KALLAM HARANATHAREDDY INSTITUTE OF TECHNOLOGY

(Approved by AICTE New Delhi & Affiliated to JNTUK, Kakinada)

NH- 5, Chowdavaram, Guntur-522 019

An ISO 9001:2015 Certified Institution, Accredited by NAAC

JAVA PROGRAMMING LAB MANUAL

by

Dr. Md. Umar Khan

For B.Tech CSE II Year

II Semester(R-16)

COURSE STRUCTURE AND SYLLABUS

For

COMPUTER SCIENCE AND ENGINEERING

(Applicable for batches admitted from 2016-2017)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY: KAKINADA
KAKINADA - 533 003, Andhra Pradesh, India

II Year – II Semester

L	T	P	C
0	0	3	2

JAVA PROGRAMMING LAB
(JNTUK Syllabus)

Exercise - 1 (Basics)
a). Write a JAVA program to display default value of all primitive data type of JAVA
b). Write a java program that display the roots of a quadratic equation ax2+bx=0. Calculate the discriminate D and basing on value of D, describe the nature of root.
c). Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers. Take as input the speed of each racer and print back the speed of qualifying racers.
d) Write a case study on **public static void main(250 words)**
Exercise - 2 (Operations, Expressions, Control-flow, Strings)
a). Write a JAVA program to search for an element in a given list of elements using binary search mechanism.
b). Write a JAVA program to sort for an element in a given list of elements using bubble sort
(c). Write a JAVA program to sort for an element in a given list of elements using merge sort.
(d) Write a JAVA program using StringBufferto delete, remove character.
Exercise - 3 (Class, Objects)
a). Write a JAVA program to implement class mechanism. – Create a class, methods and invoke them inside main method.
b). Write a JAVA program to implement constructor.
Exercise - 4 (Methods)
a). Write a JAVA program to implement constructor overloading.
b). Write a JAVA program implement method overloading.
Exercise - 5 (Inheritance)
a). Write a JAVA program to implement Single Inheritance
b). Write a JAVA program to implement multi level Inheritance
c). Write a java program for abstract class to find areas of different shapes
Exercise - 6 (Inheritance - Continued)
a). Write a JAVA program give example for "super" keyword.
b). Write a JAVA program to implement Interface. What kind of Inheritance can be achieved?
Exercise - 7 (Exception)
a).Write a JAVA program that describes exception handling mechanism
b).Write a JAVA program Illustrating Multiple catch clauses
Exercise – 8 (Runtime Polymorphism)
a). Write a JAVA program that implements Runtime polymorphism
b). Write a Case study on run time polymorphism, inheritance that implements in above problem
Exercise – 9 (User defined Exception)
a). Write a JAVA program for creation of Illustrating throw
b). Write a JAVA program for creation of Illustrating finally
c). Write a JAVA program for creation of Java Built-in Exceptions
d).Write a JAVA program for creation of User Defined Exception
Exercise – 10 (Threads)
a). Write a JAVA program that creates threads by extending Thread class .First thread display "Good Morning "every 1 sec, the second thread displays "Hello "every 2 seconds and the third display "Welcome" every 3 seconds ,(Repeat the same by implementing Runnable)
b). Write a program illustrating **isAlive** and **join ()**
c). Write a Program illustrating Daemon Threads.

Exercise - 11 (Threads continuity)

a).Write a JAVA program Producer Consumer Problem

b).Write a case study on thread Synchronization after solving the above producer consumer problem

Exercise – 12 (Packages)

a). Write a JAVA program illustrate class path

b). Write a case study on including in class path in your os environment of your package.

c). Write a JAVA program that import and use the defined your package in the previous Problem

Exercise - 13 (Applet)

a).Write a JAVA program to paint like paint brush in applet.

b) Write a JAVA program to display analog clock using Applet.

c). Write a JAVA program to create different shapes and fill colors using Applet.

Exercise - 14 (Event Handling)

a).Write a JAVA program that display the x and y position of the cursor movement using Mouse.

b).Write a JAVA program that identifies key-up key-down event user entering text in a Applet.

Exercise - 15 (Swings)

a).Write a JAVA programto build a Calculator in Swings

b). Write a JAVA program to display the digital watch in swing tutorial.

Exercise – 16 (Swings - Continued)

a). Write a JAVA program that to create a single ball bouncing inside a JPanel.

b). Write a JAVA program JTree as displaying a real tree upside down

JAVA PROGRAMMING LAB
Cycle-I Programs

Exercise - 1 (Basics)

a). Write a JAVA program to display default value of all primitive data type of JAVA
b). Write a java program that display the roots of a quadratic equation ax2+bx=0. Calculate the discriminate D and basing on value of D, describe the nature of root.
c). Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers. Take as input the speed of each racer and print back the speed of qualifying racers.
d) Write a case study on **public static void main(250 words)**

Exercise - 2 (Operations, Expressions, Control-flow, Strings)

a). Write a JAVA program to search for an element in a given list of elements using binary search mechanism.
b). Write a JAVA program to sort for an element in a given list of elements using bubble sort
(c). Write a JAVA program to sort for an element in a given list of elements using merge sort.
(d) Write a JAVA program using StringBuffer to delete, remove character.

Exercise - 3 (Class, Objects)

a). Write a JAVA program to implement class mechanism. – Create a class, methods and invoke them inside main method.
b). Write a JAVA program to implement constructor.

Exercise - 4 (Methods)

a). Write a JAVA program to implement constructor overloading.
b). Write a JAVA program implement method overloading.

Exercise - 5 (Inheritance)

a). Write a JAVA program to implement Single Inheritance
b). Write a JAVA program to implement multi level Inheritance
c). Write a java program for abstract class to find areas of different shapes

Exercise - 6 (Inheritance - Continued)

a). Write a JAVA program give example for “super” keyword.
b). Write a JAVA program to implement Interface. What kind of Inheritance can be achieved?

Exercise - 7 (Exception)

a).Write a JAVA program that describes exception handling mechanism
b).Write a JAVA program Illustrating Multiple catch clauses

Exercise – 8 (Runtime Polymorphism)

a). Write a JAVA program that implements Runtime polymorphism
b). Write a Case study on run time polymorphism, inheritance that implements in above problem

Exercise – 9 (User defined Exception)

a). Write a JAVA program for creation of Illustrating throw
b). Write a JAVA program for creation of Illustrating finally
c). Write a JAVA program for creation of Java Built-in Exceptions
d).Write a JAVA program for creation of User Defined Exception

Exercise – 10 (Packages)

a). Write a JAVA program illustrate class path
b). Write a case study on including in class path in your os environment of your package.
c). Write a JAVA program that import and use the defined your package in the previous Problem

JAVA PROGRAMMING LAB
Cycle-II Programs

Exercise – 11 (Threads)

a). Write a JAVA program that creates threads by extending Thread class .First thread display “Good Morning “every 1 sec, the second thread displays “Hello “every 2 seconds and the third display “Welcome” every 3 seconds ,(Repeat the same by implementing Runnable)
b). Write a program illustrating **isAlive** and **join ()**
c). Write a Program illustrating Daemon Threads.

Exercise - 12 (Threads continuity)

a).Write a JAVA program Producer Consumer Problem
b).Write a case study on thread Synchronization after solving the above producer consumer problem

Exercise - 13 (Applet)

a).Write a JAVA program to paint like paint brush in applet.
b) Write a JAVA program to display analog clock using Applet.
c). Write a JAVA program to create different shapes and fill colors using Applet.

Exercise - 14 (Event Handling)

a).Write a JAVA program that display the x and y position of the cursor movement using Mouse.
b).Write a JAVA program that identifies key-up key-down event user entering text in a Applet.

Exercise - 15 (Swings)

a).Write a JAVA program to build a Calculator in Swings
b). Write a JAVA program to display the digital watch in swing tutorial.

Exercise – 16 (Swings - Continued)

a). Write a JAVA program that to create a single ball bouncing inside a JPanel.
b). Write a JAVA program JTree as displaying a real tree upside down

CONTENTS

Exercise - 1 (Basics)

a) Displaying default value of all primitive data types

Aim: To write a JAVA program to display default value of all primitive data type of JAVA

Program:

```
class  defaultdemo
{
	static byte b;
	static short s;
	static int i;
	static long l;
	static float f;
	static double d;
	static char c;
	static boolean bl;
	public static void main(String[] args)
	{
		System.out.println("The default values of primitive data types are:");
		System.out.println("Byte :"+b);
		System.out.println("Short :"+s);
		System.out.println("Int :"+i);
		System.out.println("Long :"+l);
		System.out.println("Float :"+f);
		System.out.println("Double :"+d);
		System.out.println("Char :"+c);
		System.out.println("Boolean :"+bl);
	}
}
```

Output:

```
The default values of primitive data types are:
Byte :0
Short :0
Int :0
Long :0
Float :0.0
Double :0.0
Char :
Boolean :false
```

b) Roots of a quadratic equation

Aim: To write a java program that display the roots of a quadratic equation ax2+bx=0. Calculate the discriminate D and basing on value of D, describe the nature of root.

Program:

```
import java.util.*;
class quadraticdemo
{
	public static void main(String[] args)
	{
		int a, b, c;
		double r1, r2, D;
		Scanner s = new Scanner(System.in);
		System.out.println("Given quadratic equation:ax^2 + bx + c");
		System.out.print("Enter a:");
		a = s.nextInt();
		System.out.print("Enter b:");
		b = s.nextInt();
		System.out.print("Enter c:");
		c = s.nextInt();
		D = b * b - 4 * a * c;
		if(D > 0)
		{
			System.out.println("Roots are real and unequal");
			r1 = ( - b + Math.sqrt(D))/(2*a);
			r2 = (-b - Math.sqrt(D))/(2*a);
			System.out.println("First root is:"+r1);
			System.out.println("Second root is:"+r2);
		}
		else if(D == 0)
		{
			System.out.println("Roots are real and equal");
			r1 = (-b+Math.sqrt(D))/(2*a);
			System.out.println("Root:"+r1);
		}
		else
		{
			System.out.println("Roots are imaginary");
		}
	}
}
```

Output:

```
Given quadratic equation:ax^2 + bx + c
Enter a:2
Enter b:3
Enter c:1
Roots are real and unequal
First root is:-0.5
Second root is:-1.0
```

c) Bike Race

Aim: Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers. Take as input the speed of each racer and print back the speed of qualifying racers.

