj:  " + charobj);

        // objects to data types (retrieving data types from objects)
        // unwrapping objects to primitive data types
        byte bv = byteobj;
        int iv = intobj;
        float fv = floatobj;
        double dv = doubleobj;
        char cv = charobj;

        // printing the values from data types
        System.out.println("Unwrapped values (printing as data types)");
        System.out.println("byte value, bv: " + bv);
        System.out.println("int value, iv: " + iv);
        System.out.println("float value, fv: " + fv);
        System.out.println("double value, dv: " + dv);
        System.out.println("char value, cv: " + cv);
    }
}
```


Output:

```
Values of Wrapper objects (printing as objects)
Byte object byteobj: 1
Integer object intobj: 10
Float object floatobj: 18.6
Double object doubleobj: 250.5
Character object charobj: a
Unwrapped values (printing as data types)
byte value, bv: 1
int value, iv: 10
float value, fv: 18.6
```

ASSIGNMENT-6

Program1: (**Introduction of Package**)

In Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. Packages are used for:

- Preventing naming conflicts. For example there can be two classes with name Employee in two packages, college.staff.cse.Employee and college.staff.ee.Employee
- Making searching/locating and usage of classes, interfaces, enumerations and annotations easier
- Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.
- Packages can be considered as data encapsulation (or data-hiding).

All we need to do is put related classes into packages. After that we can simply write a import a class from existing packages and use it in our program. A packages is container of group of related classes where some of the classes are accessible are exposed and others are kept for internal purpose.
We can reuse existing classes from the packages as many time as we need it in our program.

How packages work?

Package names and directory structure are closely related. For example if a package name is *college.staff.cse*, then there are three directories, *college*, *staff* and *cse* such that *cse* is present in *staff* and *staff* is present *college*. Also, the directory *college* is accessible through CLASSPATH variable, i.e., path of parent directory of college is present in CLASSPATH. The idea is to make sure that classes are easy to locate.
Package naming conventions : Packages are named in reverse order of domain names, i.e., org.geeksforgeeks.practice. For example, in a college, the recommended convention is college.tech.cse, college.tech.ee, college.art.history, etc.

Adding a class to a Package : We can add more classes to an created package by using package name at the top of the program and saving it in the package directory. We need a new **java** file to define a public class, otherwise we can add the new class to an existing **.java** file and recompile it.

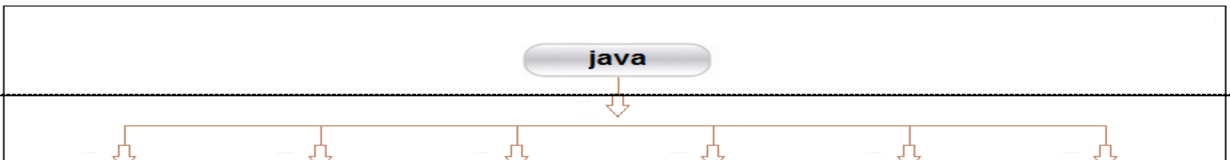
Subpackages: Packages that are inside another package are the **subpackages**. These are not imported by default, they have to imported explicitly. Also, members of a subpackage have no access privileges, i.e., they are considered as different package for protected and default access specifiers.
Example :

```
import java.util.*;
```

Note: util **is a subpackage created inside** java **package.**

Java-API Packages

Java APl(Application Program Interface) provides a large numbers of classes grouped into different packages according to functionality. Most of the time we use the packages available



with the the Java API. Following figure shows the system packages that are frequently used in the programs.

Java System Packages and Their Classes

java.lang	Language support classes. They include classes for primitive types, string, math functions, thread and exceptions.
java.util	Language utility classes such as vectors, hash tables, random numbers, data, etc.
java.io	Input/output support classes. They provide facilities for the input and output of data.
java.applet	Classes for creating and implementing applets.
java.net	Classes for networking. They include classes for communicating with local computers as well as with internet servers.
java.awt	Set of classes for implementing graphical user interface. They include classes for windows, buttons, lists, menus and so on.

Program2:

Simple example of java package

The **package keyword** is used to create a package in java.

```
//save as Simple.java
package mypack;
public class Simple
{
   public static void main(String args[])
  {
    System.out.println("Welcome to package");
   }
 }
```

```
Output:Welcome to package
```

How to compile java package

If you are not using any IDE, you need to follow the syntax given below:

1. **javac -d directory javafilename**

For **example**

1. javac -d . Simple.java

The -d switch specifies the destination where to put the generated class file. You can use any directory name like /home (in case of Linux), d:/abc (in case of windows) etc. If you want to keep the package within the same directory, you can use . (dot).

How to run java package program

You need to use fully qualified name e.g. mypack.Simple etc to run the class.

To Compile: javac -d . Simple.java

To Run: java mypack.Simple

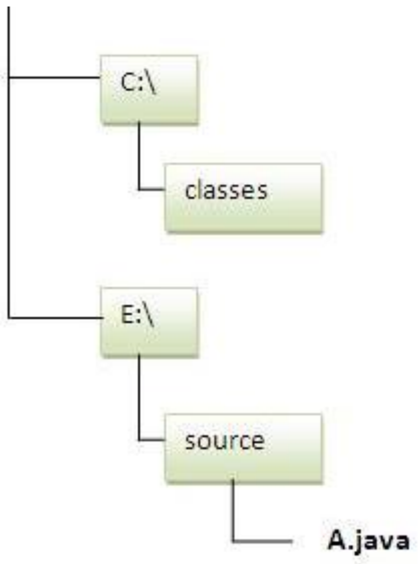
```
Output: Welcome to package
```

The -d is a switch that tells the compiler where to put the class file i.e. it represents destination. The Represents the current folder.

How to send the class file to another directory or drive?

There is a scenario; I want to put the class file of A.java source file in classes folder of c: drive. For example:

```
//save as Simple.java
package mypack;
public class Simple{
 public static void main(String args[]){
    System.out.println("Welcome to package");
   }
}
```



To Compile:

e:\sources> javac -d c:\classes Simple.java

To Run:

To run this program from e:\source directory, you need to set classpath of the directory where the class file resides.

e:\sources> set classpath=c:\classes;.;

e:\sources> java mypack.Simple

Another way to run this program by -classpath switch of java:

The -classpath switch can be used with javac and java tool.

To run this program from e:\source directory, you can use -classpath switch of java that tells where to look for class file. For example:

e:\sources> java -classpath c:\classes mypack.Simple

```
Output:Welcome to package
```

Note: Ways to load the class files or jar files

There are two ways to load the class files temporary and permanent.

- Temporary
  - By setting the classpath in the command prompt
  - By -classpath switch
- Permanent
  - By setting the classpath in the environment variables
  - By creating the jar file, that contains all the class files, and copying the jar file in the jre/lib/ext folder.

Rule: There can be only one public class in a java source file and it must be saved by the public class name.

1. //save as C.java otherwise Compile Time Error
2. class A{}
3. class B{}
4. public class C{}

How to access package from another package?

There are three ways to access the package from outside the package.

1. import package.*;
2. import package.classname;
3. fully qualified name.

*1) Using packagename.**

If you use package.* then all the classes and interfaces of this package will be accessible but not subpackages.

The import keyword is used to make the classes and interface of another package accessible to the current package.

Example of package that import the packagename.*

```
//save by A.java
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
```

```
//save by B.java
package mypack;
import pack.*;

class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

```
Output:Hello
```

---

2) Using packagename.classname

If you import package.classname then only declared class of this package will be accessible.

Example of package by import package.classname

```
//save by A.java

package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
```

```
//save by B.java
package mypack;
import pack.A;

class B{
  public static void main(String args[]){
   A obj = new A();
```

```
    obj.msg();
  }
}
```

```
Output:Hello
```

3) Using fully qualified name

If you use fully qualified name then only declared class of this package will be accessible. Now there is no need to import. But you need to use fully qualified name every time when you are accessing the class or interface.

It is generally used when two packages have same class name e.g. java.util and java.sql packages contain Date class.

Example of package by import fully qualified name

```
//save by A.java
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
```

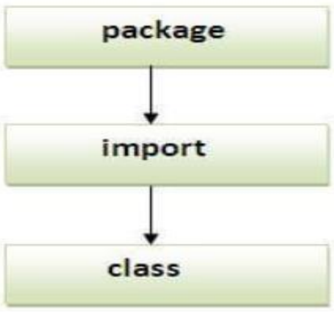
```
//save by B.java
package mypack;
class B{
  public static void main(String args[]){
   pack.A obj = new pack.A();//using fully qualified name
   obj.msg();
  }
}
```

```
Output:Hello
```

Note: If you import a package, subpackages will not be imported.

If you import a package, all the classes and interface of that package will be imported excluding the classes and interfaces of the subpackages. Hence, you need to import the subpackage as well.

Note: Sequence of the program must be package then import then class.



Program3:

Write a Java program to demonstrate Accessing of members when a corresponding class is imported and not imported.