Program:

```
import java.util.*;
class racedemo
{
	public static void main(String[] args)
	{
		float s1,s2,s3,s4,s5,average;
		Scanner s = new Scanner(System.in);
		System.out.println("Enter speed of first racer:");
		s1 = s.nextFloat();
		System.out.println("Enter speed of second racer:");
		s2 = s.nextFloat();
		System.out.println("Enter speed of third racer:");
		s3 = s.nextFloat();
		System.out.println("Enter speed of fourth racer:");
		s4 = s.nextFloat();
		System.out.println("Enter speed of fifth racer:");
		s5 = s.nextFloat();
		average=(s1+s2+s3+s4+s5)/5;
		if(s1>average)
		System.out.println("First racer is qualify racer:");
		else if(s2>average)
		System.out.println("Second racer is qualify racer:");
		else if(s3>average)
		System.out.println("Third racer is qualify racer:");
		else if(s4>average)
		System.out.println("Fourth racer is qualify racer:");
		else if(s5>average)
		System.out.println("Fifth racer is qualify racer:");
	}
}
```

Output:

```
Enter speed of first racer:
4.5
Enter speed of second racer:
6.7
Enter speed of third racer:
3.8
Enter speed of fourth racer:
5.3
Enter speed of fifth racer:
4.9
Second racer is qualify racer:
```

d) A case study

Aim: A case study on public static void main(250 words)

Case study:

- The program structure of a simple java program is given below with different steps

Step-1: Click start+run and then type notepad in run dialog box and click OK. It displays Notepad.

Step-2: In run dialogbox type cmd and click OK. It displays command prompt.

Step-3: Type the following program in the Notepad and save the program as "example.java" in a current working directory.

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

Step-4 (Compilation): To compile the program type the following in current working directory and then click enter.

```
c:\xxxx >javac example.java
```

Step-5 (Execution): To run the program type the following in current working directory and then click enter.

```
c:\xxxx>java example
```

Explanation:

- Generally the file name and class name should be same. If it is not same then the java file can be compiled but it cannot be executed. That is when execution it gives the following error

```
Exception in thread "main" java.lang.NoClassDefFoundError: ex
```

- In "public static void main(String args[])" statement
  - **public** is an access specifier. If a class is visible to all classes then public is used
  - **main()** must be declared as public since it must be called by outside of its class.
  - The keyword **static** allows **main()** to be called without creating object of the class.
  - The keyword **void** represents that **main( )** does not return a value.
  - The main method contains one parameter String args[].
  - We can send some input values (arguments) at run time to the String args[] of the main method . These arguments are called command line arguments. These command line arguments are passed at the command prompt.
- In System.out.println("Welcome"); statement
  - **System** is a predefined class that provides access to the system.
  - **out** is the output stream.
  - **println()** method display the output in different lines. If we use **print()** method it display the output in the same line

Exercise - 2 (Operations, Expressions, Control-flow, Strings)

a) Implementation of Binary search mechanism

Aim: To write a JAVA program to search for an element in a given list of elements using binary search mechanism

Program:

```
import java.util.Scanner;
class binarysearchdemo
{
	public static void main(String args[])
	{
		int n, i, num,first, last, middle;
		int a[ ]=new int[20];
		Scanner s = new Scanner(System.in);
		System.out.println("Enter total number of elements:");
		n = s.nextInt();
		System.out.println("Enter elements in sorted order:");
		for (i = 0; i < n; i++)
		a[i] = s.nextInt();
		System.out.println("Enter the search value:");
		num = s.nextInt();
		first = 0;
		last = n - 1;
		middle = (first + last)/2;
		while( first <= last )
		{
			if ( a[middle] < num )
			first = middle + 1;
			else if ( a[middle] == num )
			{
				System.out.println("number found");
				break;
			}
			else
			{
				last = middle - 1;
			}
			middle = (first + last)/2;
		 }
		 if ( first > last )
		 System.out.println( " Number is not found");
	 }
}
```

Output:

```
Enter total number of elements:
5
Enter elements:
2 4 6 8 9
Enter the search value:
8
number found
```

b) Bubble sort

Aim: To write a JAVA program to sort for an element in a given list of elements using bubble sort

Program:

```
import java.util.Scanner;
class bubbledemo
{
	public static void main(String args[])
	{
		int n, i,j, temp;
		int a[ ]=new int[20];
		Scanner s = new Scanner(System.in);
		System.out.println("Enter total number of elements:");
		n = s.nextInt();
		System.out.println("Enter elements:");
		for (i = 0; i < n; i++)
		a[i] = s.nextInt();
		for(i=0;i<n;i++)
		{
			for(j=0;j<n-1;j++)
			{
				if(a[j]>a[j+1])
				{
					temp=a[j];
					a[j]=a[j+1];
					a[j+1]=temp;
				}
			}
		}
		System.out.println("The sorted elements are:");
		for(i=0;i<n;i++)
		System.out.print("\t"+a[i]);
	 }
}
```

Output:

```
Enter total number of elements:
10
Enter elements:
3 2 5 7 6 8 9 1 4 0
The sorted elements are:
 0	1	2	3	4	5	6	7	8	9
```

c) Merge sort:

Aim: To write a JAVA program to sort for an element in a given list of elements using merge sort

Program:

```
import java.util.*;
class mergedemo
{
	public static void main(String args[])
	{
		int n1,n2,i,j,k;
		int a[ ]=new int[20];
		int b[ ]=new int[20];
		int c[ ]=new int[20];
		Scanner s = new Scanner(System.in);
		System.out.println("Enter number of elements in first array:");
		n1 = s.nextInt();
		System.out.println("Enter sorted elements of first array:");
		for (i = 0; i < n1; i++)
		a[i] = s.nextInt();
		System.out.println("Enter number of elements in second array:");
		n2 = s.nextInt();
		System.out.println("Enter sorted elements of second array:");
		for (j = 0; j < n2; j++)
		b[j] = s.nextInt();
		i = 0;
		j = 0;
		k = 0;
		while((i < n1) && (j <n2))
		{
			if(a[i] > b[j])
			c[k++] = b[j++];
			else
			c[k++] = a[i++];
		}
		while(i < n1)
		c[k++] = a[i++];
		while(j < n2)
		c[k++] = b[j++];
		System.out.println("After merging the elements are:\n");
		for(i = 0; i < (n1 + n2); i++)
		System.out.print("\t"+c[i]);
	}
}
```

Output:

```
Enter number of elements in first array:
6
Enter elements of first array:
8 9 12 13 15 18
Enter number of elements in second array:
5
Enter elements of second array:
6 7 10 11 20
After merging the elements are:
 6	7	8	9	10	11	12	13	15	18	20
```

d) Implementing StringBuffer

Aim: To write a JAVA program using StringBuffer to delete, remove character

Program:

```
class stringbufferdemo
{
	public static void main(String[] args)
	{
		StringBuffer sb1 = new StringBuffer("Hello World");
		sb1.delete(0,6);
		System.out.println(sb1);
		StringBuffer sb2 = new StringBuffer("Some Content");
		System.out.println(sb2);
		sb2.delete(0, sb2.length());
		System.out.println(sb2);
		StringBuffer sb3 = new StringBuffer("Hello World");
		sb3.deleteCharAt(0);
		System.out.println(sb3);
	}
}
```

Output:

```
World
Some Content

ello World
```

Exercise - 3 (Class, Objects)
a) Implementing Class & Objects

Aim: To write a JAVA program to implement class mechanism. – Create a class, methods and invoke them inside main method

Programs:

1.no return type and without parameter-list:

```
class A
{
        int l=10,b=20;
        void display()
        {
                System.out.println(l);
                System.out.println(b);
        }
}
class methoddemo
{
        public static void main(String args[])
        {
                A a1=new A();
                a1.display();
        }
}
```

Output:

10
20

2.no return type and with parameter-list:

```
class A
{
       void display(int l,int b)
       {
            System.out.println(l);
            System.out.println(b);
       }
 }
class methoddemo
{
      public static void main(String args[])
      {
              A a1=new A();
              a1.display(10,20);
       }
}
```

Output:

10
20

3. return type and without parameter-list

```
class A
{
       int l=10,b=20;
```

```
        int area()
        {
                return l*b;
        }
}
class methoddemo
{
        public static void main(String args[])
        {
                A a1=new A();
                int r=a1.area();
                System.out.println("The area is: "+r);
        }
}
```

Output:

The area is:200

4.return type and with parameter-list:

```
class A
{
        int area(int l,int b)
        {
                return l*b;
        }
}
class methoddemo
{
        public static void main(String args[])
        {
                A a1=new A();
                int r=a1.area(10,20);
                System.out.println("The area is:"+r);
        }
}
```

Output:

The area is:200

b) Implementing Constructor

Aim: To write a JAVA program to implement constructor

Programs:

(i)A constructor with no parameters:

```
class A
{
        int l,b;
        A()
        {
                l=10;
                b=20;
        }
        int area()
        {
                return l*b;
        }
}
class constructordemo
{
        public static void main(String args[])
        {
                A a1=new A();
                int r=a1.area();
                System.out.println("The area is: "+r);

        }
}
```

Output:

The area is:200

(ii)A constructor with parameters

```
class A
{
        int l,b;
        A(int u,int v)
        {
                l=u;
                b=v;
        }
        int area()
        {
               return l*b;
        }
}
class constructordemo
{
        public static void main(String args[])
        {
                A a1=new A(10,20);
                int r=a1.area();
                System.out.println("The area is: "+r);
        }
}
```

Output:

The area is:200

Exercise - 4 (Methods)
a) Constructor Overloading

Aim: To write a JAVA program to implement constructor overloading

Program:

```
class A
{
        int l,b;
        A()
        {
                l=10;
                b=20;
                 }
        A(int u,int v)
        {
                l=u;
                b=v;
        }
        int area()
        {
                return l*b;
        }
}
class overconstructdemo
{
         public static void main(String args[])
         {
                 A a1=new A();
                int r1=a1.area();
                System.out.println("The area is: "+r1);
                A a2=new A(30,40);
                int r2=a2.area();
                System.out.println("The area is: "+r2);
        }
}
```

Output:
The area is: 200
The area is: 1200

b) Method Overloading

Aim: To write a JAVA program implement method overloading

Program:

```
class A
{
        int l=10,b=20;
        int area()
        {
                return l*b;
        }
        int area(int l,int b)
        {
                return l*b;
        }
}
class overmethoddemo
{
        public static void main(String args[])
        {
                A a1=new A();
                int r1=a1.area();
                System.out.println("The area is: "+r1);
                int r2=a1.area(5,20);
                System.out.println("The area is: "+r2);
        }
}
```

Output:

```
The area is: 200
The area is: 100
```

Exercise - 5 (Inheritance)
a)Implementing Single Inheritance

Aim: To write a JAVA program to implement Single Inheritance

Program:

```
class A
{
    A()
    {
        System.out.println("Inside A's Constructor");
    }
}
class B extends A
{
    B()
    {
        System.out.println("Inside B's Constructor");
    }
}
class singledemo
{
     public static void main(String args[])
     {
        B b1=new B();
     }
}
```

Output:

```
Inside A's Constructor
Inside B's Constructor
```

b)Multi level Inheritance

Aim: To write a JAVA program to implement multi level Inheritance

Program:

```
class A
{
    A()
    {
        System.out.println("Inside A's Constructor");
    }
}
class B extends A
{
    B()
    {
        System.out.println("Inside B's Constructor");
    }
}
class C extends B
{
    C()
    {
        System.out.println("Inside C's Constructor");
    }
}
class multidemo
{
    public static void main(String args[])
    {
        C c1=new C();

    }
}
```

Output:

```
Inside A's Constructor
Inside B's Constructor
Inside C's Constructor
```

c)Abstract Class

Aim: To write a java program for abstract class to find areas of different shapes

Program:

```
abstract class shape
{
        abstract double area();
}
class rectangle extends shape
{
        double l=12.5,b=2.5;
        double area()
        {
                return l*b;
        }
}
class triangle extends shape
{
        double b=4.2,h=6.5;
        double area()
        {
                return 0.5*b*h;
        }
}
class square extends shape
{
        double s=6.5;
        double area()
        {
                return 4*s;
        }
}
class shapedemo
{
        public static void main(String[] args)
        {
                rectangle r1=new rectangle();
                triangle t1=new triangle();
                square s1=new square();
                System.out.println("The area of rectangle is: "+r1.area());
                System.out.println("The area of triangle is: "+t1.area());
                System.out.println("The area of square is: "+s1.area());
        }
}
```

Output:

```
The area of rectangle is: 31.25
The area of triangle is: 13.65
The area of square is: 26.0
```

Exercise - 6 (Inheritance - Continued)

a)super keyword implementation

Aim: Write a JAVA program give example for “super” keyword

Programs:

(i)Using super to call super class constructor (Without parameters)

```
class A
{
	int l,b;
	A()
	{
	    l=10;
	    b=20;
	}
}
class B extends A
{
	int h;
	B()
	{
	     super();
	     h=30;
	}
	int volume()
	{
	     return l*b*h;
	}
}
class superdemo
{
	 public static void main(String args[])
	 {
		B b1=new B();
		int r=b1.volume();
		System.out.println("The vol. is: "+r);
	 }
}
```

Output:

```
The vol. is:6000
```

(ii)Using super to call super class constructor (With parameters)

```
class A
{
	 int l,b;
	 A(int u,int v)
	 {
		l=u;
		b=v;
	 }
}
```

```
class B extends A
{
        int h;
        B(int u,int v,int w)
        {
                super(u,v);
                h=w;
        }
        int volume()
        {
                return l*b*h;
        }
}
class superdemo
{
        public static void main(String args[])
        {
                B b1=new B(30,20,30);
                int r=b1.volume();
                System.out.println("The vol. is: "+r);
        }
}
```

Output:

The vol. is:18000

b) Implementing interface

Aim: To write a JAVA program to implement Interface.

Programs:

(i) First form of interface implementation

```
interface A
{
	void display();
}
class B implements A
{
	public void display()
	{
		System.out.println("B's method");
	}
}
class C extends B
{
	public void callme()
	{
		System.out.println("C's method");
	}
}
class interfacedemo
{
	public static void main(String args[])
	{
		C c1=new C();
		c1.display();
		c1.callme();
	}
}
```

Output:

```
B's method
C's method
```

(ii) Second form of interface implementation

```
interface D
{
	void display();
}
interface E extends D
{
	void show();
}
class A
{
	void callme()
	{
		System.out.println("This is in callme method");
	}
}
```

```
class B extends A implements E
{
	public void display()
	{
		System.out.println("This is in display method");
	}
	public void show()
	{
		System.out.println("This is in show method");
	}
}
class C extends B
{
	void call()
	{
			System.out.println("This is in call method");
	}
}
class interfacedemo
{
	public static void main(String args[])
	{
		C c1=new C();
		c1.display();
		c1.show();
		c1.callme();
		c1.call();
	}
}
```

Output:

```
This is in display method
This is in show method
This is in callme method
This is in call method
```

(iii) Third form of interface implementation

```
interface A
{
	void display();
}
class B implements A
{
	public void display()
	{
			System.out.println("This is in B's method");
	}
}
class C implements A
{
	public void display()
	{
			System.out.println("This is C's method");
	}
}
class interfacedemo
{
```

```
        public static void main(String args[])
        {
                B b1=new B();
                C c1=new C();
                b1.display();
                c1.display();
        }
}
```

Output:

```
This is in B's method
This is C's method
```

(iv) Fourth form of interface implementation

```
interface A
{
        void display();
}
interface B
{
        void callme();
}
interface C extends A,B
{
        void call();
}
class D implements C
{
        public void display()
        {
                System.out.println("interface A");
        }
        public void callme()
        {
                System.out.println("interface B");
        }
        public void call()
        {
                System.out.println("interface C");
         }
}
class interfacedemo
{
         public static void main(String args[])
         {
                D d1=new D();
                d1.display();
                d1.callme();
                d1.call();
         }
}
```

Output:

```
interface A
interface B
interface C
```

Exercise - 7 (Exception)

a) Exception handling mechanism

Aim: To write a JAVA program that describes exception handling mechanism

Program:

Usage of Exception Handling:

```
class trydemo
{
	public static void main(String args[])
	{
		try
		{
			int a=10,b=0;
			int c=a/b;
			System.out.println(c);
		}
		catch(ArithmeticException e)
		{
			System.out.println(e);
		}
		System.out.println("After the catch statement");
	}
}
```

Output:

```
java.lang.ArithmeticException: / by zero
After the catch statement
```

b) Illustrating multiple catch classes

Program:

Aim: To write a JAVA program Illustrating Multiple catch clauses

```
class multitrydemo
{
	public static void main(String args[])
	{
		try
		{
			int a=10,b=5;
			int c=a/b;
			 int d[]={0,1};
			System.out.println(d[10]);
			System.out.println(c);
		}
		catch(ArithmeticException e)
		{
			System.out.println(e);
		}
		catch(ArrayIndexOutOfBoundsException e)
		{
			System.out.println(e);
		}
		System.out.println("After the catch statement");
	}
}
```

Output:

```
java.lang.ArrayIndexOutOfBoundsException: 10
After the catch statement
```

Exercise – 8 (Runtime Polymorphism)

a)Runtime Polymorphism

Program:

Aim: To write a JAVA program that implements Runtime polymorphism

```
class A
{
        void display()
        {
                System.out.println("Inside A class");
        }
}
class B extends A
{
        void display()
        {
                System.out.println("Inside B class");
        }
}
class C extends A
{
        void display()
        {
                System.out.println("Inside C class");
        }
}
class runtimedemo
{
       public static void main(String args[])
       {
                A a1=new A();
                B b1=new B();
                C c1=new C();
                A ref;
                ref=c1;
                ref.display();
                ref=b1;
                ref.display();
                ref=a1;
                ref.display();
      }
}
```

Output:

```
Inside C class
Inside B class
Inside A class
```

b)Case study on Runtime Polymorphism

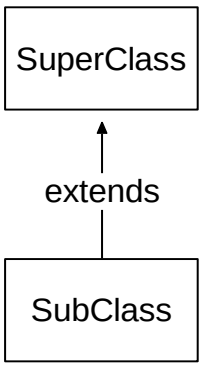
Aim: To write a Case study on run time polymorphism, inheritance that implements in above problem

Dynamic method dispatch is the mechanism by which a call to an overridden method is resolved at run time, rather than compile time.