Solution:

```
import java.util.Vector;

public class ImportDemo
{
  public ImportDemo()
  {
    // java.util.Vector is imported, hence we are
    // able to access directly in our code.
    Vector newVector = new Vector();

    // java.util.ArrayList is not imported, hence
    // we were referring to it using the complete
    // package.
    java.util.ArrayList newList = new java.util.ArrayList();
  }

  public static void main(String arg[])
  {
    new ImportDemo();
  }
}
```

Program4:

Java program to Illustration of user-defined packages:

Solution:

Creating our first package:
File name – ClassOne.java

```
package package_name;

public class ClassOne {
    public void methodClassOne() {
        System.out.println("Hello there its ClassOne");
    }
}
```

Creating our second package:
File name – ClassTwo.java

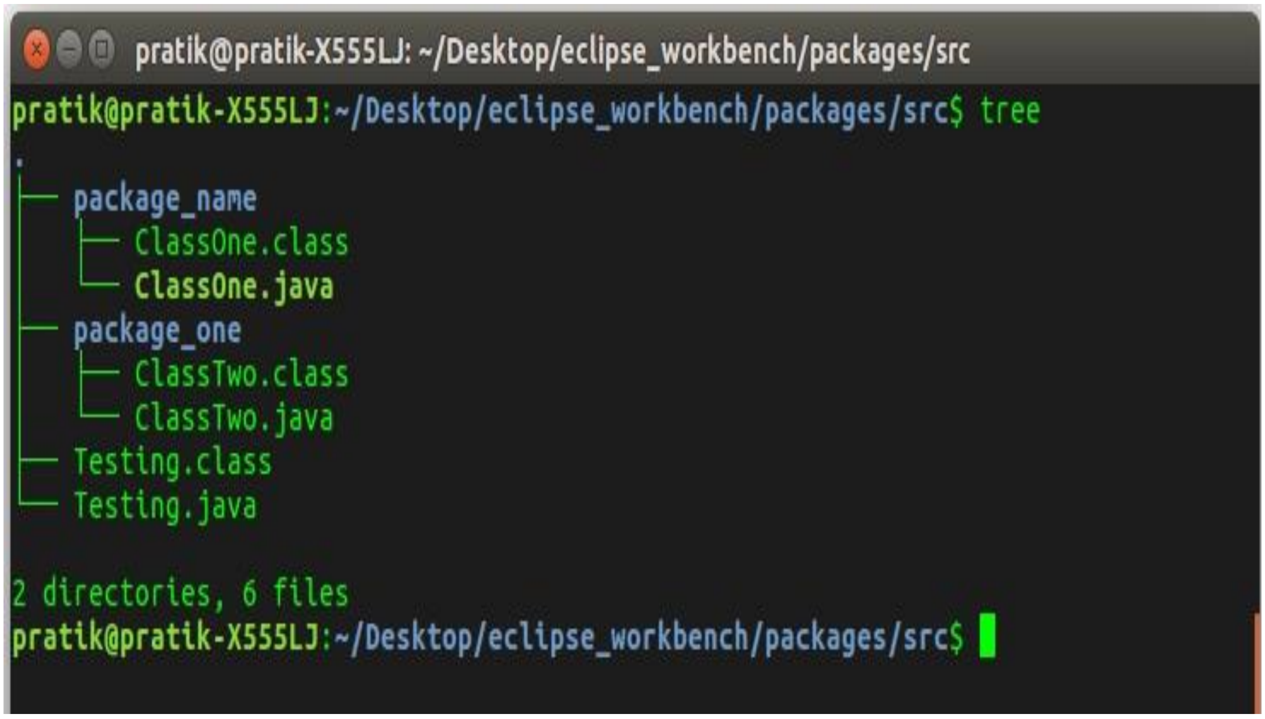
```
package package_one;

public class ClassTwo {
    public void methodClassTwo(){
        System.out.println("Hello there i am ClassTwo");
    }
}
```

Making use of both the created packages:
File name – Testing.java

```
import package_one.ClassTwo;
import package_name.ClassOne;

public class Testing {
    public static void main(String[] args){
        ClassTwo a = new ClassTwo();
        ClassOne b = new ClassOne();
        a.methodClassTwo();
        b.methodClassOne();
    }
}
```

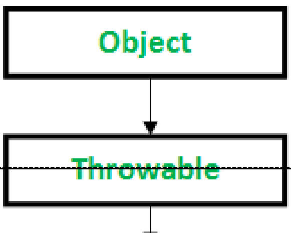
Output:

```
Hello there i am ClassTwo
Hello there its ClassOne
```

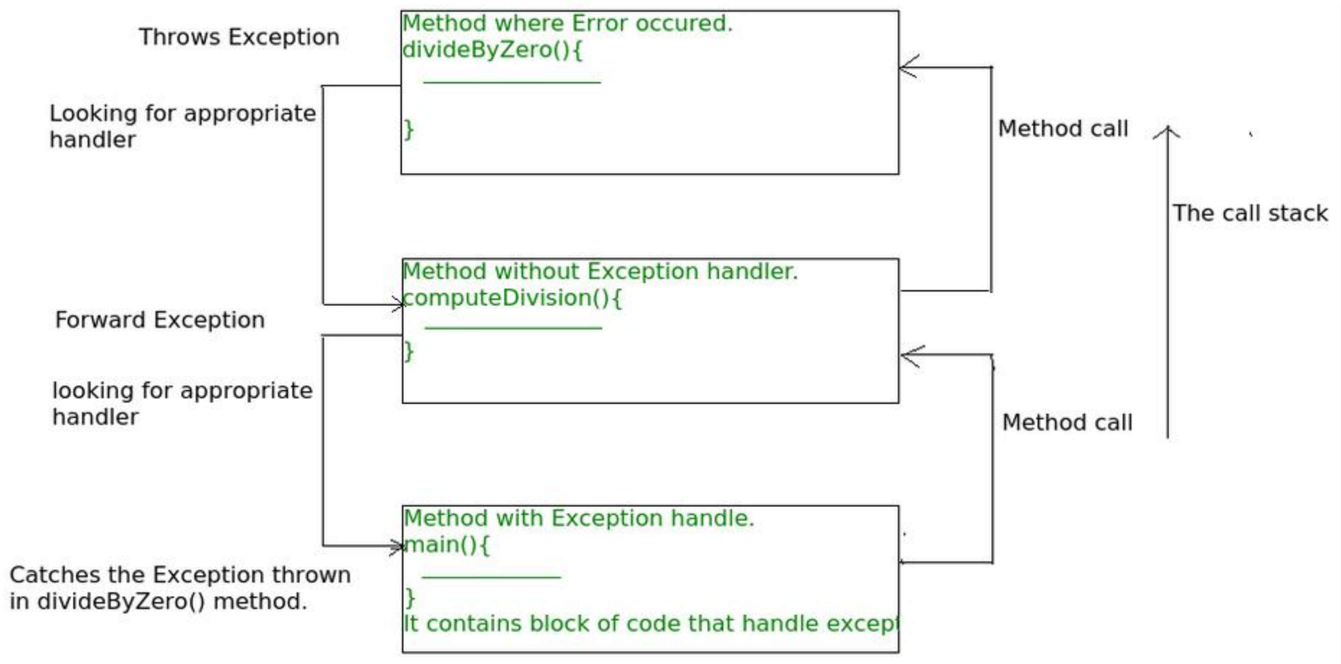

pratik@pratik-X555LJ: ~/Desktop/eclipse_workbench/packages/src

```
pratik@pratik-X555LJ:~/Desktop/eclipse_workbench/packages/src$ tree
.
├── package_name
│   ├── ClassOne.class
│   └── ClassOne.java
├── package_one
│   ├── ClassTwo.class
│   └── ClassTwo.java
├── Testing.class
└── Testing.java

2 directories, 6 files
pratik@pratik-X555LJ:~/Desktop/eclipse_workbench/packages/src$
```

Program5 **:(Exception Handling)**



See the below diagram to understand the flow of the call stack.


Throws Exception
Method where Error occured.
divideByZero(){
}
Looking for appropriate handler
Method call
The call stack
Method without Exception handler.
computeDivision(){
}
Forward Exception
looking for appropriate handler
Method call
Method with Exception handle.
main(){
}
It contains block of code that handle excep
Catches the Exception thrown in divideByZero() method.

The call stack and searching the call stack for exception handler.

Program:

Write a Java program to demonstrate exception is thrown

```
// how the runTime system searches th call stack
// to find appropriate exception handler.
```

Solution:

```
class ExceptionThrown
{
   // It throws the Exception(ArithmeticException).
   // Appropriate Exception handler is not found within this method.
   static int divideByZero(int a, int b)
  {

      // this statement will cause ArithmeticException(/ by zero)
      int i = a/b;

      return i;
   }

   // The runTime System searches the appropriate Exception handler
   // in this method also but couldn't have found. So looking forward
   // on the call stack.
   static int computeDivision(int a, int b)
   {

      int res =0;

      try
      {
        res = divideByZero(a,b);
      }
      // doesn't matches with ArithmeticException
      catch(NumberFormatException ex)
      {
         System.out.println("NumberFormatException is occured");
      }
      return res;
   }

   // In this method found appropriate Exception handler.
   // i.e. matching catch block.



  public static void main(String args[])
    {
      int a = 1;
      int b = 0;

      try
      {
         int i = computeDivision(a,b);

      }

      // matching ArithmeticException
      catch(ArithmeticException ex)
      {
         // getMessage will print description of exception(here / by zero)
         System.out.println(ex.getMessage());
      }
    }
}
```

Output:

```
/ By zero
```

ASSIGNMENT-7

Program1: (Main thread in Java)

Java provides built-in support for multithreaded programming. A multi-threaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each thread defines a separate path of execution.