- When an overridden method is called through a superclass reference, Java determines which version(superclass/subclasses) of that method is to be executed based upon the type of the object being referred to at the time the call occurs. Thus, this determination is made at run time.
- At run-time, it depends on the type of the object being referred to (not the type of the reference variable) that determines which version of an overridden method will be executed
- A superclass reference variable can refer to a subclass object. This is also known as upcasting. Java uses this fact to resolve calls to overridden methods at run time.

Upcasting

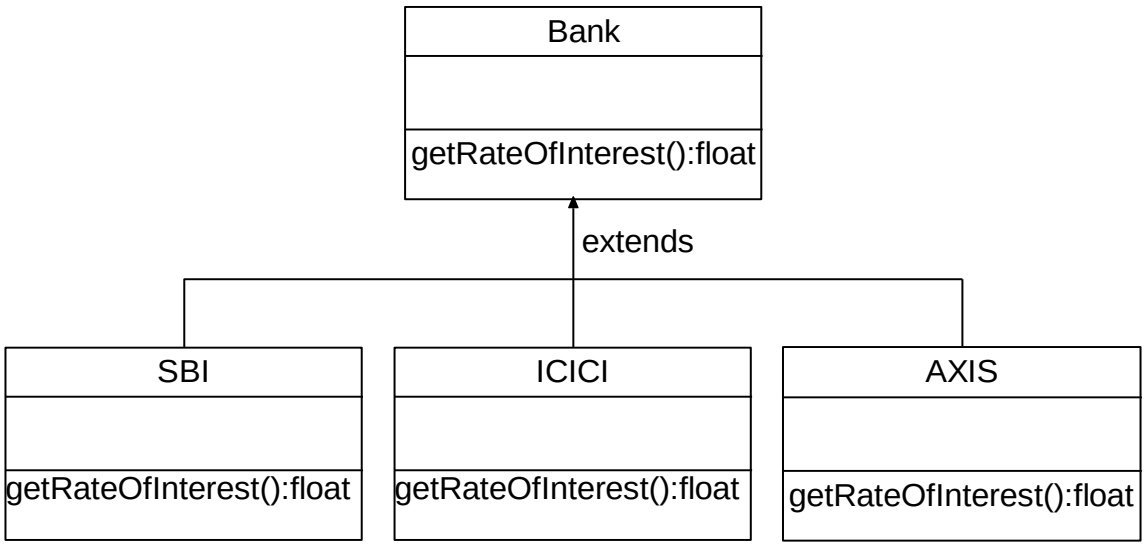
SuperClass obj=new SubClass



Therefore, if a superclass contains a method that is overridden by a subclass, then when different types of objects are referred to through a superclass reference variable, different versions of the method are executed. Here is an example that illustrates dynamic method dispatch:

The example is given by

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



Exercise – 9 (User defined Exception)

a)creation of illustrating throw

Program:

Aim: To write a JAVA program for creation of Illustrating throw

```
class throwdemo
{
        public static void main(String args[])
        {
                try
                {
                        throw new NullPointerException("demo");
                }
                catch(NullPointerException e)
                {
                        System.out.println(e);
                }
        }
}
```

Output:

```
java.lang.NullPointerException: demo
```

b)creation of illustrating finally

Aim: To write a JAVA program for creation of Illustrating finally

Program(i):

```
class finallydemo
{
	public static void main(String args[])
	{
		try
		{
			int a=10,b=0;
			int c=a/b;
			System.out.println(c);
		}
		catch(ArithmeticException e)
		{
			System.out.println(e);
		}
		finally
		{
			System.out.println("This is inside finally block");
		}
	}
}
```

Output:

```
java.lang.ArithmeticException: / by zero
This is inside finally block
```

Program(ii):

```
class finallydemo
{
	public static void main(String args[])
	{
		try
		{
			int a=10,b=5;
			int c=a/b;
			System.out.println(c);
		}
		catch(ArithmeticException e)
		{
			System.out.println(e);
		}
		finally
		{
			System.out.println("This is inside finally block");
		}
	}
}
```

Output:

```
2
This is inside finally block
```

c)creation of Java Built-in-Exceptions

Aim: To write a JAVA program for creation of Java Built-in Exceptions

Programs:

(i) Arithmetic exception

```
class arithmeticdemo
{
	public static void main(String args[])
	{
		try
		{
			int a = 10, b = 0;
			int c = a/b;
			System.out.println (c);
		}
		catch(ArithmeticException e)
		{
			System.out.println (e);
		}
	}
}
```

Output:
java.lang.ArithmeticException: / by zero

(ii)NullPointer Exception

```
class   nullpointerdemo
{
	public static void main(String args[])
	{
		try
		{
			String a = null;
			System.out.println(a.charAt(0));
		}
		catch(NullPointerException e)
		{
			System.out.println(e);
		}
	}
}
```

Output:
java.lang.NullPointerException

(iii)StringIndexOutOfBound Exception

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

```
                String a = "This is like chipping ";
                char c = a.charAt(24);
                System.out.println(c);
        }
        catch(StringIndexOutOfBoundsException e)
        {
                System.out.println(e);
        }
    }
}
```

Output:

java.lang.StringIndexOutOfBoundsException: String index out of range: 24

(iv)FileNotFound Exception

```
import java.io.*;
class filenotfounddemo
{
    public static void main(String args[])
    {
        try
        {
            File file = new File("E://file.txt");
            FileReader fr = new FileReader(file);
        }
        catch (FileNotFoundException e)
        {
            System.out.println(e);
        }
    }
}
```

Output:

java.io.FileNotFoundException: E:\file.txt (The system cannot find the file specified)

(v)NumberFormat Exception

```
class  numberformatdemo
{
    public static void main(String args[])
    {
        try
        {
            int num = Integer.parseInt ("akki") ;
            System.out.println(num);
        }
        catch(NumberFormatException e)
        {
            System.out.println(e);
        }
    }
}
```

Output:

java.lang.NumberFormatException: For input string: "akki"

(vi)ArrayIndexOutOfBounds Exception

```
class arraybounddemo
{
	public static void main(String args[])
	{
		try
		{
			int a[] = new int[5];
			a[6] = 9;
		}
		catch(ArrayIndexOutOfBoundsException e)
		{
			System.out.println (e);
		}
	}
}
```

Output:

```
java.lang.ArrayIndexOutOfBoundsException: 6
```

d)creation of User Defined Exception

Aim: To write a JAVA program for creation of User Defined Exception

Program:

```
class A extends Exception
{
        A(String s1)
        {
                super(s1);
        }
}
class owndemo
{
        public static void main(String args[])
        {
                try
                {
                        throw new A("demo ");
                }
                catch(Exception e)
                {
                        System.out.println(e);
                }
        }
}
```

Output:

A: demo

Exercise – 10 (Threads)

a)Extending Thread class

Aim: To write a JAVA program that creates threads by extending Thread class .First thread display "Good Morning "every 1 sec, the second thread displays "Hello "every 2 seconds and the third display "Welcome" every 3 seconds ,(Repeat the same by implementing Runnable)

Programs:

(i)Creating multiple threads using Thread class

```
class A extends Thread
{
	public void run()
	{
		try
		{
			for(int i=1;i<=10;i++)
			{
				sleep(1000);
				System.out.println("good morning");
			}
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
	}
}
class B extends Thread
{
	public void run()
	{
		try
		{
			for(int j=1;j<=10;j++)
			{
				sleep(2000);
				System.out.println("hello");
			}
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
	}
}
class C extends Thread
{
	public void run()
	{
		try
		{
```

```
                        for(int k=1;k<=10;k++)
                        {
                                sleep(3000);
                                System.out.println("welcome");
                        }
                }
                catch(Exception e)
                {
                        System.out.println(e);
                }
        }
}
class threaddemo
{
        public static void main(String args[])
        {
                A a1=new A();
                B b1=new B();
                C c1=new C();
                a1.start();
                b1.start();
                c1.start();
        }
}
```

Output:

```
good morning
hello
good morning
good morning
welcome
hello
good morning
good morning
hello
good morning
welcome
good morning
hello
good morning
good morning
welcome
hello
good morning
hello
welcome
hello
welcome
hello
hello
welcome
hello
welcome
welcome
welcome
welcome
```

(ii)Creating multiple threads using Runnable interface

```
class A implements Runnable
{
        public void run()
        {
                try
                {
                        for(int i=1;i<=10;i++)
                        {
                                Thread.sleep(1000);
                                System.out.println("good morning");
                        }
                }
                catch(Exception e)
                {
                        System.out.println(e);
                 }
        }
}
class B implements Runnable
{
        public void run()
        {
                try
                {
                        for(int j=1;j<=10;j++)
                        {
                                Thread.sleep(2000);
                                System.out.println("hello");
                        }
                }
                catch(Exception e)
                {
                        System.out.println(e);
                }
       }
}
class C implements Runnable
{
       public void run()
       {
                try
                {
                        for(int k=1;k<=10;k++)
                        {
                                Thread.sleep(3000);
                                System.out.println("welcome");
                        }
                }
                catch(Exception e)
                {
                        System.out.println(e);
                }
```

```
        }
}
class runnabledemo
{
        public static void main(String args[])
        {
                A a1=new A();
                B b1=new B();
                C c1=new C();
                Thread t1=new Thread(a1);
                Thread t2=new Thread(b1);
                Thread t3=new Thread(c1);
                 t1.start();
                 t2.start();
                 t3.start();
       }
}
```

Output:

```
good morning
good morning
hello
good morning
welcome
good morning
hello
good morning
good morning
welcome
hello
good morning
good morning
hello
good morning
welcome
good morning
hello
welcome
hello
hello
welcome
hello
welcome
hello
hello
welcome
welcome
welcome
welcome
```

(b)Implementing isAlive() and join()

Aim: To write a program illustrating isAlive and join ()

Program:

```
class A extends Thread
{
	public void run()
	{
		try
		{
			for(int i=1;i<=10;i++)
			{
				sleep(1000);
				System.out.println("good morning");
			}
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
	}
}
class B extends Thread
{
	public void run()
	{
		try
		{
			for(int j=1;j<=10;j++)
			{
				sleep(2000);
				System.out.println("hello");
			}
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
	}
}
class C extends Thread
{
	public void run()
	{
		try
		{
			for(int k=1;k<=10;k++)
			{
				sleep(3000);
				System.out.println("welcome");
			}
```

```
                }
                catch(Exception e)
                {
                        System.out.println(e);
                }
        }
}
class isalivedemo
{
        public static void main(String args[])
        {
                A a1=new A();
                B b1=new B();
                C c1=new C();
                a1.start();
                b1.start();
                c1.start();
                System.out.println(a1.isAlive());
                System.out.println(b1.isAlive());
                System.out.println(c1.isAlive());
                try
                {
                        a1.join();
                        b1.join();
                        c1.join();
                }
                catch(InterruptedException e)
                {
                        System.out.println(e);
                }
                System.out.println(a1.isAlive());
                System.out.println(b1.isAlive());
                System.out.println(c1.isAlive());
        }
}
```

Output:

```
true
true
true
good morning
good morning
hello
good morning
welcome
good morning
hello
good morning
good morning
welcome
hello
good morning
good morning
hello
good morning
welcome
good morning
hello
hello
welcome
hello
welcome
hello
welcome
hello
hello
welcome
welcome
welcome
welcome
false
false
false
```

c) Implementation of Daemon Threads

Aim: To write a Program illustrating Daemon Threads

Program:

```
class A extends Thread
{
	public void run()
	{
		if(Thread.currentThread().isDaemon())
		System.out.println("daemon thread work");
		else
		System.out.println("user thread work");
	}
}
class daemondemo
{
	public static void main(String[] args)
	{
		A a1=new A();
		A a2=new A();
		A a3=new A();
		a1.setDaemon(true);
		a1.start();
		a2.start();
		a3.start();
	 }
}
```

Output:

```
daemon thread work
user thread work
user thread work
```

Exercise - 11 (Threads continuity)

a)Producer-Consumer problem

Aim: Write a JAVA program Producer Consumer Problem

Program:

```
class A
{
	int n;
	boolean b=false;
	synchronized int get()
	{
		if(!b)
		try
		{
			wait();
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
		System.out.println("Got:"+n);
		b=false;
		notify();
		return n;
	}
	synchronized void put(int n)
	{
		if(b)
		try
		{
			wait();
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
		this.n=n;
		b=true;
		System.out.println("Put:"+n);
		notify();
	}
}
class producer implements Runnable
{
	A a1;
	Thread t1;
	producer(A a1)
	{
		this.a1=a1;
		t1=new Thread(this);
		t1.start();
	}
	public void run()
	{
```

```
                for(int i=1;i<=10;i++)
                {
                        a1.put(i);
                }
        }
}

class consumer implements Runnable
{
        A a1;
        Thread t1;
        consumer(A a1)
        {
                this.a1=a1;
                t1=new Thread(this);
                t1.start();
        }
        public void run()
        {
                for(int j=1;j<=10;j++)
                {
                        a1.get();
                }
        }
}
class interdemo
{
       public static void main(String args[])
       {
                A a1=new A();
                producer p1=new producer(a1);
                consumer c1=new consumer(a1);
       }
}
```

Output:

```
Put:1
Got:1
Put:2
Got:2
Put:3
Got:3
Put:4
Got:4
Put:5
Got:5
Put:6
Got:6
Put:7
Got:7
Put:8
Got:8
Put:9
Got:9
Put:10
Got:10
```

b)Case study on thread synchronization

Aim: To write a case study on thread Synchronization after solving the above producer consumer problem

A case study on thread synchronization after solving producer consumer problem:

- We can use wait, notify and notifyAll methods to communicate between threads in Java.
- For example, if we have two threads running in your program e.g.Producer and Consumer then producer thread can communicate to the consumer that it can start consuming now because there are items to consume in the queue.
- Similarly, a consumer thread can tell the producer that it can also start putting items now because there is some space in the queue, which is created as a result of consumption.
- A thread can use wait() method to pause and do nothing depending upon some condition.
- For example, in the producer-consumer problem, producer thread should wait if the queue is full and consumer thread should wait if the queue is empty.
- If some thread is waiting for some condition to become true, we can use notify and notifyAll methods to inform them that condition is now changed and they can wake up.
- Both notify() and notifyAll() method sends a notification but notify sends the notification to only one of the waiting thread, no guarantee which thread will receive notification and notifyAll() sends the notification to all threads.

Things to remember:

1. We can use wait() and notify() method to implement inter-thread communication in Java. Not just one or two threads but multiple threads can communicate to each other by using these methods.
2. Always call wait(), notify() and notifyAll() methods from synchronized method or synchronized block otherwise JVM will throw IllegalMonitorStateException.
3. Always call wait and notify method from a loop and never from if() block, because loop test waiting condition before and after sleeping and handles notification even if waiting for the condition is not changed.
4. Always call wait in shared object e.g. shared queue in this example.
5. Prefer notifyAll() over notify() method due to reasons given in this article

Exercise – 12 (Packages)

a) Illustration of class path

Aim: To write a JAVA program illustrate class path

```
import java.net.URL;
import java.net.URLClassLoader;
public class App
{
	public static void main(String[] args)
	{
		ClassLoader sysClassLoader = ClassLoader.getSystemClassLoader();
		URL[] urls = ((URLClassLoader)sysClassLoader).getURLs();
		for(int i=0; i< urls.length; i++)
		{
			System.out.println(urls[i].getFile());
		}
	}
}
```

Output:

```
E:/java%20work/
```

b) A case study on including in class path in os environment

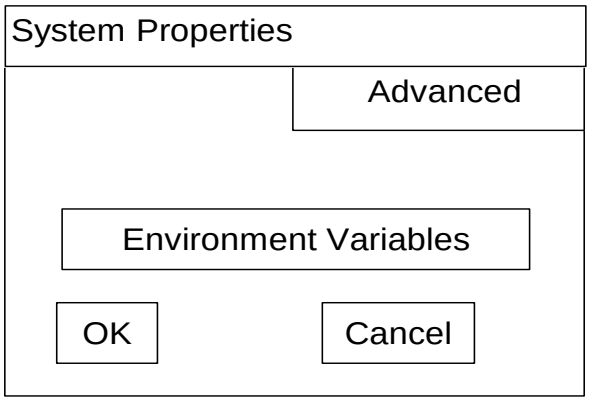
Aim: To write a case study on including in class path in your os environment of your package.

- The differences between path and classpath are given by.
  (1) The PATH is an environment variable used to locate "java" or "javac" command, to run java program and compile java source file. The CLASSPATH is an environment variable used to set path for java classes.
  (2) In order to set PATH in Java, we need to include **bin** directory in PATH environment while in order to set CLASSPATH we need to include all directories where we have put either our .class file or JAR file, which is required by our Java application.
  (3) PATH environment variable is used by operating system while CLASSPATH is used by Java ClassLoaders to load class files.
  (4) Path refers to the system while classpath refers to the Developing Environment.
- By default the java run time system uses the current working directory.
- Normally to execute a java program in any directory we have to set the path by as follows
  set path= c:\Program Files\java\jdk1.5.0_10\bin;

Setting environmental variable in windows xp:

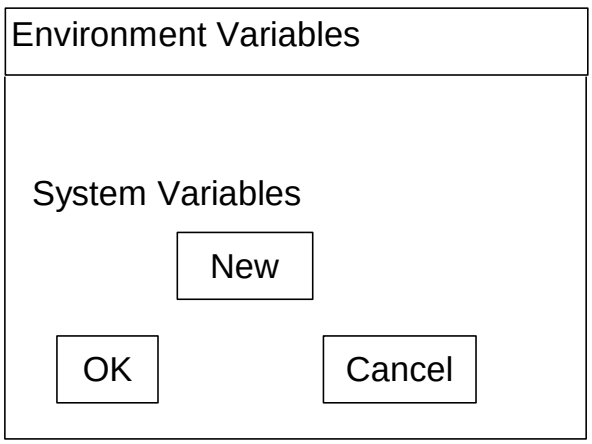
Step-1:

- Select My computer on the desktop and right click the mouse and then select properties.
- It displays the following “System Properties” dialog.



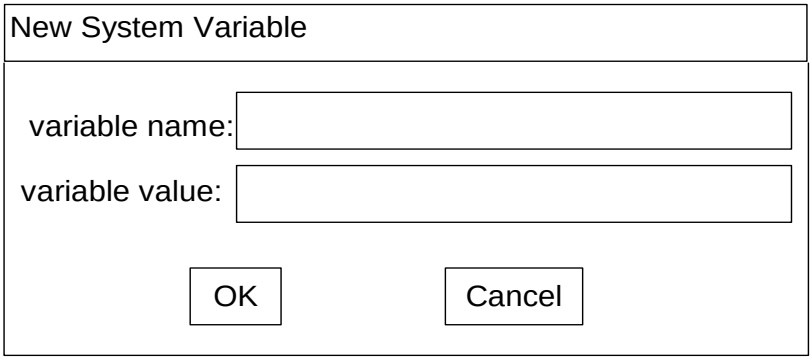
Step-2:

- In System Properties click Advanced and then click Environment Variables.
- It displays the following “Environment Variables” dialog.