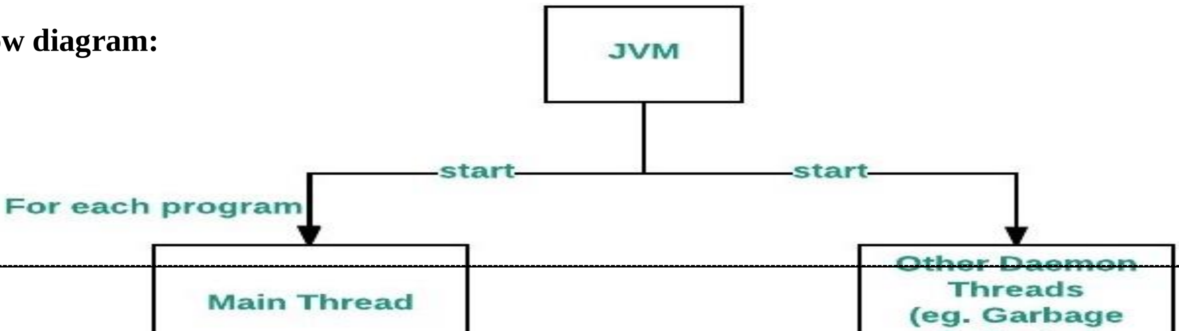
Main Thread

When a Java program starts up, one thread begins running immediately. This is usually called the *main* thread of our program, because it is the one that is executed when our program begins.

Properties:

- It is the thread from which other "child" threads will be spawned.
- Often, it must be the last thread to finish execution because it performs various shutdown actions

Flow diagram:



How to control Main thread

The main thread is created automatically when our program is started. To control it we must obtain a reference to it. This can be done by calling the method *current Thread ( )* which is present in Thread class. This method returns a reference to the thread on which it is called. The default priority of Main thread is 5 and for all remaining user threads priority will be inherited from parent to child.

Program:

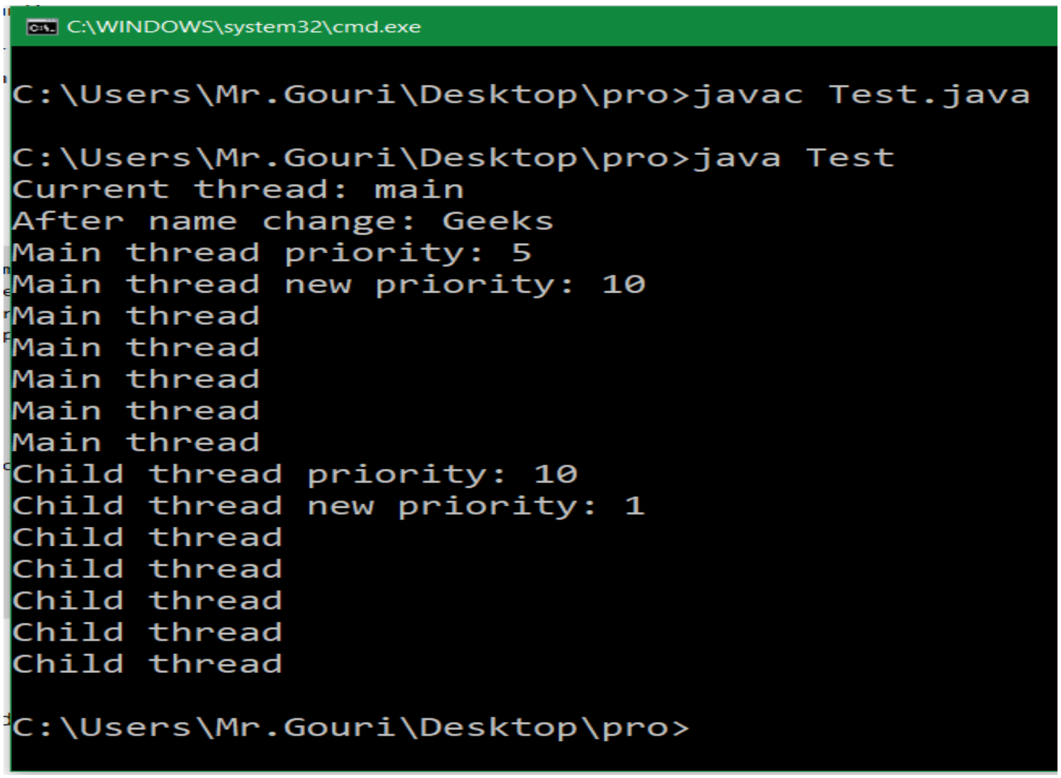
Write a Java program to control the Main Thread

Solution:

```
public class Test extends Thread
{
   public static void main(String[] args)
   {
      // getting reference to Main thread
      Thread t = Thread.currentThread();

      // getting name of Main thread
      System.out.println("Current thread: " + t.getName());

      // changing the name of Main thread
      t.setName("Geeks");
      System.out.println("After name change: " + t.getName());
```

```
        // getting priority of Main thread
        System.out.println("Main thread priority: "+ t.getPriority());

        // setting priority of Main thread to MAX(10)
        t.setPriority(MAX_PRIORITY);

        System.out.println("Main thread new priority: "+ t.getPriority());


        for (int i = 0; i < 5; i++)
        {
            System.out.println("Main thread");
        }

        // Main thread creating a child thread
        ChildThread ct = new ChildThread();

        // getting priority of child thread
        // which will be inherited from Main thread
        // as it is created by Main thread
        System.out.println("Child thread priority: "+ ct.getPriority());

        // setting priority of Main thread to MIN(1)
        ct.setPriority(MIN_PRIORITY);

        System.out.println("Child thread new priority: "+ ct.getPriority());

        // starting child thread
        ct.start();
    }
}

// Child Thread class
class ChildThread extends Thread
{
    @Override
    public void run()
    {
        for (int i = 0; i < 5; i++)
        {
            System.out.println("Child thread");
        }
    }
}
```

Output:



C:\WINDOWS\system32\cmd.exe

```
C:\Users\Mr.Gouri\Desktop\pro>javac Test.java

C:\Users\Mr.Gouri\Desktop\pro>java Test
Current thread: main
After name change: Geeks
Main thread priority: 5
Main thread new priority: 10
Main thread
Main thread
Main thread
Main thread
Main thread
Child thread priority: 10
Child thread new priority: 1
Child thread
Child thread
Child thread
Child thread
Child thread

C:\Users\Mr.Gouri\Desktop\pro>
```

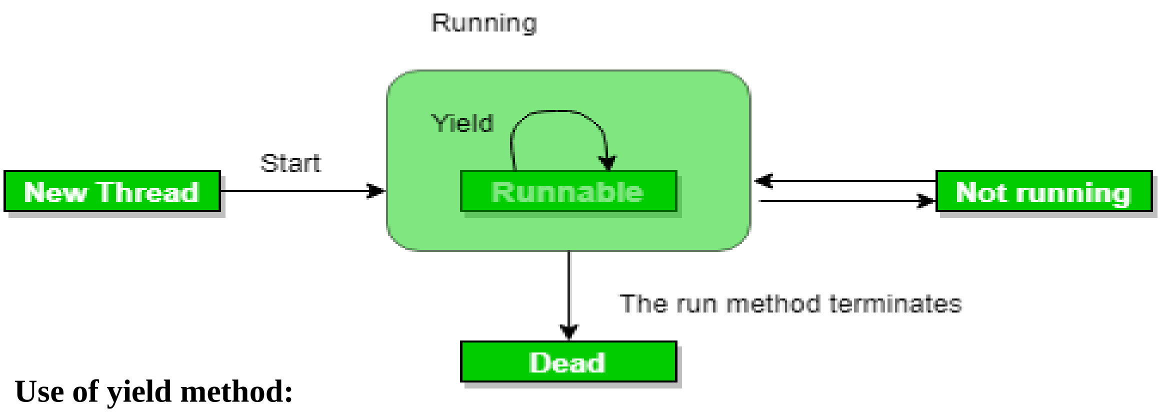
Note: Relation between the main () method and main thread in Java

For each program, a Main thread is created by JVM (Java Virtual Machine). The “Main” thread first verifies the existence of the main () method, and then it initializes the class. Note that from JDK 6, main () method is mandatory in a standalone java application.

Program2: (Java Concurrency – yield (), sleep () and join () methods)

We can prevent the execution of a thread by using one of the following methods of Thread class.

yield(): Suppose there are three threads t1, t2, and t3. Thread t1 gets the processor and starts its execution and thread t2 and t3 are in Ready/Runnable state. Completion time for thread t1 is 5 hour and completion time for t2 is 5 minutes. Since t1 will complete its execution after 5 hours, t2 has to wait for 5 hours to just finish 5 minutes job. In such scenarios where one thread is taking too much time to complete its execution, we need a way to prevent execution of a thread in between if something important is pending. yeild() helps us in doing so.
yield () basically means that the thread is not doing anything particularly important and if any other threads or processes need to be run, they should run. Otherwise, the current thread will continue to run.



Use of yield method:

- Whenever a thread calls java.lang.Thread.yield method, it gives hint to the thread scheduler that it is ready to pause its execution. Thread scheduler is free to ignore this hint.
- If any thread executes yield method, thread scheduler checks if there is any thread with same or high priority than this thread. If processor finds any thread with higher or same priority then it will move the current thread to Ready/Runnable state and give processor to other thread and if not – current thread will keep executing.

Syntax: public static native void yield()

Program:

Write a Java program to illustrate yield() method

Solution:

```
import java.lang.*;

// MyThread extending Thread

class MyThread extends Thread
{
   public void run()
   {
      for (int i=0; i<5 ; i++)
      System.out.println(Thread.currentThread().getName() + " in control");
   }
}

// Driver Class

public class yieldDemo
{
   public static void main(String[]args)
   {
      MyThread t = new MyThread();
      t.start();

      for (int i=0; i<5; i++)
      {
         // Control passes to child thread
         Thread.yield();

         // After execution of child Thread
         // main thread takes over
         System.out.println(Thread.currentThread().getName()
                        + " in control");
      }
   }
}
```

Output:

```
Thread-0 in control
Thread-0 in control
Thread-0 in control
Thread-0 in control
```

```
Thread-0 in control
main in control
main in control
main in control
main in control
main in control
```

Output may vary in machine to machine but chances of execution of yield() thread first is higher than the other thread because main thread is always pausing its execution and giving chance to child thread(with same priority).