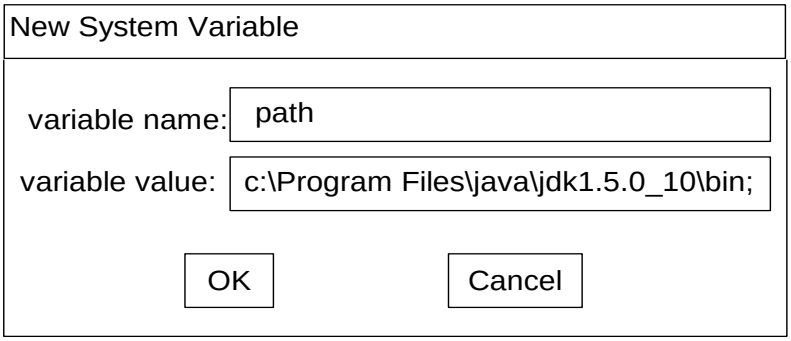
Step-3:

- In Environment Variables click New in System variables.
- It displays the following “New System Variable” dialog box.



Step-4:

- Now type variable name as a path and then variable value as c:\Program Files\java\jdk1.5.0_10\bin;



Step-5:

- Click OK

c) Creating and importing a package

Aim: To write a JAVA program that import and use the defined your package in the previous Problem

(i) Creating a package:

Steps:

1. First declare the name of the package using package keyword
   Example: package mypack;
2. Type the following program under this package statement. In package : class ,data, methods all are **public**

```
package mypack;
public class box
{
    public int l=10,b=20;
    public void display()
    {
        System.out.println(l);
        System.out.println(b);
    }
}
```

3. Create sub directory with a name same that of package name under the current working directory by as follows. d:\>md mypack
4. Under this subdirectory store the above program with a file name “box.java”.

(ii) importing a package:

Steps:

1. packages can be accessed by using the import statement
   General form: import pack1[.pack2].(classname/*);
   Example: import java.io.*;
   Here pack1 is name of top level package and pack2 is name of sub package
2. Type the following program under the current working directory and save the program with a file name “example.java”.

```
import mypack.box;
class packagedemo
{
    public static void main(String args[])
    {
        box b1=new box();
        b1.display();
    }
}
```

3. Now compile the above program in the current working directory d:\
   javac packagedemo.java
4. Execute the above program in current working directory
   java packagedemo

Output:

10
20

Exercise - 13 (Applet)

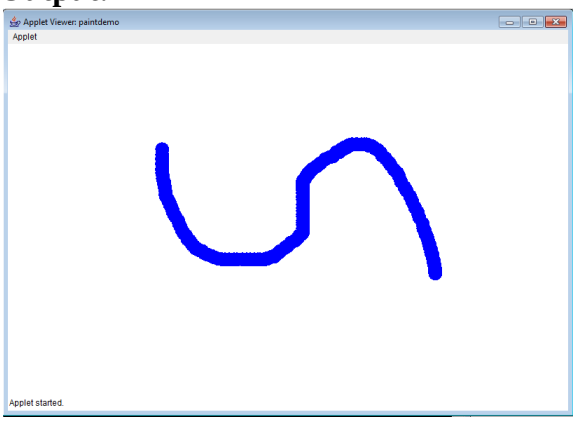
a) Paint like Paint Brush in Applet

Aim: To write a JAVA program to paint like paint brush in applet.

Program:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
 //<applet code="paintdemo"  width="800" height="500"></applet>
public class paintdemo extends Applet implements MouseMotionListener
{
        int w, h;
        Image i;
        Graphics g1;
        public void init()
        {
                w = getSize().width;  h = getSize().height;
                i = createImage( w, h );
                g1 = i.getGraphics();
                g1.setColor( Color.white ); g1.fillRect( 0, 0, w, h ); g1.setColor( Color.red );
                i = createImage( w, h );
                g1 = i.getGraphics();
                g1.setColor( Color.white ); g1.fillRect( 0, 0, w, h ); g1.setColor( Color.blue );
                addMouseMotionListener( this );
        }
        public void mouseMoved( MouseEvent e ) { }
        public void mouseDragged( MouseEvent me )
        {
                int x = me.getX(); int y = me.getY();
                g1.fillOval(x-10,y-10,20,20);
                repaint();
                me.consume();
        }
        public void update( Graphics g )
        {
                g.drawImage( i, 0, 0, this );
        }
        public void paint( Graphics g )
        {
                update(g);
        }
}
```

Output:



b) Display Analog Clock using Applet

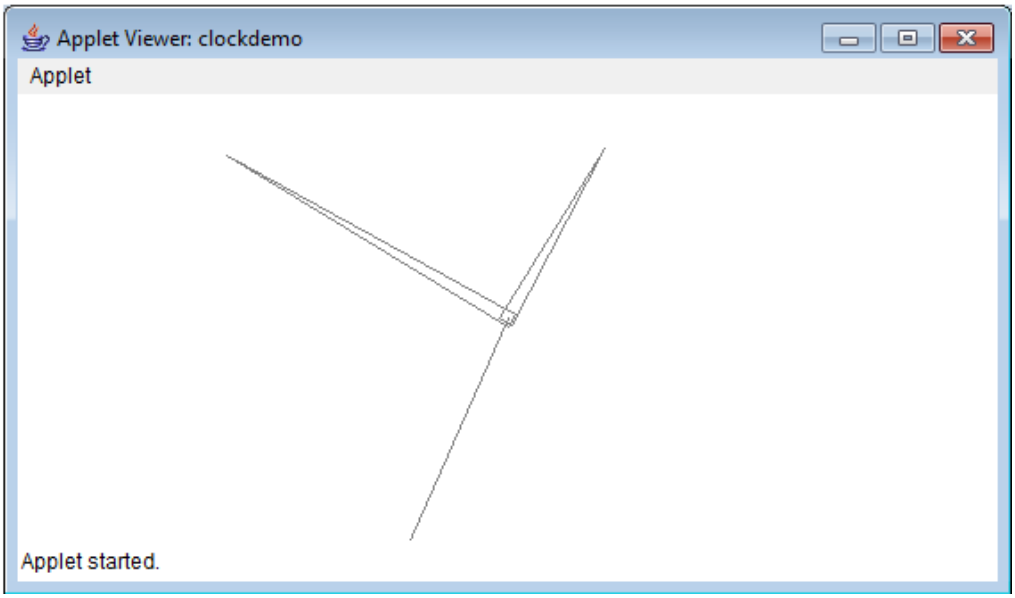
Aim: To write a JAVA program to display analog clock using Applet.

Program:

```
import java.util.*;
import java.text.*;
import java.applet.*;
import java.awt.*;
//<applet code="clockdemo" width="550" height="250"></applet
public class clockdemo extends Applet implements Runnable
{
	int h=0, m=0, s=0;
	String str=""; int wt, ht;  Thread thr=null;  boolean b;
	public void init()
	{
		wt=getSize().width; ht=getSize().height;
	}
	public void start()
	{
		if (thr==null)
		{
			thr=new Thread(this);
			b=false;
			thr.start();
		}
		else
		{
			if(b)
			{
				b=false;
				synchronized(this)
				{
					notify();
				}
			}
		}
	}
	public void stop()
	{
		b=true;
	}
	public void run()
	{
		try
		{
			while(true)
			{
				Calendar clndr=Calendar.getInstance();
				h=clndr.get(Calendar.HOUR_OF_DAY);
				if(h>12)h-=12;
				m=clndr.get(Calendar.MINUTE); s=clndr.get(Calendar.SECOND);
				SimpleDateFormat frmatter=new SimpleDateFormat("hh:mm:ss",
                                                    Locale.getDefault());
				Date d=clndr.getTime();  str=frmatter.format(d);
				if(b)
				{
					synchronized (this)
					{
```

```
                                        while(b)
                                        {
                                                wait();
                                        }
                                }
                        }
                        repaint();
                        thr.sleep(1000);
                }
        }
        catch(Exception e)
        {
                System.out.println(e);
        }
    }
    void drawHand(double angle, int radius, Graphics grp)
    {
        angle-=0.5*Math.PI;
        int a=(int)(radius*Math.cos(angle)); int b=(int)(radius*Math.sin(angle));
        grp.drawLine(wt/2,ht/2,wt/2+a,ht/2+b);
    }
    void drawWedge(double angle,int radius, Graphics grp)
    {
        angle-=0.5*Math.PI;
        int a=(int)(radius*Math.cos(angle)); int b=(int)(radius*Math.sin(angle));
        angle+=2*Math.PI/3;
        int a2=(int)(5*Math.cos(angle)); int b2=(int)(5*Math.sin(angle));
        angle+=2*Math.PI/3;
        int a3=(int)(5*Math.cos(angle)); int b3=(int)(5*Math.sin(angle));
        grp.drawLine(wt/2+a2, ht/2+b2,wt/2+a,ht/2+b);
        grp.drawLine(wt/2+a3, ht/2+b3,wt/2+a,ht/2+b);
        grp.drawLine(wt/2+a2, ht/2+b2,wt/2+a3,ht/2+b3);
     }
     public void paint(Graphics grp)
     {
        grp.setColor(Color.gray);
        drawWedge(2*Math.PI*h/12,wt/5,grp);  drawWedge(2*Math.PI*m/60,wt/3,grp);
        drawHand(2*Math.PI*s/60,wt/2,grp);
     }
}
```

Output:



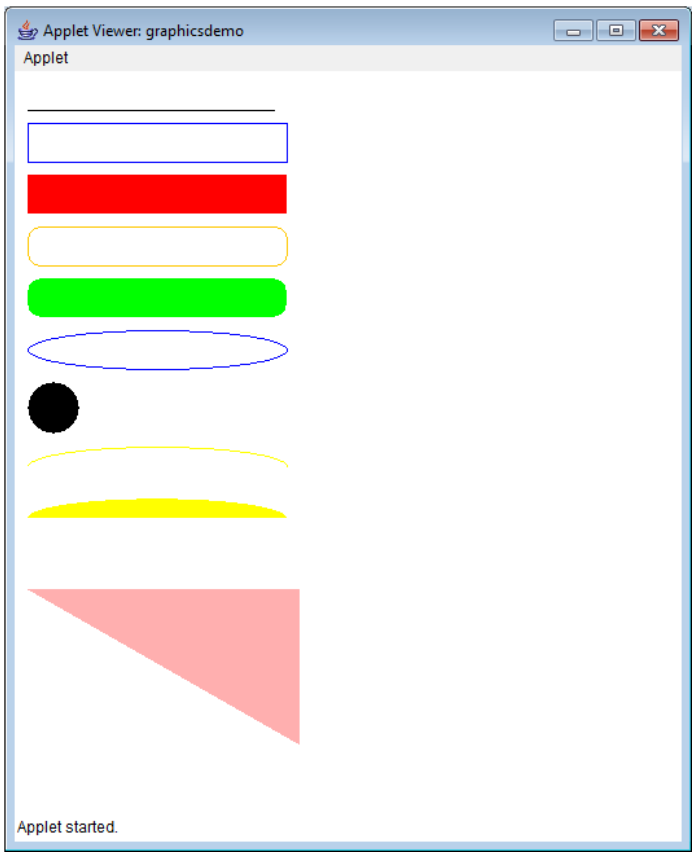
c) Display Analog Clock using Applet

Aim: To write a JAVA program to create different shapes and fill colors using Applet

Program:

```
import java.awt.*;
import java.applet.*;
//<applet code="graphicsdemo" width="400" height="400"></applet>
public class graphicsdemo extends Applet
{
        public void paint(Graphics g)
        {
                int x[]={10,220,220};
                int y[]={400,400,520};
                int n=3;
                g.drawLine(10,30,200,30);
                g.setColor(Color.blue);          g.drawRect(10,40,200,30);
                g.setColor(Color.red);           g.fillRect(10,80,200,30);
                g.setColor(Color.orange);    g.drawRoundRect(10,120,200,30,20,20);
                g.setColor(Color.green);     g.fillRoundRect(10,160,200,30,20,20);
                g.setColor(Color.blue);          g.drawOval(10,200,200,30);
                g.setColor(Color.black);         g.fillOval(10,240,40,40);
                g.setColor(Color.yellow);    g.drawArc(10,290,200,30,0,180);
                g.setColor(Color.yellow);    g.fillArc(10,330,200,30,0,180);
                g.setColor(Color.pink);          g.fillPolygon(x,y,n);
        }
}
```

Output:



Exercise - 14 (Event Handling)

a) Cursor movement using mouse

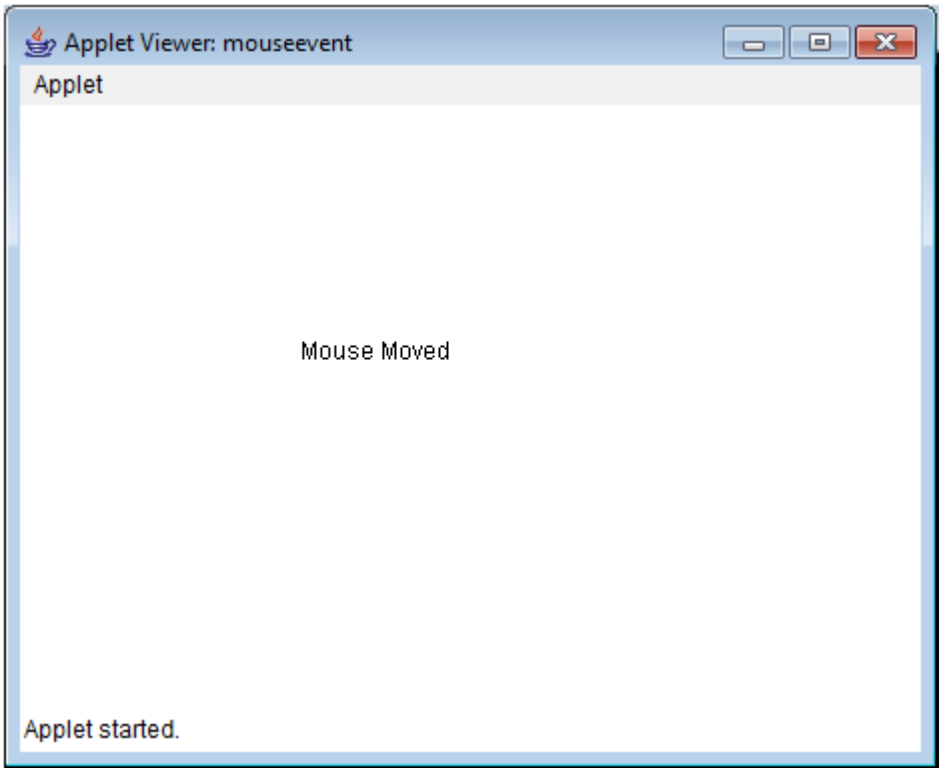
Aim: To write a JAVA program that display the x and y position of the cursor movement using Mouse.

Program:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
//<applet code="mouseevent"  width=450 height=300></applet>
public class mouseevent extends Applet
implements MouseListener, MouseMotionListener
{
	String s1=" ";
	int x,y;
	public void init()
	{
		addMouseListener(this);
		addMouseMotionListener(this);
	}
	public void mouseClicked(MouseEvent me)
	{
		x=100;
		y=100;
		s1="Mouse clicked";
		repaint();
	}
	public void mouseEntered(MouseEvent me)
	{
		x=100;
		y=200;
		s1="Mouse entered";
		repaint();
	}
	public void mouseExited(MouseEvent me)
	{
		x=100;
		y=300;
		s1="Mouse exited";
		repaint();
	}
	public void mousePressed(MouseEvent me)
	{
		x=me.getX();
		y=me.getY();
		s1="Mouse Pressed";
		repaint();
	}
	public void mouseReleased(MouseEvent me)
	{
		x=me.getX();
		y=me.getY();
		s1="Mouse Realeased";
```

```
            repaint();
        }
        public void mouseDragged(MouseEvent me)
        {
            x=me.getX();
            y=me.getY();
            s1="Mouse Dragged";
            repaint();
       }
       public void mouseMoved(MouseEvent me)
       {
            x=me.getX();
            y=me.getY();
            s1="Mouse Moved";
            repaint();
       }
       public void paint(Graphics g)
       {
            g.drawString(s1,x,y);
       }
}
```

Output:



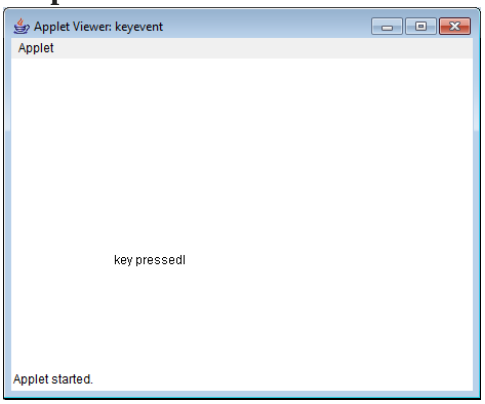
b) Key-up and Key-down event

Aim: To write a JAVA program that identifies key-up key-down event user entering text in a Applet.

Program:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
//<applet code="keyevent" width=450 height=300></applet>
public class keyevent extends Applet implements KeyListener
{
	String s1=" ";
	int x,y;
	public void init()
	{
		addKeyListener(this);
		requestFocus();
	}
	public void keyPressed(KeyEvent ke)
	{
		x=100;
		y=200;
		s1= "key pressed ";
		repaint();
	}
	public void keyReleased(KeyEvent ke)
	{
		x=100;
		y=400;
		s1= "key Released ";
		repaint();
	}
	public void keyTyped(KeyEvent ke)
	{
		s1=s1+ke.getKeyChar();
		repaint();
	}
	public void paint(Graphics g)
	{
		g.drawString(s1,x,y);
	}
}
```

Output:



Exercise - 15 (Swings)

a) Building a calculator in swings

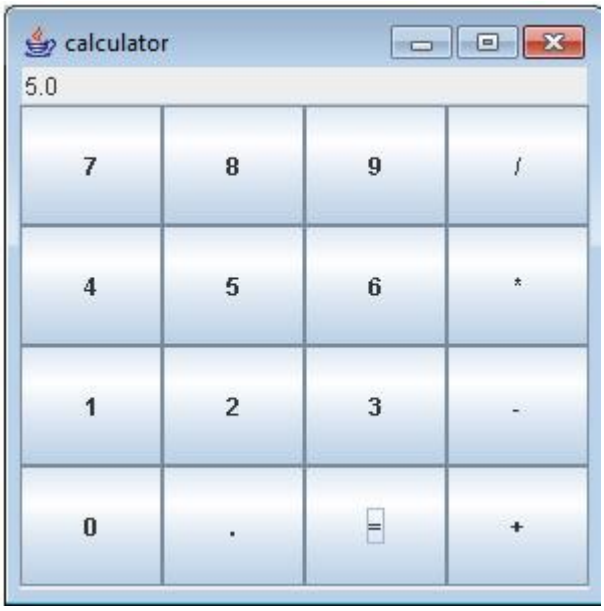
Aim: To write a JAVA program to build a Calculator in Swings

Program:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class calculator extends JPanel  implements ActionListener
{
        JTextField jt = new JTextField();
        double d= 0;
        String op = "=";
        boolean b1 = true;
        calculator()
        {
                setLayout(new BorderLayout());
                jt.setEditable(false);
                add(jt, "North");
                JPanel jp = new JPanel();
                jp.setLayout(new GridLayout(4, 4));
                String s1 = "789/456*123-0.=+";
                for (int i = 0; i < s1.length(); i++)
                {
                        JButton b = new JButton(s1.substring(i, i + 1));
                        jp.add(b);
                        b.addActionListener(this);
                }
                add(jp, "Center");
        }
        public void actionPerformed(ActionEvent ae)
        {
                String s1 = ae.getActionCommand();
                if ('0' <= s1.charAt(0) && s1.charAt(0) <= '9' || s1.equals("."))
                {
                        if (b1)
                        jt.setText(s1);
                        else
                        jt.setText(jt.getText() + s1);
                        b1 = false;
                }
                else
                {
                        if (b1)
                        {
                                if (s1.equals("-"))
                                {
                                        jt.setText(s1);
                                        b1 = false;
                                }
                                else
                                op = s1;
```

```
                        }
                        else
                        {
                                double x = Double.parseDouble(jt.getText());
                                calculate(x);
                                op = s1;
                                b1 = true;
                        }
                }
        }
        private void calculate(double n)
        {
                if (op.equals("+"))
                d += n;
                else if (op.equals("-"))
                d -= n;
                else if (op.equals("*"))
                d *= n;
                else if (op.equals("/"))
                d /= n;
                else if (op.equals("="))
                d = n;
                jt.setText("" + d);
         }
         public static void main(String[] args)
         {
                JFrame jf = new JFrame();
                jf.setTitle("calculator");
                jf.setSize(300, 300);
                Container c = jf.getContentPane();
                c.add(new calculator());
                jf.show();
        }
}
```