Note:

- Once a thread has executed yield method and there are many threads with same priority is waiting for processor, then we can't specify which thread will get execution chance first.
- The thread which executes the yield method will enter in the Runnable state from running state.
- Once a thread pauses its execution, we can't specify when it will get chance again it depends on thread scheduler.
- Underlying platform must provide support for preemptive scheduling if we are using yield method.

sleep(): This method causes the currently executing thread to sleep for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.

Syntax:

```
//  sleep for the specified number of milliseconds
public static void sleep(long millis) throws InterruptedException

//sleep for the specified number of milliseconds plus nano seconds
public static void sleep(long millis, int nanos)throws InterruptedException
```

Program:

Write a Java program to illustrate sleep () method in Java

Solution:

```
import java.lang.*;
public class SleepDemo implements Runnable
{
   Thread t;
   public void run()
   {
      for (int i = 0; i < 4; i++)
      {
         System.out.println(Thread.currentThread().getName() + "  " + i);
```

```
            try
            {
               // thread to sleep for 1000 milliseconds
               Thread.sleep(1000);
            }

            catch (Exception e)
            {
               System.out.println(e);
            }
         }
      }

      public static void main(String[] args) throws Exception
      {
         Thread t = new Thread(new SleepDemo());

         // call run() function
         t.start();

         Thread t2 = new Thread(new SleepDemo());

         // call run() function
         t2.start();
      }
   }
```

Output:

```
Thread-0  0
Thread-1  0
Thread-0  1
Thread-1  1
Thread-0  2
Thread-1  2
Thread-0  3
Thread-1  3
```

Note:

- Based on the requirement we can make a thread to be in sleeping state for a specified period of time

- Sleep() causes the thread to definitely stop executing for a given amount of time; if no other thread or process needs to be run, the CPU will be idle (and probably enter a power saving mode).

yield() vs sleep()

yield :() indicates that the thread is not doing anything particularly important and if any other threads or processes need to be run, they can. Otherwise, the **current thread will continue to run.**

sleep(): causes the thread to definitely stop executing for a given amount of time; if no other thread or process needs to be run, **the CPU will be idle** (and probably enter a power saving mode).

join(): The join() method of a Thread instance is used to join the start of a thread's execution to end of other thread's execution such that a thread does not start running until another thread ends. If join() is called on a Thread instance, the currently running thread will block until the Thread instance has finished executing.
The join() method waits at most this much milliseconds for this thread to die. A timeout of 0 means to wait forever
Syntax:

```
// waits for this thread to die.
public final void join() throws InterruptedException

// waits at most this much milliseconds for this thread to die
public final void join(long millis) throws InterruptedException

// waits at most milliseconds plus nanoseconds for this thread to die.
The java.lang.Thread.join(long millis, int nanos)
```

Program:

Write a Java program to illustrate join () method in Java

Solution:

```
import java.lang.*;

public class JoinDemo implements Runnable
{
   public void run()
   {
      Thread t = Thread.currentThread();
      System.out.println("Current thread: " + t.getName());
```

```
        // checks if current thread is alive
        System.out.println("Is Alive? " + t.isAlive());
    }

    public static void main(String args[]) throws Exception
    {
        Thread t = new Thread(new JoinDemo());
        t.start();

        // Waits for 1000ms this thread to die.
        t.join(1000);

        System.out.println("\nJoining after 1000"+" mili seconds: \n");
        System.out.println("Current thread: " + t.getName());


        // Checks if this thread is alive
        System.out.println("Is alive? " + t.isAlive());
    }
}
```

Output:

```
Current thread: Thread-0
Is Alive? true

Joining after 1000 mili seconds:

Current thread: Thread-0
Is alive? false
```

Note:

- If any executing thread t1 calls join() on t2 i.e; t2.join() immediately t1 will enter into waiting state until t2 completes its execution.
- Giving a timeout within join(), will make the join() effect to be nullified after the specific timeout.

Program3:

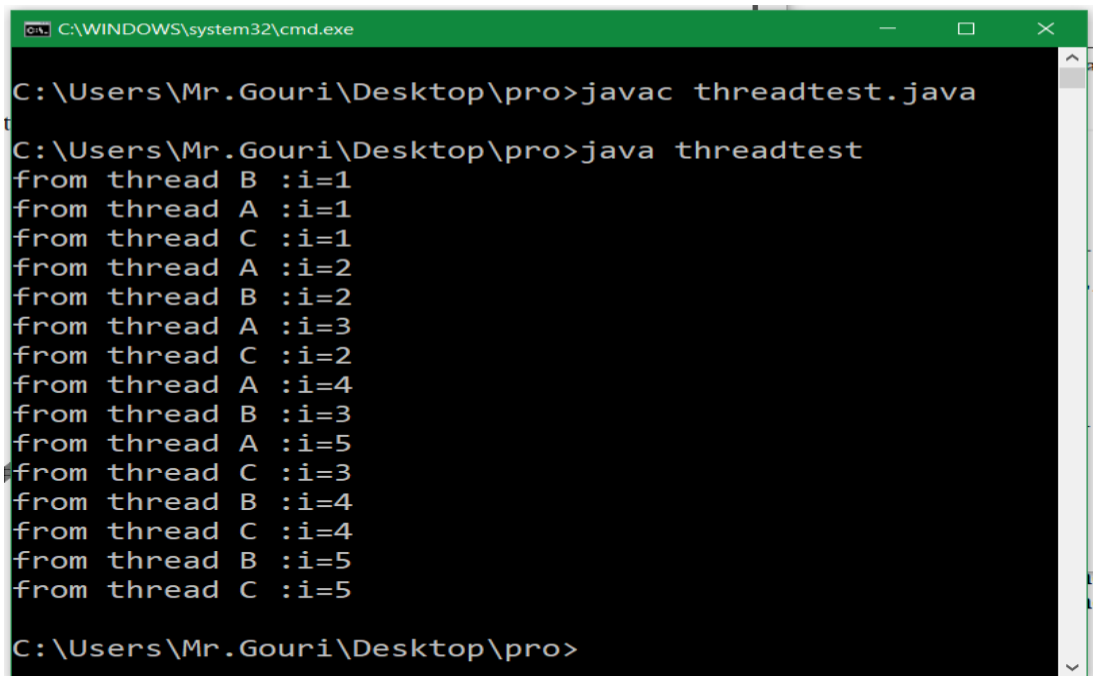
Write a Java program to implement Multithreading

Solution:

```
class A extends Thread
{
      public void run()
      {
          for(int i=1;i<=5;i++)
         {
          System.out.println
          ("from thread A :i="+i);
          }
      }
}
class B extends Thread
      public void run()
      {
         for(int i=1;i<=5;i++)
         {
          System.out.println
          ("from thread B :i="+i);
          }
      }
}
```

```
class C extends Thread
{
   public void run()
  {
      for(int i=1;i<=5;i++)
      {
          System.out.println("from thread C :i="+i);

       }
    }
}
 class threadtest
 {
      public static void main(String args[])
      {
               new A().start();
               new B().start();
               new C().start();
       }
    }
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac threadtest.java

C:\Users\Mr.Gouri\Desktop\pro>java threadtest
from thread B :i=1
from thread A :i=1
from thread C :i=1
from thread A :i=2
from thread B :i=2
from thread A :i=3
from thread C :i=2
from thread A :i=4
from thread B :i=3
from thread A :i=5
from thread C :i=3
from thread B :i=4
from thread C :i=4
from thread B :i=5
from thread C :i=5

C:\Users\Mr.Gouri\Desktop\pro>
```

Program4:

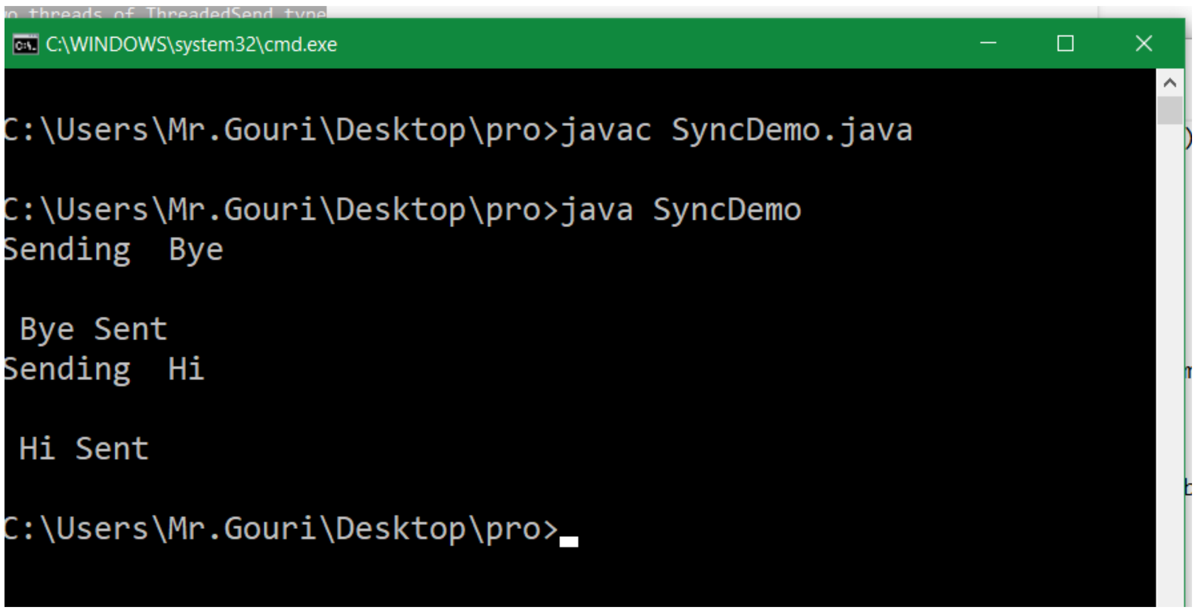
Write a Java program to implement Multithreading with Synchronized.

Solution:

```
import java.io.*;
import java.util.*;

// A Class used to send a message
class Sender
{
   public void send(String msg)
   {
      System.out.println("Sending\t"  + msg );
      try
      {
         Thread.sleep(1000);
      }
      catch (Exception e)
      {
         System.out.println("Thread  interrupted.");
      }
      System.out.println("\n" + msg + "Sent");
   }
}

// Class for send a message using Threads
class ThreadedSend extends Thread
{
   private String msg;
   private Thread t;
   Sender  sender;

   // Recieves a message object and a string
   // message to be sent
   ThreadedSend(String m,  Sender obj)
   {
      msg = m;
      sender = obj;
   }
```

```
    public void run()
    {
        // Only one thread can send a message
        // at a time.
        synchronized(sender)
        {
            // synchronizing the snd object
            sender.send(msg);
        }
    }
}

// Driver class
class SyncDemo
{
    public static void main(String args[])
    {
        Sender snd = new Sender();
        ThreadedSend S1 =
            new ThreadedSend( " Hi " , snd );
        ThreadedSend S2 =
            new ThreadedSend( " Bye " , snd );

        // Start two threads of ThreadedSend type
        S1.start();
        S2.start();

        // wait for threads to end
        try
        {
            S1.join();
            S2.join();
        }
        catch(Exception e)
        {
            System.out.println("Interrupted");
        }
    }
}
```

Output:



```
C:\Users\Mr.Gouri\Desktop\pro>javac SyncDemo.java

C:\Users\Mr.Gouri\Desktop\pro>java SyncDemo
Sending  Bye

 Bye Sent
Sending  Hi

 Hi Sent

C:\Users\Mr.Gouri\Desktop\pro>
```

ASSIGNMENT-8

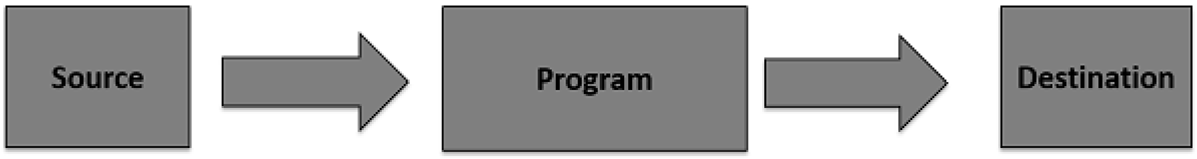
IO Streams (java.io package)

The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java. All these streams represent an input source and an output destination. The stream in the java.io package supports many data such as primitives, object, localized characters, etc.

Stream

A stream can be defined as a sequence of data. There are two kinds of Streams –

- **InPutStream** – The InputStream is used to read data from a source.
- **OutPutStream** – The OutputStream is used for writing data to a destination.



Java provides strong but flexible support for I/O related to files and networks but this tutorial covers very basic functionality related to streams and I/O. We will see the most commonly used examples one by one –

Program1: Byte Streams

Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classes related to byte streams but the most frequently used classes are, FileInputStream and FileOutputStream. Following is an example –

Example

```
//Write a program to makes use of two
classes(FileInputStream  and FileOutputStream) to copy an input file into an
output file.
import java.io.*;
public class CopyFile {

   public static void main(String args[]) throws IOException {
      FileInputStream in = null;
      FileOutputStream out = null;

      try {
         in = new FileInputStream("input.txt");
         out = new FileOutputStream("output.txt");

         int c;
         while ((c = in.read()) != -1) {
            out.write(c);
         }
      }finally {
         if (in != null) {
            in.close();
         }
         if (out != null) {
            out.close();
         }
      }
   }
}
```

Now let's have a file **input.txt** with the following content –

```
This is test for copy file.
```

As a next step, compile the above program and execute it, which will result in creating output.txt file with the same content as we have in input.txt. So let's put the above code in CopyFile.java file and do the following –

```
$javac CopyFile.java
$java CopyFile
```

Program2: Character Streams

Java Byte streams are used to perform input and output of 8-bit bytes, whereas Java Character streams are used to perform input and output for 16-bit unicode. Though there are many classes related to character streams but the most frequently used classes are, FileReader and FileWriter. Though internally FileReader uses FileInputStream and FileWriter uses FileOutputStream but here the major difference is that FileReader reads two bytes at a time and FileWriter writes two bytes at a time.

We can re-write the above example, which makes the use of these two classes to copy an input file (having unicode characters) into an output file −

Example

//Write a program to makes the use of two classes(FileReader and FileWriter) to copy an input file (having unicode characters) into an output file.

```
import java.io.*;
public class CopyFile {

   public static void main(String args[]) throws IOException {
      FileReader in = null;
      FileWriter out = null;

      try {
         in = new FileReader("input.txt");
         out = new FileWriter("output.txt");

         int c;
         while ((c = in.read()) != -1) {
            out.write(c);
         }
      }finally {
         if (in != null) {
            in.close();
         }
         if (out != null) {
            out.close();
         }
      }
   }
}
```

Now let's have a file **input.txt** with the following content −

```
This is test for copy file.
```

As a next step, compile the above program and execute it, which will result in creating output.txt file with the same content as we have in input.txt. So let's put the above code in CopyFile.java file and do the following −

```
$javac CopyFile.java
$java CopyFile
```

Program3: Files and Random Access Files

Using a random access file, we can read from a file as well as write to the file.
Reading and writing using the file input and output streams are a sequential process.
Using a random access file, we can read or write at any position within the file.
An object of the RandomAccessFile class can do the random file access. We can read/write bytes and all primitive types values to a file.
RandomAccessFile can work with strings using its readUTF() and writeUTF() methods.
The RandomAccessFile class is not in the class hierarchy of the InputStream and OutputStream classes.

Mode

A random access file can be created in four different access modes. The access mode value is a string. They are listed as follows:

Mode	Meaning
"r"	The file is opened in a read-only mode.
"rw"	The file is opened in a read-write mode. The file is created if it does not exist.
"rws"	The file is opened in a read-write mode. Any modifications to the file's content and its metadata are written to the storage device immediately.
"rwd"	The file is opened in a read-write mode. Any modifications to the file's content are written t the storage device immediately.

Read and Write

We create an instance of the RandomAccessFile class by specifying the file name and the access mode.

```java
RandomAccessFile  raf = new RandomAccessFile("randomtest.txt", "rw");
```

A random access file has a file pointer that moves forward when we read data from it or write data to it.

The file pointer is a cursor where our next read or write will start.

Its value indicates the distance of the cursor from the beginning of the file in bytes.

We can get the value of file pointer by using its getFilePointer () method.

When we create an object of the RandomAccessFile class, the file pointer is set to zero.

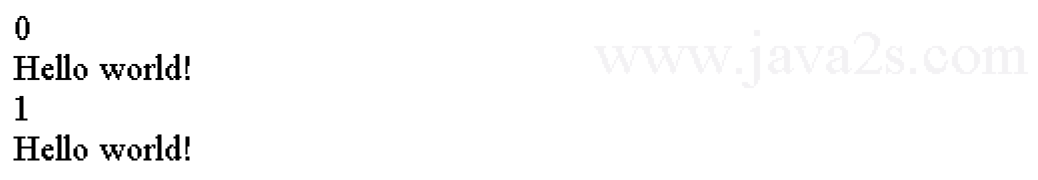
We can set the file pointer at a specific location in the file using the seek () method.

The length () method of a RandomAccessFile returns the current length of the file. We can extend or truncate a file by using its setLength() method.

Example

```java
//Write a java program to shows how to read and write Files Using a RandomAccessFi
le Object.
import java.io.File;
import java.io.IOException;
import java.io.RandomAccessFile;
// w ww  .j av a  2s.  c  om
public class Main {
  public static void main(String[] args) throws IOException {
    String fileName = "randomaccessfile.txt";
    File fileObject = new File(fileName);

    if (!fileObject.exists()) {
      initialWrite(fileName);
    }
    readFile(fileName);
    readFile(fileName);
  }

  public static void readFile(String fileName) throws IOException {
    RandomAccessFile raf = new RandomAccessFile(fileName, "rw");
    int counter = raf.readInt();
    String msg = raf.readUTF();

    System.out.println(counter);
    System.out.println(msg);
    incrementReadCounter(raf);
    raf.close();
  }

  public static void incrementReadCounter(RandomAccessFile raf)
      throws IOException {
    long currentPosition = raf.getFilePointer();
    raf.seek(0);
    int counter = raf.readInt();
    counter++;
    raf.seek(0);
    raf.writeInt(counter);
    raf.seek(currentPosition);
  }

  public static void initialWrite(String fileName) throws IOException {
    RandomAccessFile raf = new RandomAccessFile(fileName, "rw");
    raf.writeInt(0);
    raf.writeUTF("Hello world!");
    raf.close();
  }
}
```

OUTPUT:.



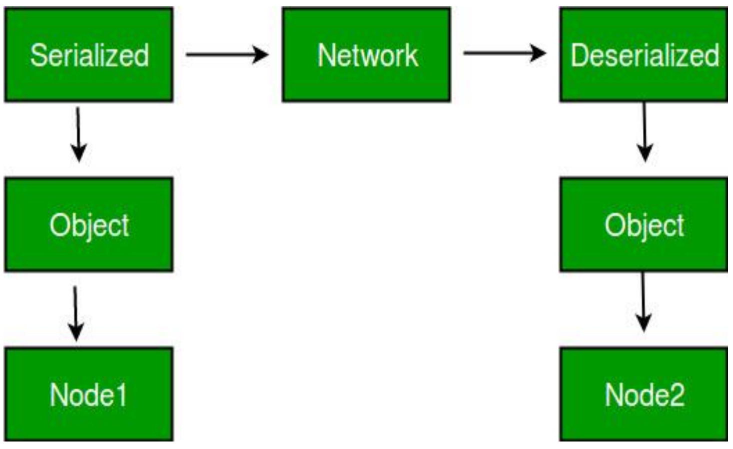
```
0
Hello world!
1
Hello world!
```

Program4: Serialization

Serialization in java is a mechanism of *writing the state of an object into a byte stream*.
It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies.
The reverse operation of serialization is called *deserialization*.

Advantages of Serialization

1. To save/persist state of an object.
2. To travel an object across a network.



Only the objects of those classes can be serialized which are implementing **java.io.Serializable** interface.
Serializable is a **marker interface** (has no data member and method). It is used to "mark" java classes so that objects of these classes may get certain capability. Other examples of marker interfaces are:- Cloneable and Remote.

Points to remember

1. If a parent class has implemented Serializable interface then child class doesn't need to implement it but vice-versa is not true.
2. Only non-static data members are saved via Serialization process.
3. Static data members and transient data members are not saved via Serialization process.So, if you don't want to save value of a non-static data member then make it transient.
4. Constructor of object is never called when an object is deserialized.
5. Associated objects must be implementing Serializable interface.

Example :

```
class A implements Serializable{


// B also implements Serializable
// interface.
B ob=new B();
}
```

Serializing an Object

The ObjectOutputStream class is used to serialize an Object. The following SerializeDemo program instantiates an Employee object and serializes it to a file.When the program is done executing; a file named employee.ser is created. The program does not generate any output, but study the code and try to determine what the program is doing.

Note − When serializing an object to a file, the standard convention in Java is to give the file a .ser extension.

Example

```
//Write a program to implement the Serializing an object.

import java.io.*;

public class SerializeDemo {

      public static void main(String [] args) {

      Employee e = new Employee();

      e.name = "Reyan Ali";

      e.address = "Phokka Kuan, Ambehta Peer";

      e.SSN = 11122333;

      e.number = 101;

      try {

         FileOutputStream fileOut =

         new FileOutputStream("/tmp/employee.ser");

         ObjectOutputStream out = new ObjectOutputStream(fileOut);

         out.writeObject(e);

         out.close();

         fileOut.close();

         System.out.printf("Serialized data is saved in /tmp/employee.ser");

      } catch (IOException i) {

         i.printStackTrace();

      }

   }

}
```

Program5: Collection Frame Work (java.util)

A Collection is a group of individual objects represented as a single unit. Java provides Collection Framework which defines several classes and interfaces to represent a group of objects as a single unit.

The Collection interface (java.util.Collection) and Map interface (java.util.Map) are two main root interfaces of Java collection classes.

Need for Collection Framework:

Before Collection Framework (or before JDK 1.2) was introduced, the standard methods for grouping Java objects (or collections) were array or Vector or Hash table. All three of these collections had no common interface.
For example, if we want to access elements of array, vector or Hash table. All these three have different methods and syntax for accessing members:

```
// Java program to show why collection framework was needed
import java.io.*;
import java.util.*;

class Test
{
    public static void main (String[] args)
    {
        // Creating instances of array, vector and hashtable
        int arr[] = new int[] {1, 2, 3, 4};
        Vector<Integer> v = new Vector();
        Hashtable<Integer, String> h = new Hashtable();
        v.addElement(1);
        v.addElement(2);
        h.put(1,"geeks");
        h.put(2,"4geeks");

        // Array instance creation requires [], while Vector
        // and hastable require ()
        // Vector element insertion requires addElement(), but
        // hashtable element insertion requires put()
         // Accessing first element of array, vector and hashtable
        System.out.println(arr[0]);
        System.out.println(v.elementAt(0));
        System.out.println(h.get(1));

        // Array elements are accessed using [], vector elements
        // using elementAt() and hashtable elements using get()
    }
}
```

Output:

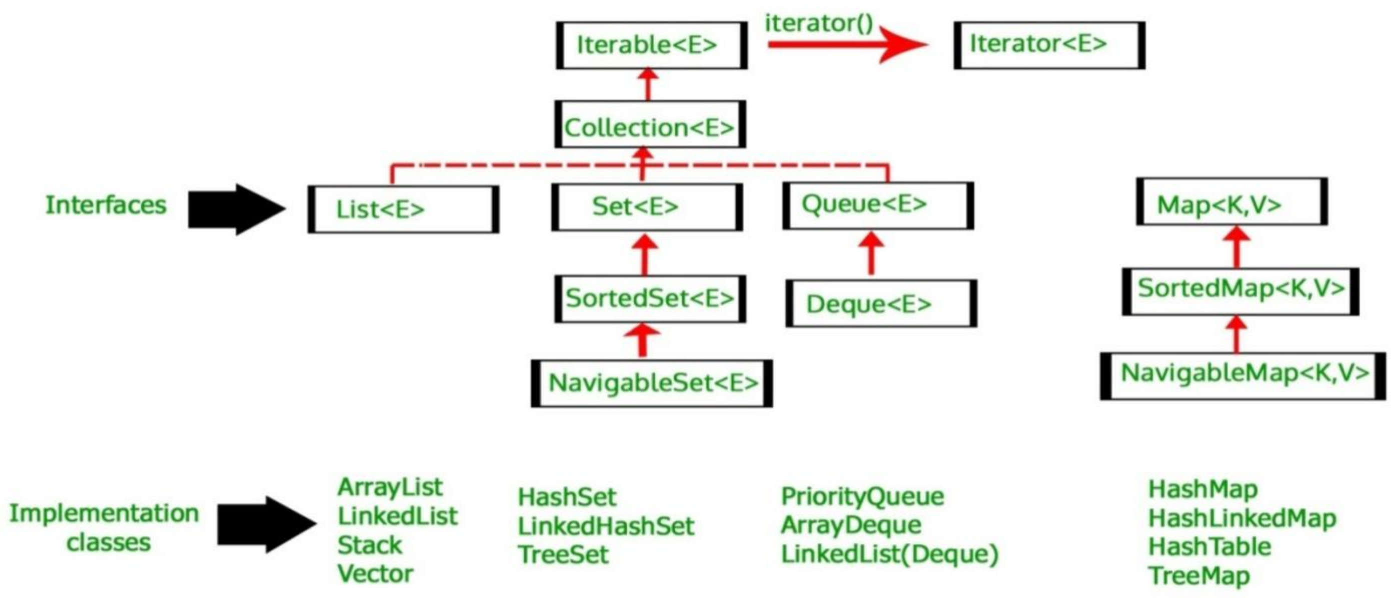
```
1
1
geek
```

Advantages of Collection Framework:

1. Consistent API: The API has basic set of interfaces like Collection, Set, List, or Map. All those classes (such as ArrayList, LinkedList, Vector etc) which implements, these interfaces have some common set of methods.
2. Reduces programming effort: The programmer need not to worry about design of Collection rather than he can focus on its best use in his program.
3. Increases program speed and quality: Increases performance by providing high-performance implementations of useful data structures and algorithms.

Hierarchy of Collection Framework

```
             Collection                Map
         /     /    \      \            |
        /      /      \     \           |
     Set    List    Queue  Dequeue   SortedMap
     /
    /
 SortedSet
            Core Interfaces in Collections

Note that this diagram shows only core interfaces.
Collection : Root interface with basic methods like add(), remove(),
             contains(), isEmpty(), addAll(), ... etc.

Set : Doesn't allow duplicates. Example implementations of Set
      interface are HashSet (Hashing based) and TreeSet (balanced
      BST based). Note that TreeSet implements SortedSet.

List : Can contain duplicates and elements are ordered. Example
       implementations are LinkedList (linked list based) and
       ArrayList (dynamic array based)

Queue : Typically order elements in FIFO order except exceptions
        like PriorityQueue.

Deque : Elements can be inserted and removed at both ends. Allows
        both LIFO and FIFO.

Map : Contains Key value pairs. Doesn't allow duplicates.  Example
      implementation are HashMap and TreeMap.
      TreeMap implements SortedMap.

The difference between Set and Map interface is, in Set we have only
keys, but in Map, we have key value pairs.
```



ASSIGNMENT-9

Util Package interfaces (List, Set, and Map)

It contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).

Program1: (**List)**

The **java.util.ArrayList** class provides resizable-array and implements the **List** interface. Following are the important points about ArrayList –

- It implements all optional list operations and it also permits all elements, includes null.
- It provides methods to manipulate the size of the array that is used internally to store the list.
- The constant factor is low compared to that for the LinkedList implementation.

Class declaration

Following is the declaration for **java.util.ArrayList** class −

```
public class ArrayList<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, Serializable
```

Here **<E>** represents an Element. For example, if you're building an array list of Integers then you'd initialize it as

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

Program2: (**Set)**

The **java.util.EnumSet** class is specialized Set implementations for uses with enum types.Following are the important points about EnumSet −

- All of the elements in an enum set must come from a single enum type that is specified, explicitly or implicitly, when the set is created.
- Enum sets are represented internally as bit vectors.
- EnumSet is not synchronized. If multiple threads access an enum set concurrently, and at least one of the threads modifies the set, it should be synchronized externally.

Class declaration

Following is the declaration for **java.util.EnumSet** class −

```
public abstract class EnumSet<E extends Enum<E>>
extends AbstractSet<E>
implements Cloneable, Serializable
```

Program3: (**Map)**

The **java.util.HashMap** class is the Hash table based implementation of the Map interface. Following are the important points about HashMap −

- This class makes no guarantees as to the iteration order of the map; in particular, it does not guarantee that the order will remain constant over time.
- This class permits null values and the null key.

Class declaration

Following is the declaration for **java.util.HashMap** class −

```
public class HashMap<K,V>
extends AbstractMap<K,V>
implements Map<K,V>, Cloneable, Serializable
```

Parameters

Following is the parameter for **java.util.HashMap** class −

- **K** − This is the type of keys maintained by this map.
- **V** − This is the type of mapped values.

(Applet)

Java Applet Basics

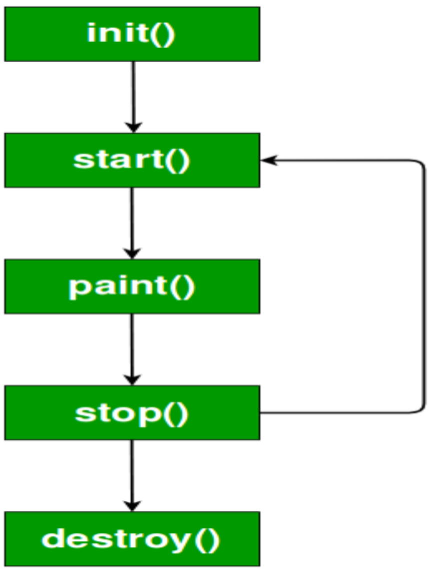
Applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. Applet is embedded in a HTML page using the APPLET or OBJECT tag and hosted on a web server.

Applets are used to make the web site more dynamic and entertaining.

Some important points:

1. All applets are sub-classes (either directly or indirectly) of java.applet.Applet class.
2. Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
3. In general, execution of an applet does not begin at main() method.
4. Output of an applet window is not performed by *System.out.println()*. Rather it is handled with various AWT methods, such as *drawString()*.

Life cycle of an applet:



It is important to understand the order in which the various methods shown in the above image are called. When an applet begins, the following methods are called, in this sequence:

1. init( )
2. start( )
3. paint( )

When an applet is terminated, the following sequence of method calls takes place:
1. stop( )
2. destroy( )
Let's look more closely at these methods.

init( ) : The **init( )** method is the first method to be called. This is where you should initialize variables. This method is called **only once** during the run time of your applet.

start( ) : The **start( )** method is called after **init( )**. It is also called to restart an applet after it has been stopped. Note that **init( )** is called once i.e. when the first time an applet is loaded whereas **start( )** is called each time an applet's HTML document is displayed onscreen. So, if a user leaves a web page and comes back, the applet resumes execution at **start( )**.

paint( ) : The **paint( )** method is called each time an AWT-based applet's output must be redrawn. This situation can occur for several reasons. For example, the window in which the applet is running may be overwritten by another window and then uncovered. Or the applet window may be minimized and then restored.

paint( ) is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, **paint( )** is called.

The **paint( )** method has one parameter of type Graphics. This parameter will contain the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.

stop( ) : The **stop( )** method is called when a web browser leaves the HTML document containing the applet—when it goes to another page, for example. When **stop( )** is called, the applet is probably running. You should use **stop( )** to suspend threads that don't need to run when the applet is not visible. You can restart them when **start( )** is called if the user returns to the page.

destroy ( ) : The **destroy( )** method is called when the environment determines that your applet needs to be removed completely from memory. At this point, you should free up any resources the applet may be using. The **stop( )** method is always called before **destroy( )**.

Creating Hello World applet:

```
// A Hello World Applet
// Save file as HelloWorld.java

import java.applet.Applet;
import java.awt.Graphics;
```

```
// HelloWorld class extends Applet

public class HelloWorld extends Applet
{
   // Overriding paint() method
   @Override
   public void paint(Graphics g)
   {
      g.drawString("Hello World", 20, 20);
   }
 }
```

Explanation:

1. The above java program begins with two import statements. The first import statement imports the Applet class from applet package. Every AWT-based(Abstract Window Toolkit) applet that you create must be a subclass (either directly or indirectly) of Applet class. The second statement imports the Graphics class from awt package.
2. The next line in the program declares the class HelloWorld. This class must be declared as public, because it will be accessed by code that is outside the program. Inside HelloWorld, paint( ) is declared. This method is defined by the AWT and must be overridden by the applet.
3. Inside paint( ) is a call to *drawString( )*, which is a member of the Graphics class. This method outputs a string beginning at the specified X,Y location. It has the following general form:
4. void drawString(String message, int x, int y)

   Here, message is the string to be output beginning at x,y. In a Java window, the upper-left corner is location 0,0. The call to *drawString( )* in the applet causes the message "Hello World" to be displayed beginning at location 20,20.

Notice that the applet does not have a main( ) method. Unlike Java programs, applets do not begin execution at main( ). In fact, most applets don't even have a main( ) method. Instead, an applet begins execution when the name of its class is passed to an applet viewer or to a network browser.

Running the HelloWorld Applet :

After you enter the source code for HelloWorld.java, compile in the same way that you have been compiling java programs(using *javac* command). However running HelloWorld with the *java* command will generate an error because it is not an application.

```
java HelloWorld

Error: Main method not found in class HelloWorld, please define the main
method as:
   public static void main(String[] args)
```

There are **two** standard ways in which you can run an applet:

1. Executing the applet within a Java-compatible web browser.
2. Using an applet viewer, such as the standard tool, appletviewer. An applet viewer executes your applet in a window. This is generally the fastest and easiest way to test your applet.

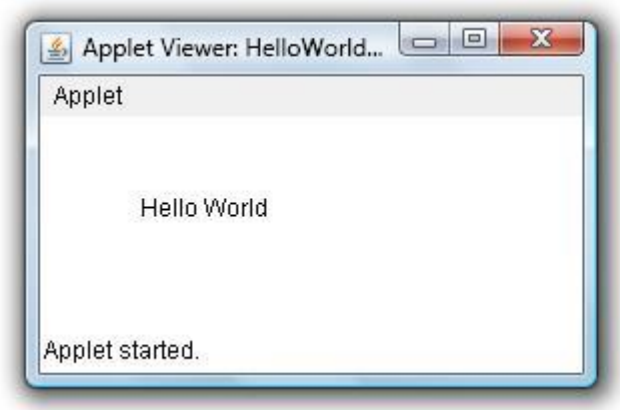
Each of these methods is described next.

1. **Using java enabled web browser :** To execute an applet in a web browser we have to write a short HTML text file that contains a tag that loads the applet. We can use APPLET or OBJECT tag for this purpose. Using APPLET, here is the HTML file that executes HelloWorld :
2. `<applet code="HelloWorld" width=200 height=60>`
3. `</applet>`

   The width and height statements specify the dimensions of the display area used by the applet. The APPLET tag contains several other options. After you create this html file, you can use it to execute the applet.

   NOTE : Chrome and Firefox no longer supports NPAPI (technology required for Java applets). Refer here

4. **Using appletviewer :** This is the easiest way to run an applet. To execute HelloWorld with an applet viewer, you may also execute the HTML file shown earlier. For example, if the preceding HTML file is saved with RunHelloWorld.html,then the following command line will run HelloWorld :
5. `appletviewer RunHelloWorld.html`



```
Hello World

Applet started.
```

6. **appletviewer with java source file :** If you include a comment at the head of your Java source code file that contains the APPLET tag then your code is documented with a prototype of the necessary HTML statements, and you can run your compiled applet merely by starting the applet viewer with your Java source code file. If you use this method, the HelloWorld source file looks like this :

```
// A Hello World Applet
// Save file as HelloWorld.java

import java.applet.Applet;
import java.awt.Graphics;

/*
<applet code="HelloWorld" width=200 height=60>
</applet>
*/

// HelloWorld class extends Applet
public class HelloWorld extends Applet
{
    // Overriding paint() method
    @Override
    public void paint(Graphics g)
    {
        g.drawString("Hello World", 20, 20);
    }

}
```

• with this approach, first compile HelloWorld.java file and then simply run below command to run applet:

```
appletviewer HelloWorld
```

Features of Applets over HTML

- Displaying dynamic web pages of a web application.
- Playing sound files.
- Displaying documents
- Playing animations

Restrictions imposed on Java applets
Due to security reasons, the following restrictions are imposed on Java applets:

1. An applet cannot load libraries or define native methods.
2. An applet cannot ordinarily read or write files on the execution host.
3. An applet cannot read certain system properties.
4. An applet cannot make network connections except to the host that it came from.
5. An applet cannot start any program on the host that’s executing it.

Program4:

Write a Java program to showing Simple example of Applet by html file

To execute the applet by html file, create an applet and compile it. After that create an html file and place the applet code in html file. Now click the html file.

Solution:

```
//Step- 1 First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet
{
    public void paint(Graphics g)
    {
    g.drawString("welcome",150,150);
    }
}
//Step-2 myapplet.html

1.  <html>
2.  <body>
3.  <applet code="First.class" width="300" height="300">
4.  </applet>
5.  </body>
6.  </html>
```

Event and Listener (Java Event Handling)

Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling.

Java Event classes and Listener interfaces

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener

ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

Program5:

Write a Java program to implement Event Handling in Applet.

Solution:

```
//Step- 1 Save as demo.java
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class EventApplet extends Applet implements ActionListener
{
    Button b;
    TextField tf;

    public void init()
    {
            tf=new TextField();
            tf.setBounds(30,40,150,20);

            b=new Button("Click");
            b.setBounds(80,150,60,50);

            add(b);add(tf);
            b.addActionListener(this);

            setLayout(null);
     }
    public void actionPerformed(ActionEvent e)
    {
            tf.setText("Welcome");
    }
}
```

//Step-2 Save as demo.html

```
1.  <html>
2.  <body>
3.  <applet code="EventApplet.class" width="300" height="300">
4.  </applet>
5.  </body>
6.  </html>
```

ASSIGNMENT-10

Program1:

Write a Java program to display digital clock by using Applet.

Solution:

//Step- 1 Save as demo.java

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

public class DigitalClock extends Applet implements Runnable
 {

  Thread t = null;
  int hours=0, minutes=0, seconds=0;
  String timeString = "";

  public void init() {
    setBackground( Color.green);
  }

  public void start() {
      t = new Thread( this );
      t.start();
  }
public void run()
{
    try
    {
      while (true)
     {

         Calendar cal = Calendar.getInstance();
         hours = cal.get( Calendar.HOUR_OF_DAY );
         if ( hours > 12 ) hours -= 12;
         minutes = cal.get( Calendar.MINUTE );
         seconds = cal.get( Calendar.SECOND );

         SimpleDateFormat formatter = new SimpleDateFormat("hh:mm:ss");
         Date date = cal.getTime();
         timeString = formatter.format( date );

         repaint();
         t.sleep( 1000 );  // interval given in milliseconds
      }
    }
    catch (Exception e) { }
  }
public void paint( Graphics g )
{
    g.setColor( Color.blue );
    g.drawString( timeString, 50, 50 );
  }
}
```

//Step-2 myapplet.html

```
<html>
<body>
<applet code="DigitalClock.class" width="300" height="300">
</applet>
</body>
</html>
```

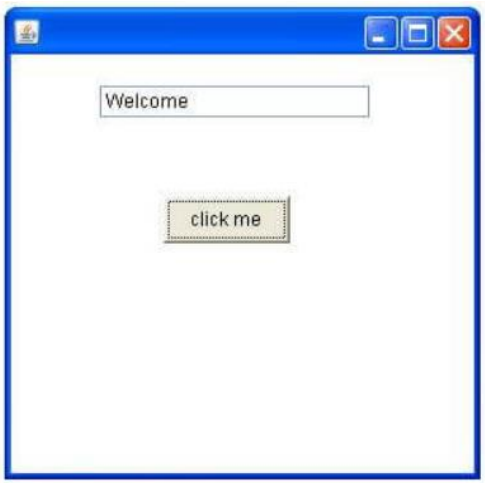
Program2:

Write a Java program of event handling by implementing Action Listener.

Solution:

```
import java.awt.*;
import java.awt.event.*;
class AEvent extends Frame implements ActionListener
{
    TextField tf;
    AEvent()
    {
            //create components
            tf=new TextField();
            tf.setBounds(60,50,170,20);
            Button b=new Button("click me");
            b.setBounds(100,120,80,30);

            //register listener
            b.addActionListener(this);//passing current instance

            //add components and set size, layout and visibility
            add(b);add(tf);
            setSize(300,300);
            setLayout(null);
            setVisible(true);
    }
    public void actionPerformed(ActionEvent e)
    {
            tf.setText("Welcome");
    }
    public static void main(String args[])
    {
            new AEvent();
    }
}
```

Output:



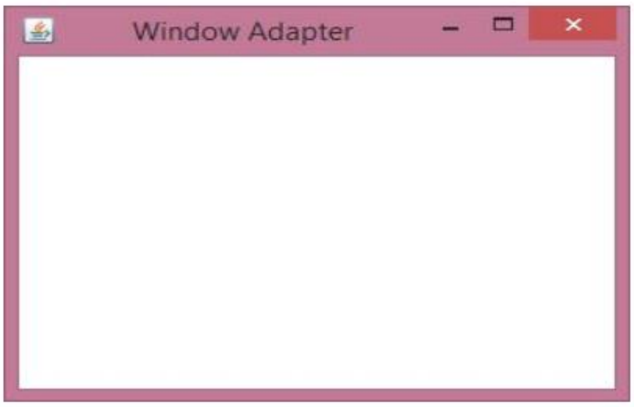
Program3:

Write a Java program to implement window Adapter.

Solution:

```
import java.awt.*;
import java.awt.event.*;
public class AdapterExample
{
    Frame f;
    AdapterExample(){
        f=new Frame("Window Adapter");
        f.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e) {
                f.dispose();
            }
        });
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
   public static void main(String[] args)
  {
    new AdapterExample();
   }
}
```

Output:



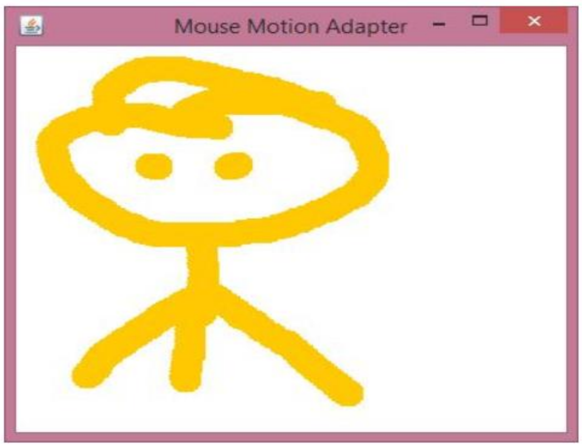
Program4:

Write a Java program to implement MouseMotion Adapter.

Solution:

```
import java.awt.*;
import java.awt.event.*;
public class MouseMotionAdapterExample extends MouseMotionAdapter{
   Frame f;
   MouseMotionAdapterExample(){
      f=new Frame("Mouse Motion Adapter");
      f.addMouseMotionListener(this);


      f.setSize(300,300);
      f.setLayout(null);
      f.setVisible(true);
   }
public void mouseDragged(MouseEvent e) {
   Graphics g=f.getGraphics();
   g.setColor(Color.ORANGE);
   g.fillOval(e.getX(),e.getY(),20,20);
}
public static void main(String[] args) {
   new MouseMotionAdapterExample();
}
}
```

Output:



Program5:

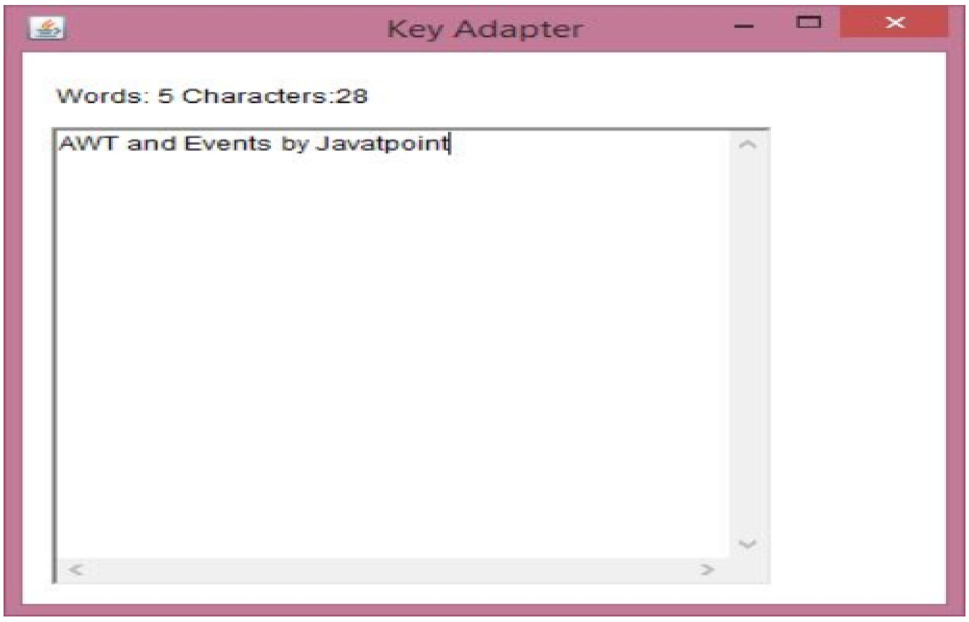
Write a Java program to implement Key Adapter.

Solution:

```
import java.awt.*;
import java.awt.event.*;
public class KeyAdapterExample extends KeyAdapter{
   Label l;
   TextArea area;
   Frame f;
   KeyAdapterExample(){
      f=new Frame("Key Adapter");
      l=new Label();
      l.setBounds(20,50,200,20);
      area=new TextArea();
      area.setBounds(20,80,300, 300);
      area.addKeyListener(this);

      f.add(l);f.add(area);
      f.setSize(400,400);
      f.setLayout(null);
      f.setVisible(true);
   }
   public void keyReleased(KeyEvent e) {
      String text=area.getText();
      String words[]=text.split("\\s");
      l.setText("Words: "+words.length+" Characters:"+text.length());
   }
```

```
    public static void main(String[] args) {
        new KeyAdapterExample();
    }
}
```

Output:



```
Words: 5 Characters:28
AWT and Events by Javatpoint
```

========================**********========================