Output:



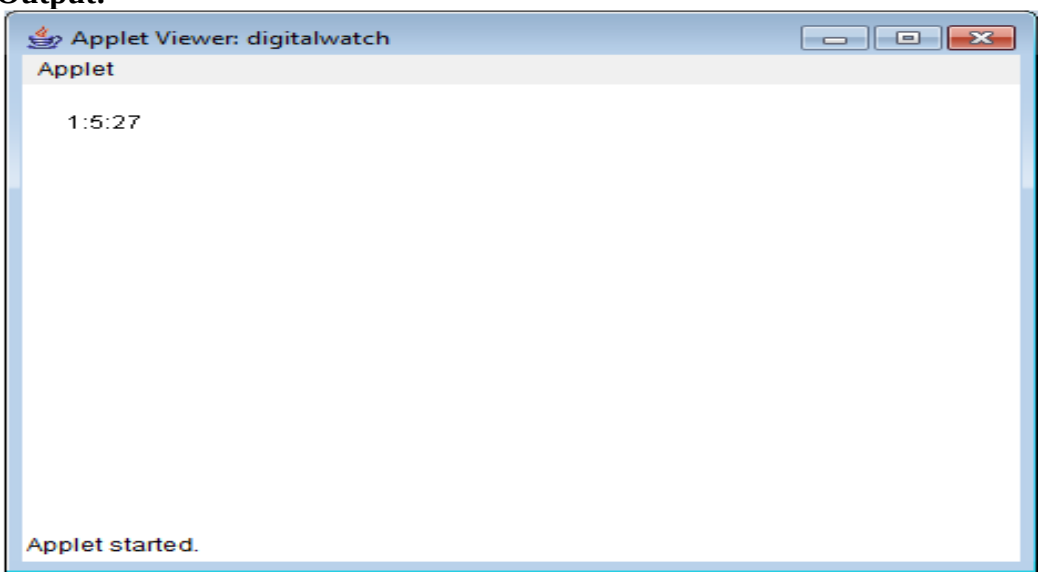
b) Displaying digital watch

Aim: To write a JAVA program to display the digital watch in swing tutorial

Program:

```
import java.awt.*;
import java.applet.*;
import java.util.*;
//<applet code="digitalwatch" width=450 height=300></applet>
public class digitalwatch extends Applet implements Runnable
{
	Thread t,t1;
	public void start()
	{
		t = new Thread(this);
		t.start();
	}
	public void run()
	{
		t1 = Thread.currentThread();
		while(t1 == t)
		{
			repaint();
			try
			{
				t1.sleep(1000);
			}
			catch(Exception e)
			{
				System.out.println(e);
			}
		}
	}
	public void paint(Graphics g)
	{
		Calendar cal = new GregorianCalendar();
		String h = String.valueOf(cal.get(Calendar.HOUR));
		String m = String.valueOf(cal.get(Calendar.MINUTE));
		String s = String.valueOf(cal.get(Calendar.SECOND));
		g.drawString(h + ":" + m + ":" + s, 20, 30);
	}
}
```

Output:



Exercise – 16 (Swings - Continued)

a)Ball bouncing inside a JPanel

Aim: To write a JAVA program that to create a single ball bouncing inside a JPanel.

Program:

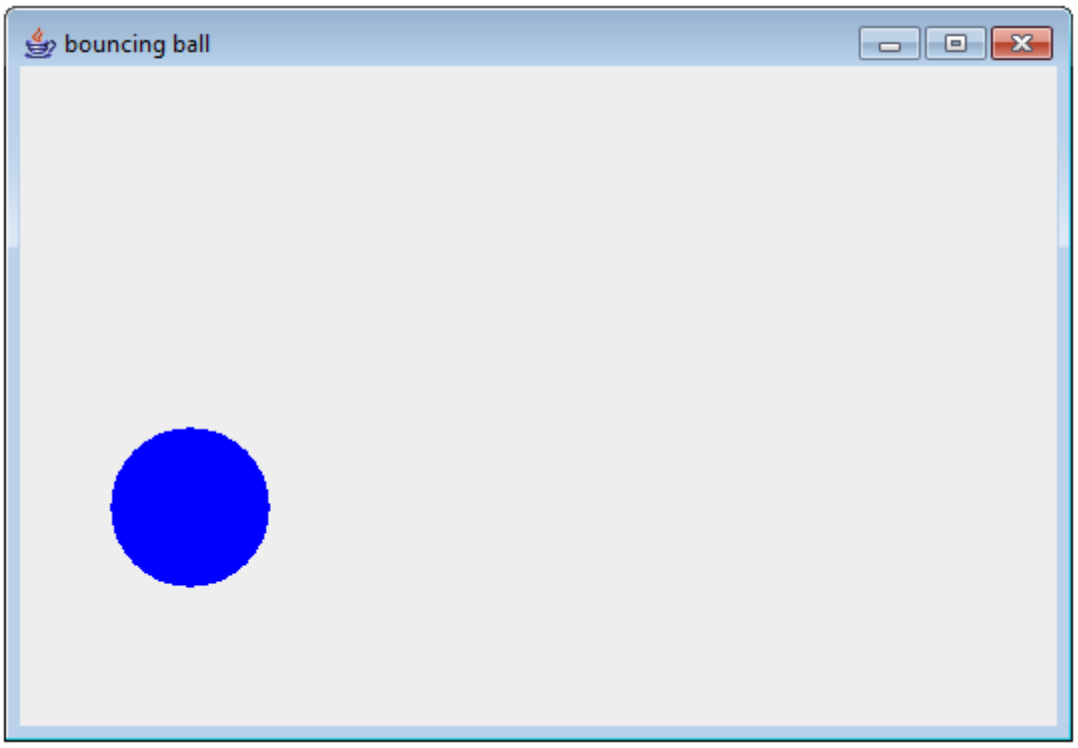
```
import java.awt.*;
import javax.swing.*;
public class bouncingball extends JPanel
{
	int w,h;
	float r = 40,d= r * 2,X = r + 50,Y = r + 20,dx = 3,dy = 3;
	public bouncingball()
	{
		Thread thread = new Thread()
		{
			public void run()
			{
				while (true)
				{
					w = getWidth();
					h = getHeight();
					X = X + dx ;
					Y = Y + dy;
					if (X - r < 0)
					{
						dx = -dx;
						X = r;
					}
					else if (X + r > w)
					{
						dx = -dx;
						X = w - r;
					}
					if (Y - r < 0)
					{
						dy = -dy;
						Y = r;
					}
					else if (Y + r > h)
					{
						dy = -dy;
						Y = h - r;
					}
					repaint();
					try
					{
						Thread.sleep(50);
					}

					catch (Exception e)
					{
```

```
                                        System.out.println(e);
                                    }

                                }
                            }
                        };
                        thread.start();
                }
                public void paintComponent(Graphics g)
                {
                        super.paintComponent(g);
                        g.setColor(Color.BLUE);
                        g.fillOval((int)(X-r), (int)(Y-r), (int)d, (int)d);
                }
                public static void main(String[] args)
                {
                        JFrame jf = new JFrame("bouncing ball");
                        jf.setSize(300, 200);
                        jf.setContentPane(new bouncingball());
                        jf.setVisible(true);
                }
}
```

Output:



b)Displaying a real tree

Aim: To write a JAVA program JTree as displaying a real tree upside down

Program:

```
import javax.swing.*;
import javax.swing.tree.*;
class realtree
{
        public static void main(String[] args)
        {
                JFrame jf = new JFrame();
                DefaultMutableTreeNode d1 = new DefaultMutableTreeNode("Color", true);
                DefaultMutableTreeNode d2 = new DefaultMutableTreeNode("Black");
                DefaultMutableTreeNode d3 = new DefaultMutableTreeNode("Blue");
                DefaultMutableTreeNode d4 = new DefaultMutableTreeNode("Navy Blue");
                DefaultMutableTreeNode d5 = new DefaultMutableTreeNode("Dark Blue");
               DefaultMutableTreeNode d6 = new DefaultMutableTreeNode("Green");
               DefaultMutableTreeNode d7 = new DefaultMutableTreeNode("White");
               d1.add(d2);
               d1.add(d3);
               d3.add(d4);
               d3.add(d5);
               d1.add(d6);
               d1.add(d7);
               JTree jt = new JTree(d1);
               jf.add(jt);
               jf.setSize(200,200);
               jf.setVisible(true);
        }
}
```

Output: