JAVA PROGRAMMING LAB PROGRAMS LIST

1. Write a JAVA program to display default value of all primitive data types of JAVA.

2. Write a JAVA program to display the Fibonacci sequence

3. Write a JAVA program give example for command line arguments.

4. Write a JAVA program to sort given list of numbers.

5. Write a JAVA program to search for an element in a given list of elements (linear search).

6. Write a JAVA program to search for an element in a given list of elements using binary search mechanism.

7. Write a JAVA program to determine multiplication of two matrices.

8. Write a JAVA program to sort an array of strings

9. Write a JAVA program to check whether given string is palindrome or not.

10. Write a JAVA program for call by value and call by reference.

11. Write a JAVA program to give the example for _this‘ operator. And also use the _this‘ keyword as return statement.

12. Write a JAVA program to demonstrate static variables, methods, and blocks.

13. Write a JAVA program using StringTokenizer class, which reads a line of integers and then displays each integer and the sum of all integers.

14. Write a JAVA program to give the example for _super‘ keyword.

15. Write a JAVA program that illustrates simple inheritance.

16. Write a JAVA program to maintain Student Grading Database using multilevel inheritance. Student is Super class, which contains roll no, name, address. Marks derived from Student class, which contains subject names and respective marks. Result is derived from Marks class, which contains total, grade.

17. Write a JAVA program demonstrating the difference between method overloading and method overriding.

18. Write a JAVA program demonstrating the difference between method overloading and constructor overloading.

19. Write a JAVA program that describes exception handling mechanism.

20. Write a JAVA program for example of try and catch block. In this check whether the given array size is negative or not.

21. Write a JAVA program to illustrate sub class exception precedence over base class.

22. Write a JAVA program for creation of user defined exception.

23. Write a JAVA program to illustrate creation of threads using runnable interface (start method start each of the newly created thread. Inside the run method there is sleep() for suspend the thread for 500 milliseconds).

24. Write a JAVA program to create a class MyThread in this class a constructor, call the base class constructor, using super and starts the thread. The run method of the class starts after this. It can be observed that both main thread and created child thread are executed concurrently.

25. Write a JAVA program illustrating multiple inheritance using interfaces.

26. Write a JAVA program to create a package named pl, and implement this package in Ex class.

27. Write a JAVA program to create a package named mypack and import it in Circle class. 28. Write a JAVA program to create an abstract class named Shape, that contains an empty method named numberOfSides ().Provide three classes named Trapezoid, Triangle and Hexagon, such that each one of the classes contains only the method numberOfSides (), that contains the number of sides in the given geometrical figure.

29. Write a JAVA program that describes the life cycle of an applet.

30. Write a JAVA program to create a border layout control.

31. Write a JAVA program to create a grid layout control.

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

Write a JAVA program that allows user to draw lines, rectangles and ovals

1. **Program Statement:**

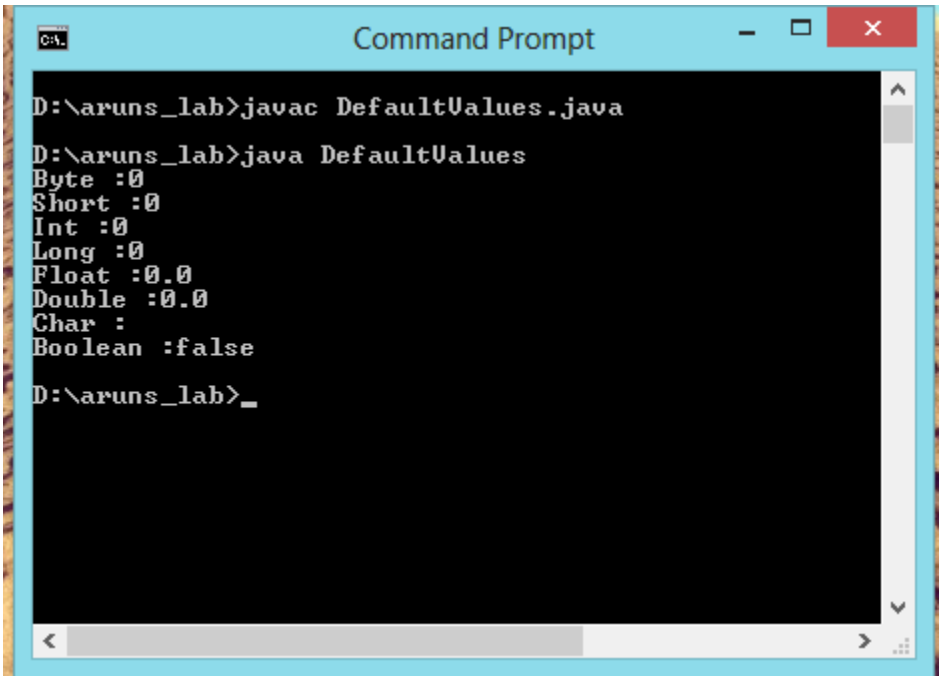
   Write a java program to display default value of all primitive data types of java

   Program:

```
class  DefaultValues
{
   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("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:



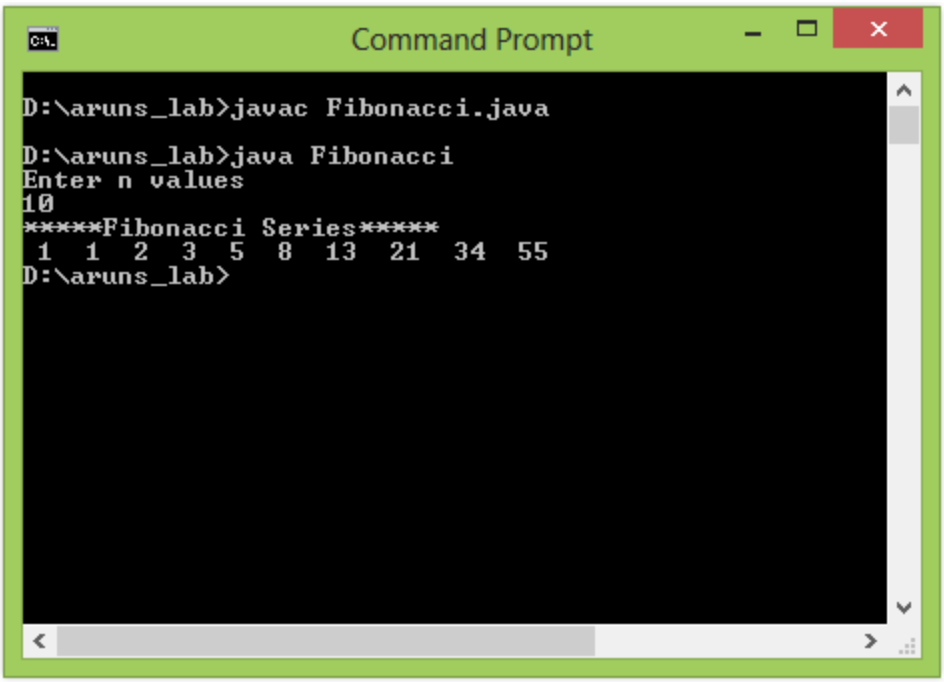
0. Program Statement:

Write a java program to display Fibonacci series of a given number

Program:

```
import java.io.*;
import java.util.*;
class Fibonacci
{
    public static void main(String args[])
{
     Scanner s=new Scanner(System.in);
     int n;
     System.out.println("Enter n values");
     n=s.nextInt();
     System.out.println("*****Fibonacci Series*****");
      int f1, f2=0, f3=1;
    for(int i=1;i<=n;i++)
{
    System.out.print(" "+f3+" ");
    f1 = f2;
    f2 = f3;
    f3 = f1 + f2;
            }
    }
}
```

Output:



0. Program Statement:

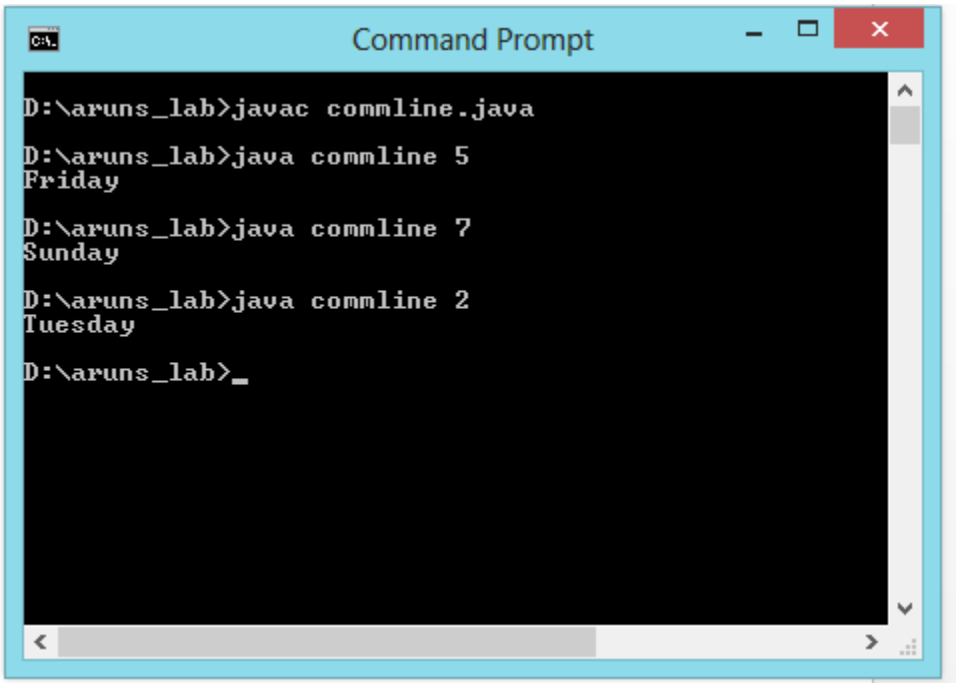
Write a java program for command line arguments

Program:

```
import java.io.*;
import java.util.*;
public class commline
{
public static void main(String key[])
   {

   int x=Integer.parseInt(key[0]);
   switch(x){
   case 1: System.out.println("Monday");
   break;
   case 2: System.out.println("Tuesday");
   break;
```

```
    case 3: System.out.println("Wednesday");
    break;
    case 4: System.out.println("Thursday");
    break;
    case 5: System.out.println("Friday");
    break;
    case 6: System.out.println("Saturday");
    break;
    case 7: System.out.println("Sunday");
    break;
    default :System.out.println("Invalid Number of Day");
       }
    }
}
```

Output:



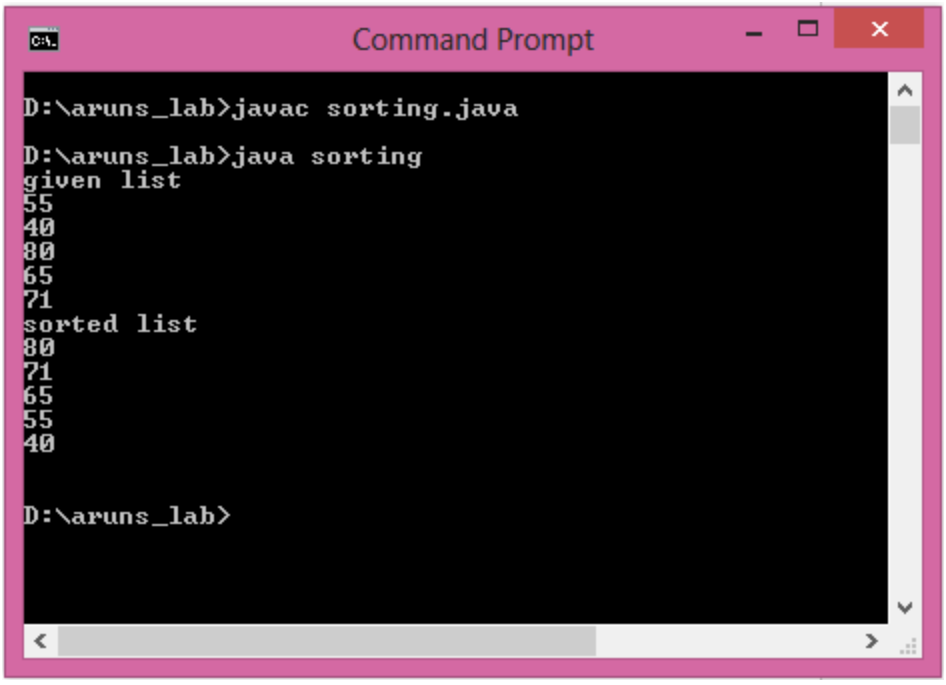
0. Program Statement:

Write a java program to sort list of numbers

Program:

```
import java.io.*;
import java.util.*;
class sorting
{
public static void main(String args[])
{
int number[]={55,40,80,65,71};
int n=number.length;
System.out.println("given list");
for(int i=0;i<n;i++)
{
System.out.println(""+number[i]);
}
for(int i=0;i<n;i++)
{
for(int j=i+1;j<n;j++)
{
if(number[i]<number[j])
{
int temp=number[i];
number[i]=number[j];
number[j]=temp;
}
}
}
System.out.println("sorted list");
for(int i=0;i<n;i++)
{
System.out.println(""+number[i]);
}
System.out.println("");
}
}
```

Output:



0. Program Statement:

Write a java program to search an element in a given list of numbers using linear search

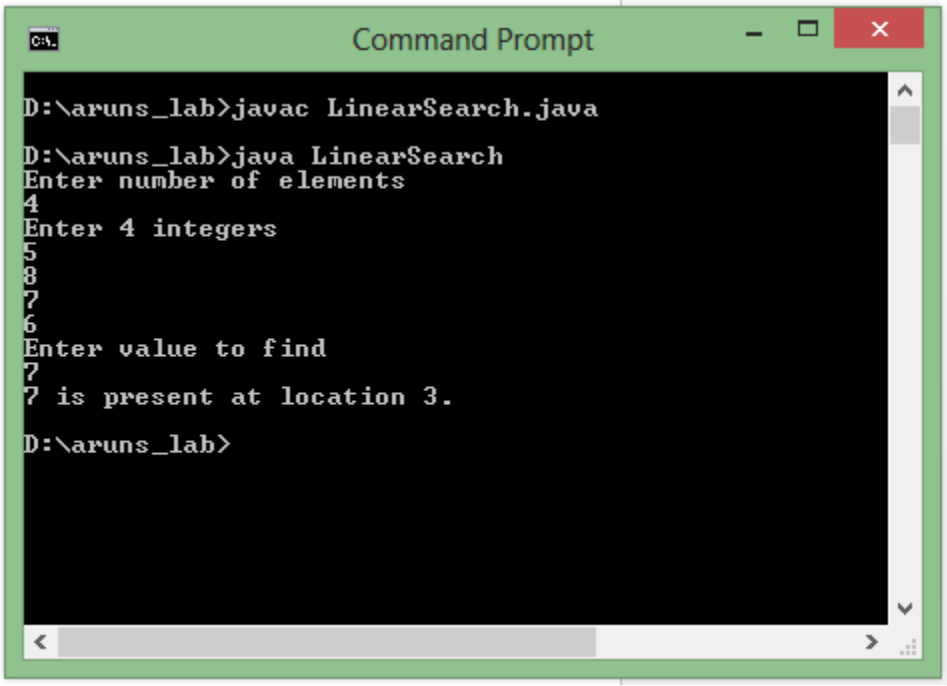
Linear search: To check if an element is present in the given list we compare search element with every element in the list. If the number is found then success occurs otherwise the list doesn't contain the element we are searching.

Program:

```
import java.util.Scanner;

class LinearSearch
{
 public static void main(String args[])
 {
  int c, n, search, array[];

  Scanner in = new Scanner(System.in);
  System.out.println("Enter number of elements");
  n = in.nextInt();
  array = new int[n];

  System.out.println("Enter " + n + " integers");

  for (c = 0; c < n; c++)
```

```
      array[c] = in.nextInt();

    System.out.println("Enter value to find");
    search = in.nextInt();

    for (c = 0; c < n; c++)
    {
      if (array[c] == search)     /* Searching element is present */
      {
        System.out.println(search + " is present at location " + (c + 1) + ".");
         break;
      }
    }
    if (c == n)  /* Searching element is absent */
      System.out.println(search + " is not present in array.");
  }
}
```

Output:



0. Program Statement:

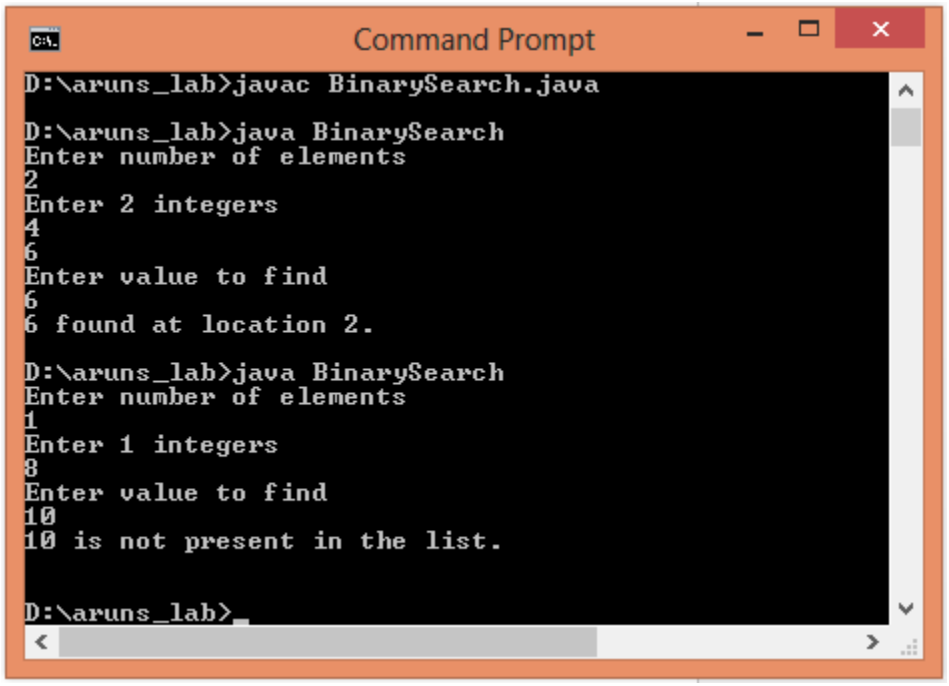
Write a java program to search an element in a given list of numbers using binary search

Binary search: This code implements binary search algorithm. Please note input numbers must be in *ascending* order.

Program:

```
import java.util.Scanner;

class BinarySearch
{
  public static void main(String args[])
  {
    int c, first, last, middle, n, search, array[];

    Scanner in = new Scanner(System.in);
    System.out.println("Enter number of elements");
    n = in.nextInt();
    array = new int[n];

    System.out.println("Enter " + n + " integers");


    for (c = 0; c < n; c++)
      array[c] = in.nextInt();

    System.out.println("Enter value to find");
    search = in.nextInt();

    first  = 0;
    last   = n - 1;
    middle = (first + last)/2;

    while( first <= last )
    {
      if ( array[middle] < search )
        first = middle + 1;
      else if ( array[middle] == search )
      {
        System.out.println(search + " found at location " + (middle + 1) + ".");
        break;
      }
      else
         last = middle - 1;

      middle = (first + last)/2;
    }
    if ( first > last )
      System.out.println(search + " is not present in the list.\n");
  }
}
```

Output:



0. Program Statement:

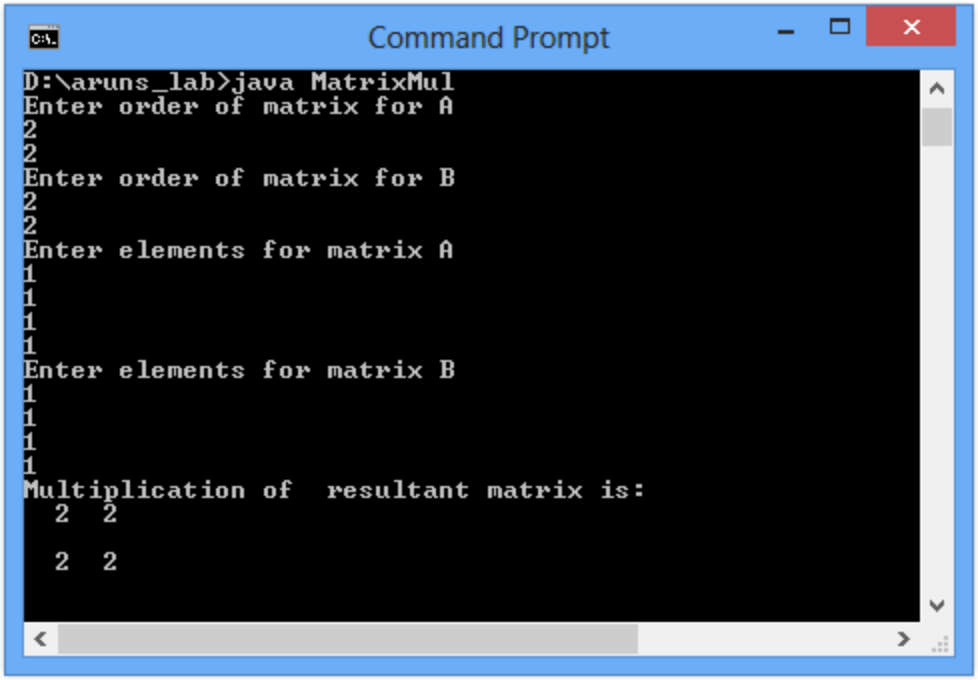
Write a Java Program to determine the Multiplication of two matrices.

Program:

```
import java.io.*;

import java.util.*;

    class MatrixMul
    {
     public static void main(String arg[])
     {
    int i=0,j=0,k=0;
    Scanner s=new Scanner(System.in);
    System.out.println("Enter order of matrix for A");
    int m=s.nextInt();
    int n=s.nextInt();
    System.out.println("Enter order of matrix for B");
    int p=s.nextInt();
    int q=s.nextInt();
    int a[][]=new int[10][10];
    int b[][]=new int[10][10];
    int c[][]=new int[10][10];


        if(n==p)
```

```
{
System.out.println("Enter elements for matrix A");
for(i=0;i<m;i++)
   for(j=0;j<n;j++)
a[i][j]= s.nextInt();

System.out.println("Enter elements for matrix B ");
  for(i=0;i<m;i++)
     for(j=0;j<n;j++)
b[i][j]=s.nextInt();

  for(i=0;i<m;i++)
  {
   for(j=0;j<q;j++)
    {
for(k=0;k<p;k++)


    c[i][j]=c[i][j]+a[i][k]*b[k][j];
   }
  }
System.out.print("Multiplication of  resultant matrix is:");
   for(i=0;i<m;i++)
  {
   for(j=0;j<q;j++)
    {
     System.out.print("  " +c[i][j]);
    }
     System.out.println("\n");
   }
 }
}
}
```

Output:



0. Program Statement:

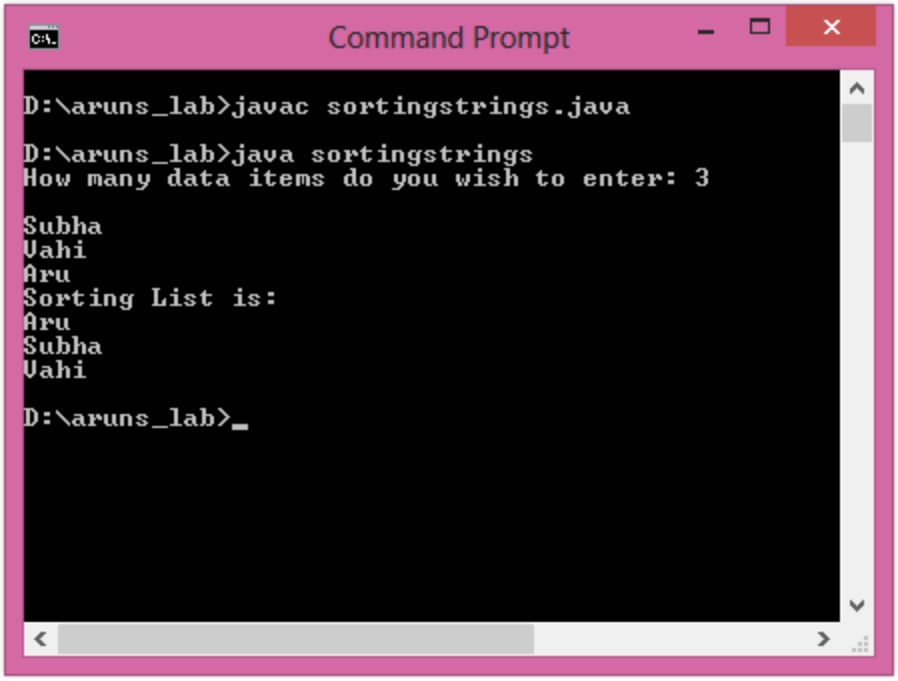
Write a Java Program to sort an array of Strings.

Program:

```
import java.util.*;
public class sortingstrings{
    public static void main(String[] argsv){
      Scanner s = new Scanner(System.in);
      System.out.print("How many data items do you wish to enter: ");
      int num = new Integer(s.nextLine()).intValue();
      String[] names = new String[num];
      System.out.println();
      int i = 0;
      while(i < num){
        names[i] = s.nextLine();
        i++;
```

```
        }
        StringComparator strcomp = new StringComparator();
        Arrays.sort(names, strcomp);
System.out.println("Sorting List is:");
        for (String name: names){
System.out.println(name);

        }
    }
  }

   class StringComparator implements Comparator<String>{
      public int compare(String a, String b){
        return a.compareTo(b);
    }
  }
```

Output:



0. Program Statement:

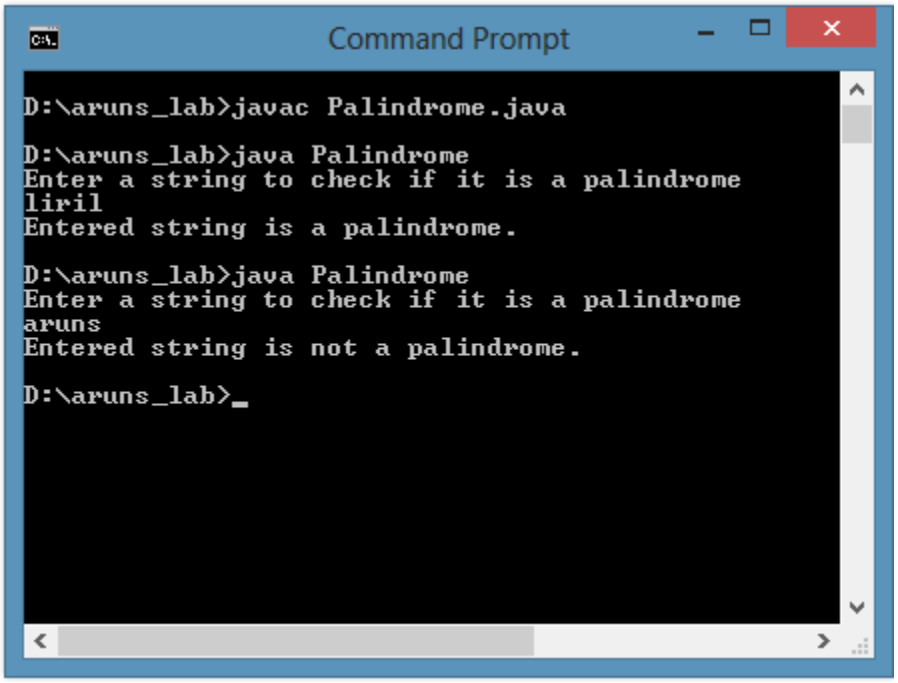
Write a Java Program to check whether the given string is palindrome or not.

Program:

```
import java.util.*;
public class Palindrome
{
  public static void main(String args[])
  {
    String original, reverse = "";
    Scanner in = new Scanner(System.in);
     System.out.println("Enter a string to check if it is a palindrome");
    original = in.nextLine();
     int length = original.length();
     for ( int i = length - 1; i >= 0; i-- )
       reverse = reverse + original.charAt(i);
     if (original.equals(reverse))
```

```
            System.out.println("Entered string is a palindrome.");
        else
            System.out.println("Entered string is not a palindrome.");
    }
}
```

Output:



0. Program Statement:

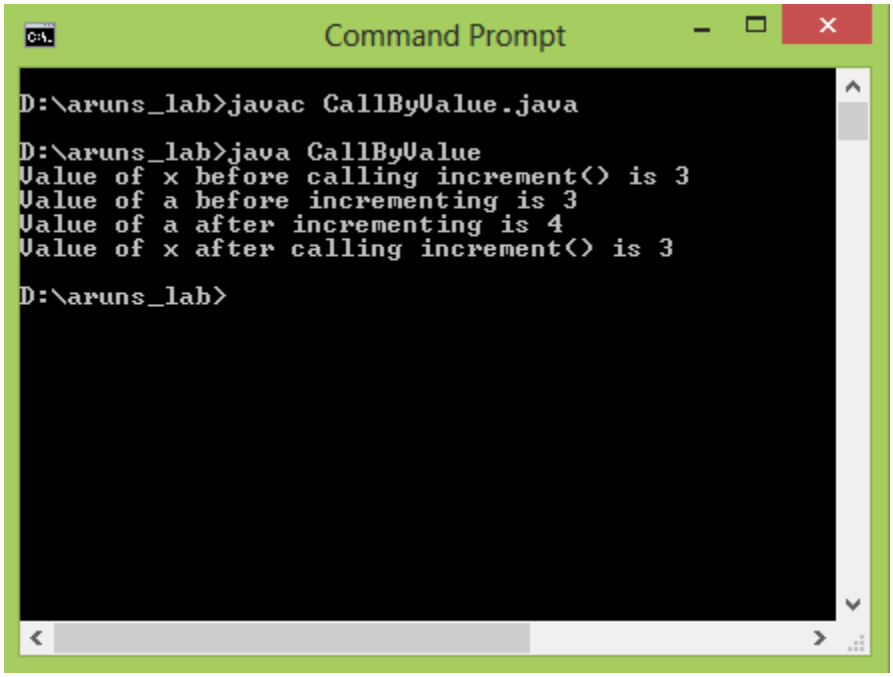
Write a Java Program for call by value and call by reference

Program: Call by Value

```
class CallByValue {

  public static void main ( String[] args ) {
    int x =3;
    System.out.println ( "Value of x before calling increment() is "+x);
    increment(x);
    System.out.println ( "Value of x after calling increment() is "+x);
```

```
  }

  public static void increment ( int a ) {
    System.out.println ( "Value of a before incrementing is "+a);
    a= a+1;
    System.out.println ( "Value of a after incrementing is "+a);
  }
}
```

Output:

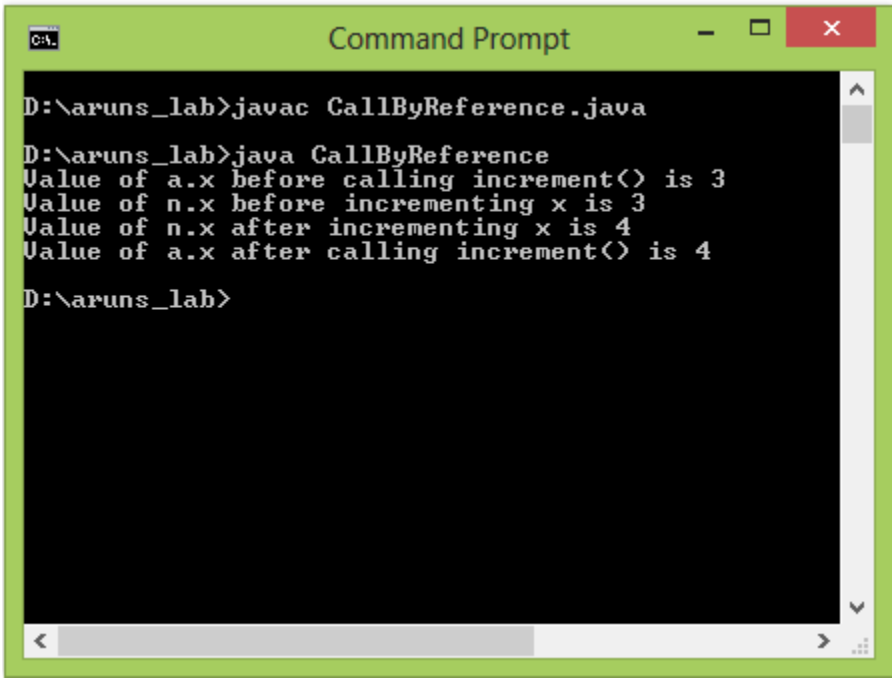


Program: Call by Reference

```
class Number {
  int x;
}

class CallByReference {

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

```
    Number a = new Number();
    a.x=3;
    System.out.println("Value of a.x before calling increment() is "+a.x);
    increment(a);
    System.out.println("Value of a.x after calling increment() is "+a.x);
  }

  public static void increment(Number n) {
    System.out.println("Value of n.x before incrementing x is "+n.x);
    n.x=n.x+1;
    System.out.println("Value of n.x after incrementing x is "+n.x);
  }
}
```

Output:



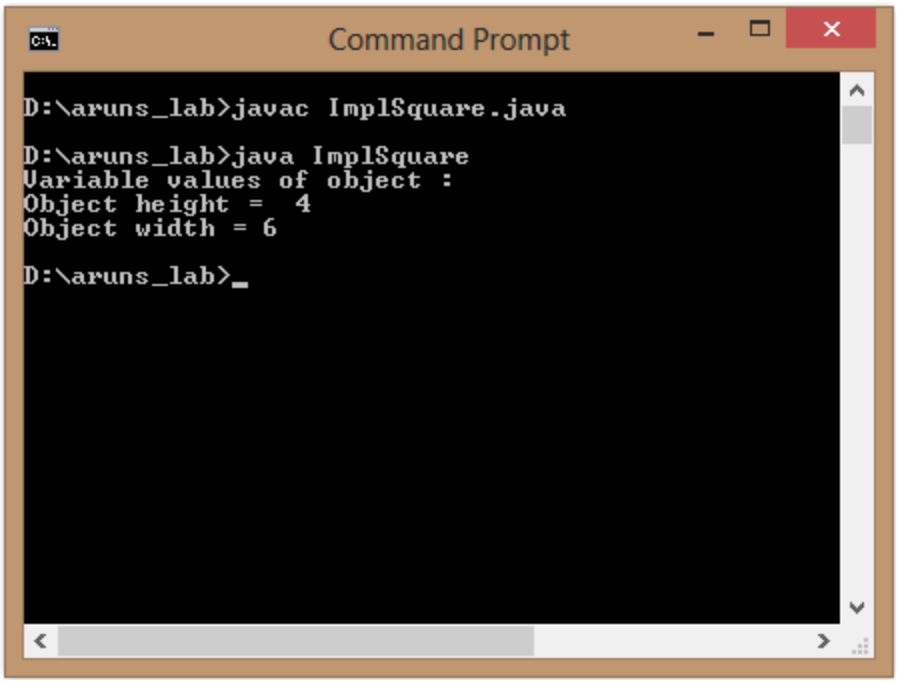
0. Program Statement

Write a Java Program for “this” keyword

Program:

```
class Square
{
      int height;
      int width;
 Square(int height, int width)
        {
             this.height = height;
             this.width = width;
         }
}

class ImplSquare
{
    public static void main(String args[])
    {
          Square sObj = new Square(4,6);
                    System.out.println("Variable values of object : ");
          System.out.println("Object height =  " + sObj.height);
          System.out.println("Object width = " + sObj.width);
     }
}
```

Output:



0. Program Statement:

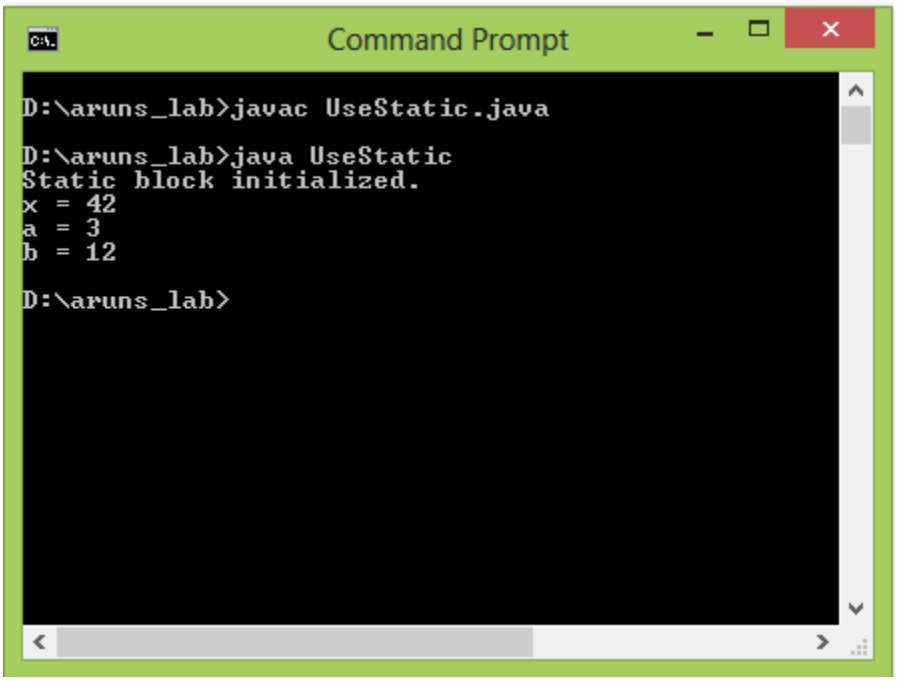
Write a Java Program to demonstrate static variables, methods and blocks

Program:

```
class UseStatic {
  static int a = 3;
  static int b;

  static void meth(int x) {
    System.out.println("x = " + x);
    System.out.println("a = " + a);
    System.out.println("b = " + b);
  }

  static {
    System.out.println("Static block initialized.");
    b = a * 4;
```

```
  }

  public static void main(String args[]) {
    meth(42);
  }
}
```

Output:



14.Program Statement:

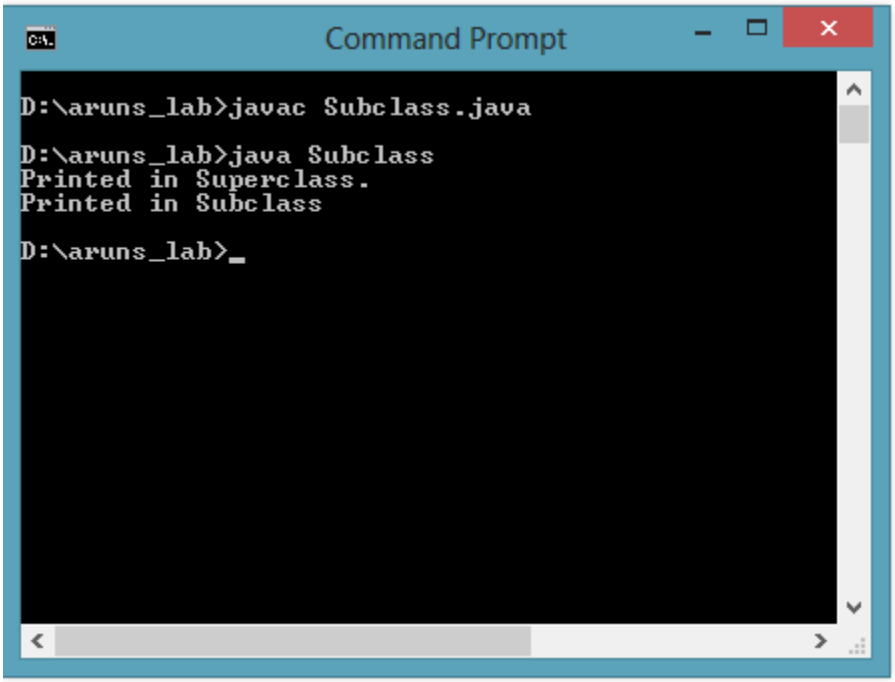
Write a Java Program to give the example for “super” keyword

Program:

```
class Superclass {

    public void printMethod() {
        System.out.println("Printed in Superclass.");
    }
```

```
}
//Here is a subclass, called Subclass, that overrides printMethod():

public class Subclass extends Superclass {

   // overrides printMethod in Superclass
   public void printMethod() {
      super.printMethod();
      System.out.println("Printed in Subclass");
   }
   public static void main(String[] args) {
      Subclass s = new Subclass();
      s.printMethod();
   }
}
```

Output:



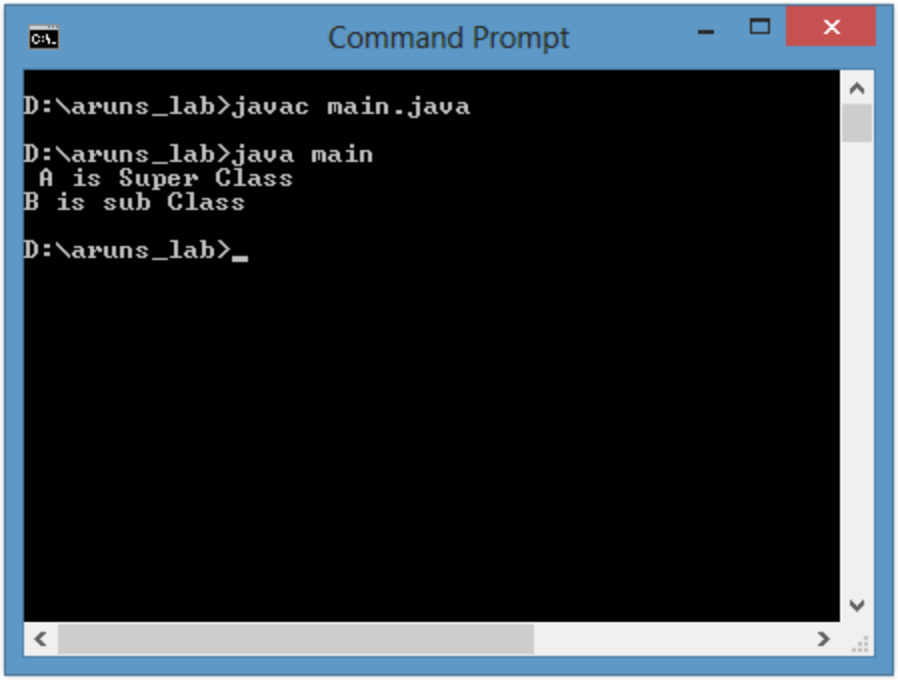
15.Program Statement:

Write a Java Program that illustrates simple inheritance

Program:

```
class A {

   public void show() {

       System.out.println(" A is Super Class");

   }

}

class B extends A {

   public void display() {

         System.out.println("B is sub Class");

   }

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

Output:



16.Program Statement:

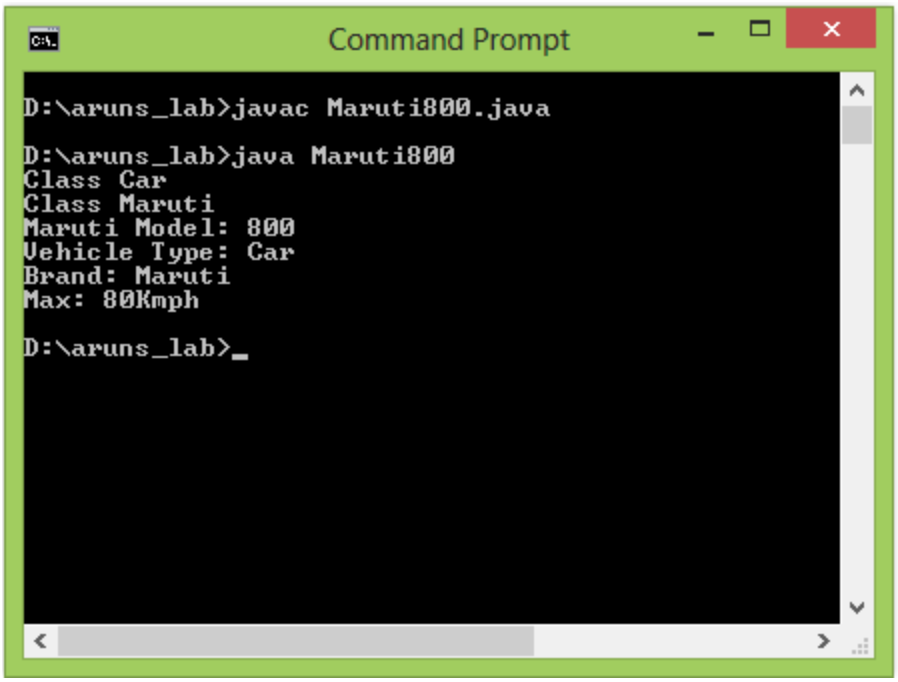
Write a Java Program that illustrates multi-level inheritance.

Program:

```
class Car{
        public Car()
        {
                System.out.println("Class Car");
        }
        public void vehicleType()
        {
                System.out.println("Vehicle Type: Car");
        }
}
class Maruti extends Car{
        public Maruti()
        {
                System.out.println("Class Maruti");
```

```
        }
        public void brand()
        {
                System.out.println("Brand: Maruti");
        }
        public void speed()
        {
                System.out.println("Max: 90Kmph");
        }
}
public class Maruti800 extends Maruti{

        public Maruti800()
        {
                System.out.println("Maruti Model: 800");
        }
        public void speed()
                {
                        System.out.println("Max: 80Kmph");
                }
        public static void main(String args[])
        {
                Maruti800 obj=new Maruti800();
                obj.vehicleType();
                obj.brand();
                obj.speed();
        }
}
```

Output:



17.Program Statement:

Write a Java Program demonstrating the difference between method overloading and method overriding.

Difference between Overloading and Overriding in Java

1) First and major difference between Overloading and Overriding is that former occur during compile time while later occur during runtime.

2) Second difference between Overloading and Overriding is that, you can overload method in same class but you can only override method in sub class.

3) Third difference is that you can overload static method in Java but you cannot override static method in Java. In fact when you declare same method in Sub Class it's known as method hiding because it hide super class method instead of overriding it.

4) Overloaded methods are bonded using static binding and Type of reference variable is used, while overridden method are bonded using dynamic bonding based upon actual Object.

5) Rules of Overloading and Overriding is different in Java. In order to overload a method you need to change its method signature but that is not required for overriding any method in Java.

6) Another difference between method overloading and overriding is that private and final method cannot be overridden but can be overloaded in Java.

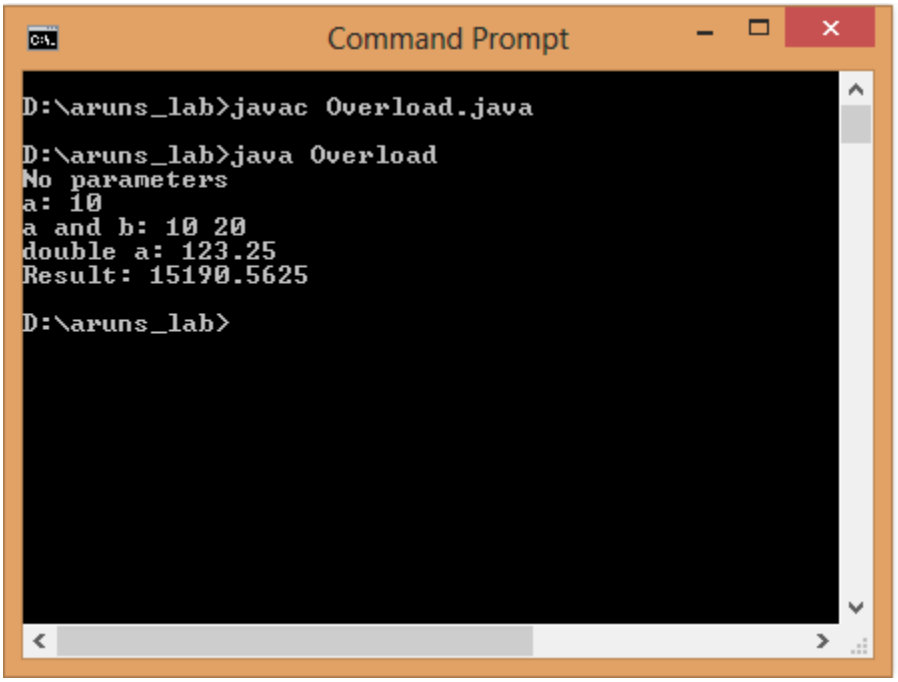
7) Overloaded method are fast as compare to Overridden method in Java.

Program: Method Overloading

```
      class  OverloadDemo
 {
void test()
{
System.out.println("No parameters");
}
// Overload test for one integer parameter.
void test(int a)
 {
System.out.println("a: " + a);
}
// Overload test for two integer parameters.
void test(int a, int b)
 {
System.out.println("a and b: " + a + " " + b);
}
// overload test for a double parameter
double test(double a)
 {
System.out.println("double a: " + a);
return a*a;
}
}
class Overload
{
public static void main(String args[])
{
OverloadDemo ob = new OverloadDemo();
double result;
// call all versions of test()
```

```
ob.test();
ob.test(10);
ob.test(10, 20);
result = ob.test(123.25);
System.out.println("Result: " + result);
}
}
```

Output:

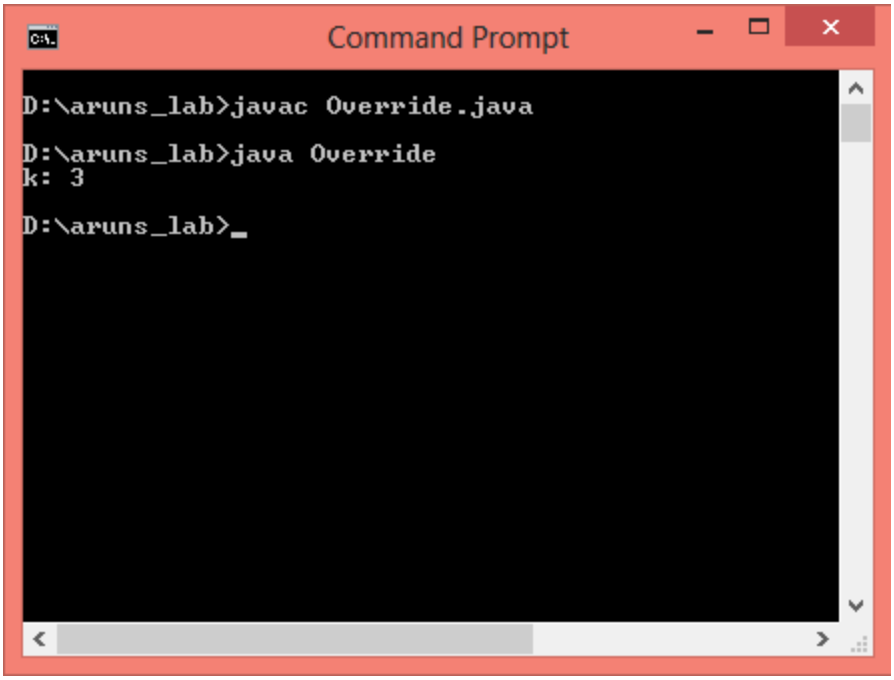


Program: Method Overriding

```
 class A
{
int i, j;
A(int a, int b) {
i = a;
j = b;
}
```

```
void show() // display i and j
{
System.out.println("i and j: " + i + " " + j);
}
}

class B extends A
{
int k;

B(int a, int b, int c)
{
super(a, b);
k = c;
}

void show()
{
System.out.println("k: " + k);
}
}

class Override {
public static void main(String args[])
{
B subOb = new B(1, 2, 3);

subOb.show(); // this calls show() in B
}
}
```

Output:



18.Program Statement:

Write a Java Program demonstrating the difference between method overloading and constructor overloading

In Java it is possible to define two or more methods within the same class that share the same name, as long as their parameter declarations are different , the methods are said to be *overloaded .This process is called* ***method overloading.***

Method overloading is one of the ways that Java supports polymorphism.

In Java it is possible to define two or more constructors within the same class like as class name that share the same name, as long as their parameter declarations are different , the constructors are said to be *overloaded .This process is called* ***constructor overloading***.

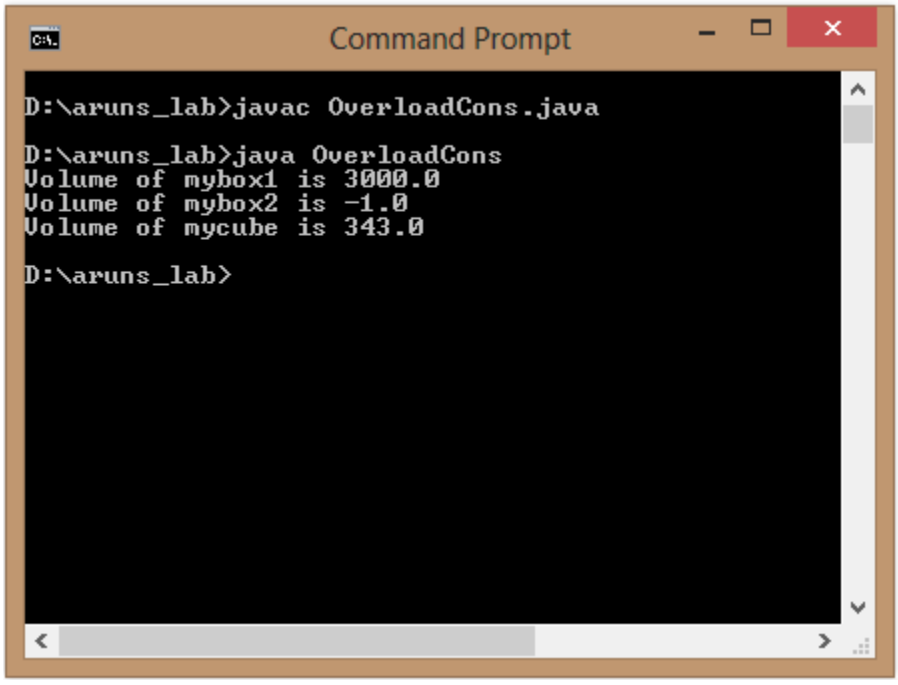
Program: Constructor Overloading

```
class Box
{
double width;
double height;
double depth;
// constructor used when all dimensions specified
Box(double w, double h, double d)
```

```
{
width = w;
height = h;
depth = d;
}
// constructor used when no dimensions specified
Box()
{
width = -1;
height = -1;
depth = -1;
}
// constructor used when cube is created
Box(double len)
{
width = height = depth = len;
}
// compute and return volume
double volume()
 {
return width * height * depth;
}
}
class OverloadCons
{
public static void main(String args[])
{
// create boxes using the various constructors
Box mybox1 = new Box(10, 20, 15);
Box mybox2 = new Box();
```

```
Box mycube = new Box(7);
double vol;
// get volume of first box
vol = mybox1.volume();
System.out.println("Volume of mybox1 is " + vol);
// get volume of second box
vol = mybox2.volume();
System.out.println("Volume of mybox2 is " + vol);
// get volume of cube
vol = mycube.volume();
System.out.println("Volume of mycube is " + vol);
}
}
```

Output:



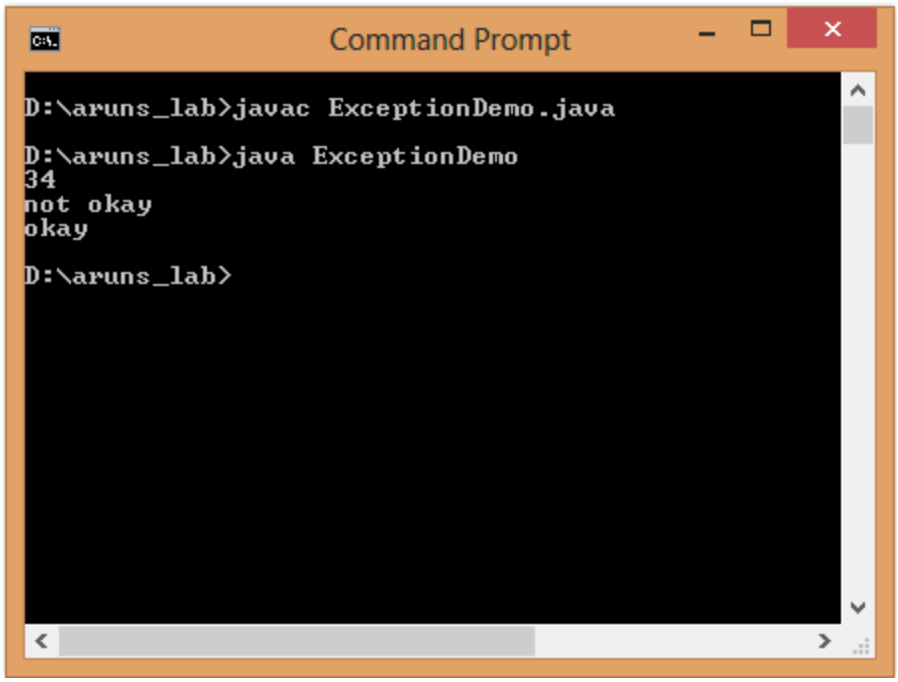
19.Program Statement:

Write a Java Program that describes the Exception Handling

Program:

```
import java.io.*;
import java.lang.*;
public class ExceptionDemo
{
  public static void main(String args[])throws IOException
  {
    int subject[]={12,23,34,21};
    try
    {
      System.out.println(subject[2]);
      System.out.println("not okay");
    }
    catch(ArrayIndexOutOfBoundsException e)
    {
     System.out.println("i caught the exception:"+e);
      throw e;
    }
   finally
   {
    System.out.println("okay");
   }
 }
}
```

Output:



20.Program Statement:

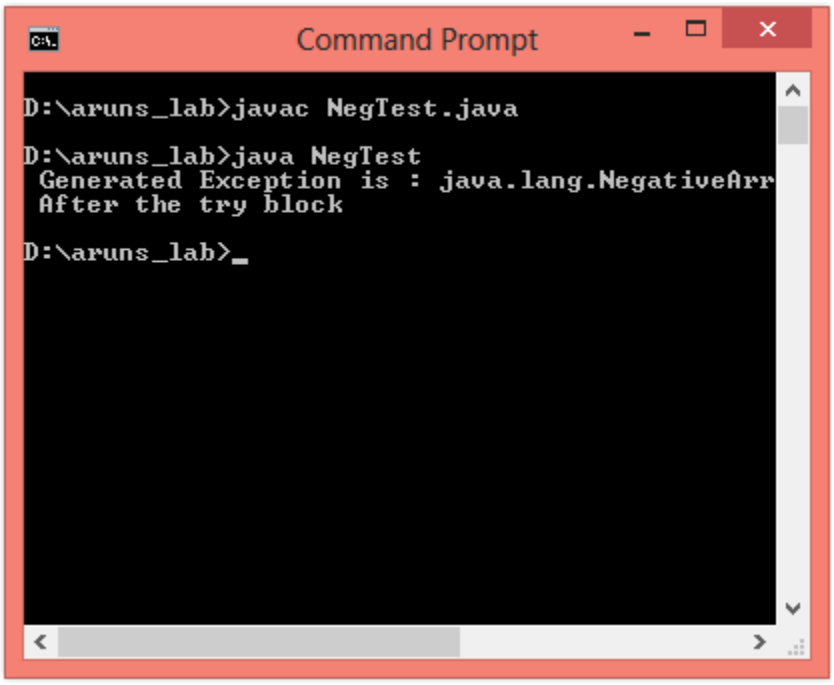
Write a Java Program for example of try and catch block.in this check whether the given array size is negative or not.

Program:

```
class NegTest
{
public static void main(String a[])
 {
try
 {
 int a1[] = new int[-2];
 System.out.println("first element : "+a1[0]);
 }
 catch(NegativeArraySizeException n)
 {
 System.out.println(" Generated Exception is : " + n);
 }
```

```
 System.out.println(" After the try block");
 }
}
```

Output:



0. Program Statement:

Write a Java Program to illustrate sub class exception precedence over base class.

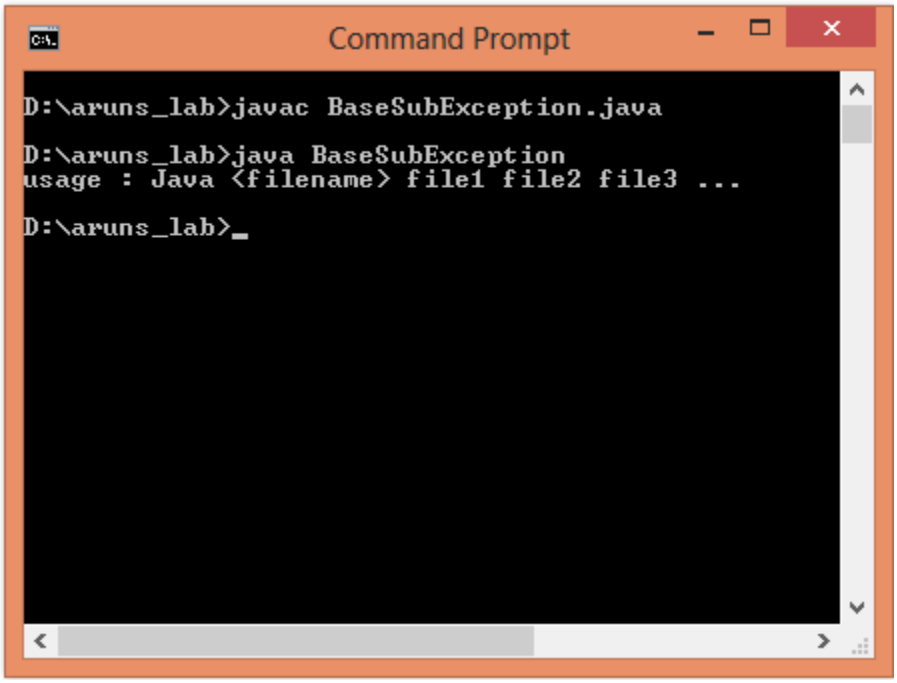
Program:

```
// Accept file name as command line arguments.
import java.io.*;
class BaseSubException
 {
 public static void main(String a[])
 {
 if(a.length == 0)
{
System.out.println("usage : Java <filename> file1 file2 file3 ...");
```

```
 return;
 }

for(int I=0;I<a.length; I++)
 {
 File f = new File(a[I]);
try {
 String line;
 DataInput d = new DataInputStream(new FileInputStream(a[I]));
 if (f.exists() && f.isFile())
 {
 System.out.println("file exists");
 System.out.println(f + "is ordinary file");
 System.out.println("printing the contents of file named : " + a[I]);
 System.out.println("========================");
 while((line = d.readLine()) != null)
 {
 System.out.println(line);
 }
 }
} catch(FileNotFoundException e)
 {
 if(f.exists() && f.isDirectory())
 {
 System.out.println("======================");
 System.out.println("Name : " + f + "is a directory");
System.out.println("inside catch of FileNotFoundException");
System.out.println("========================");
}
```

```
else {
 System.out.println("======================");
 System.out.println("Name : " + a[I] + "does not exists");
 System.out.println("generated exception :"+e);
 System.out.println("======================");
 }
 } catch(IOException p) {
System.out.println("super class is higher up in the program");
}
 }
 }
 }
```

Output:



0. Program Statement:

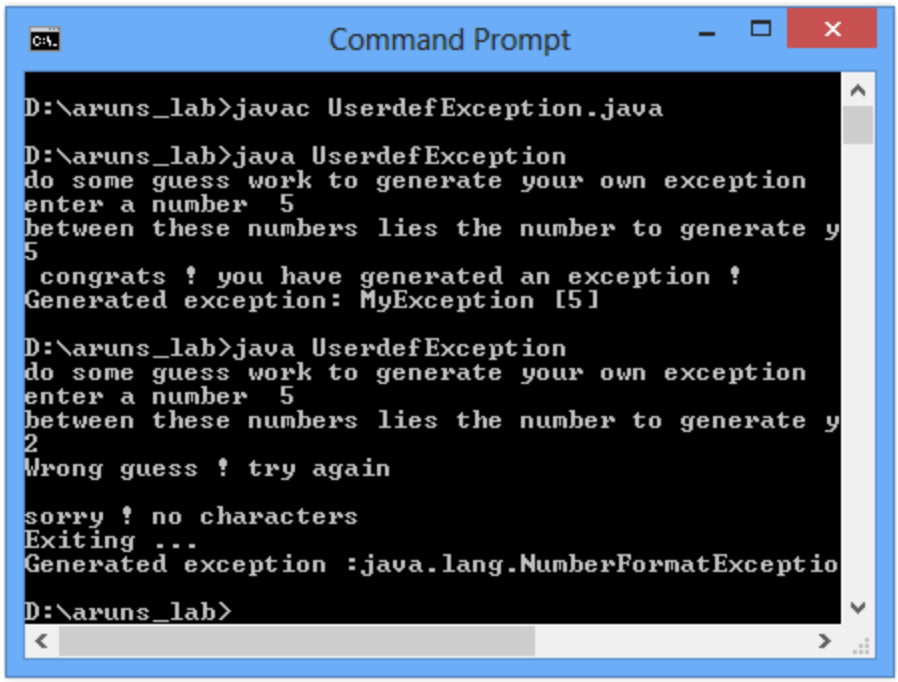
Write a Java Program for creation of user defined exception.

Program:

```
import java.io.*;
class MyException extends Exception
 {
 private int a;
 MyException(int b)
 {
 a = b;
 }
 public String toString()
 {
 return "MyException [" + a + "]";
 }
 }
class UserdefException
 {
public int x;
 final int k = 5;
 void getInt()
 {
 try {
 BufferedReader d = new BufferedReader(new InputStreamReader(System.in));
 System.out.println("do some guess work to generate your own exception ");
 System.out.println("enter a number  5");
 System.out.println("between these numbers lies the number to generate your own exception
");
 String line;
 while((line = d.readLine()) != null)
 {
 x = Integer.parseInt(line);
 if ( x == 5)
```

```
{
System.out.println(" congrats ! you have generated an exception !");
throw new MyException(x);
}
else
System.out.println("Wrong guess ! try again");
continue;
}
} catch(MyException m) {
System.out.println("Generated exception: " +m);
} catch(NumberFormatException n)
{
System.out.println("sorry ! no characters");
System.out.println("Exiting ...");
System.out.println("Generated exception :"+n);
}
catch(IOException e) { }
}
public static void main(String a[]) {
UserdefException u = new UserdefException();
u.getInt();
}
}
```

Output:



0. Program Statement:

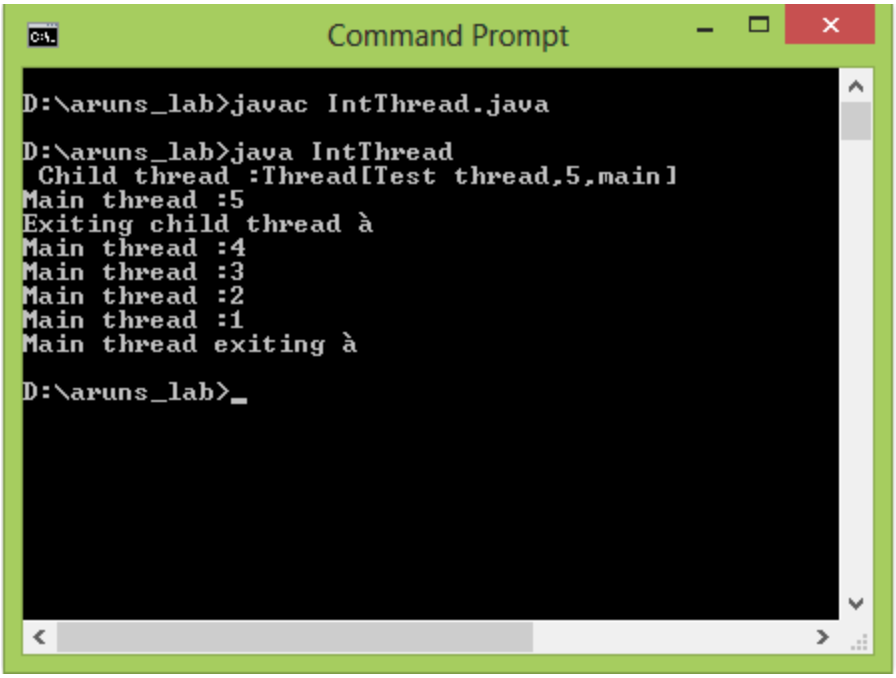
Write a Java Program to illustrate creation of threads using runnable interface.

Program:

```
class IntThread implements Runnable
 {
 Thread t;
 IntThread()
 {
 t = new Thread ( this, "Test thread");
 System.out.println (" Child thread :" + t);
 t.start();
 }
 public void run()
 {
 try {
```

```
for (int i= 5; i<0; i--)
{
System.out.println ("Child thread :" + i);
Thread.sleep (500);
}
}catch (InterruptedException e) { }
System.out.println ("Exiting child thread …" );
}
public static void main (String args[])
{
IntThread i = new IntThread();
try {
for ( int j=5; j >0; j--)
{
System.out.println ("Main thread :" + j);
Thread.sleep (1000);
}
} catch (InterruptedException e) { }
System.out.println ( "Main thread exiting …");
}
}
```

Output:



0. Program Statement:

Write a Java Program to create class MyThread in this class a constructor, call the base class constructor, using super and starts the thread. The run method of the class starts after this. It can be observed that both main thread and created child thread are executed concurrently.

Program:

0. Program Statement:

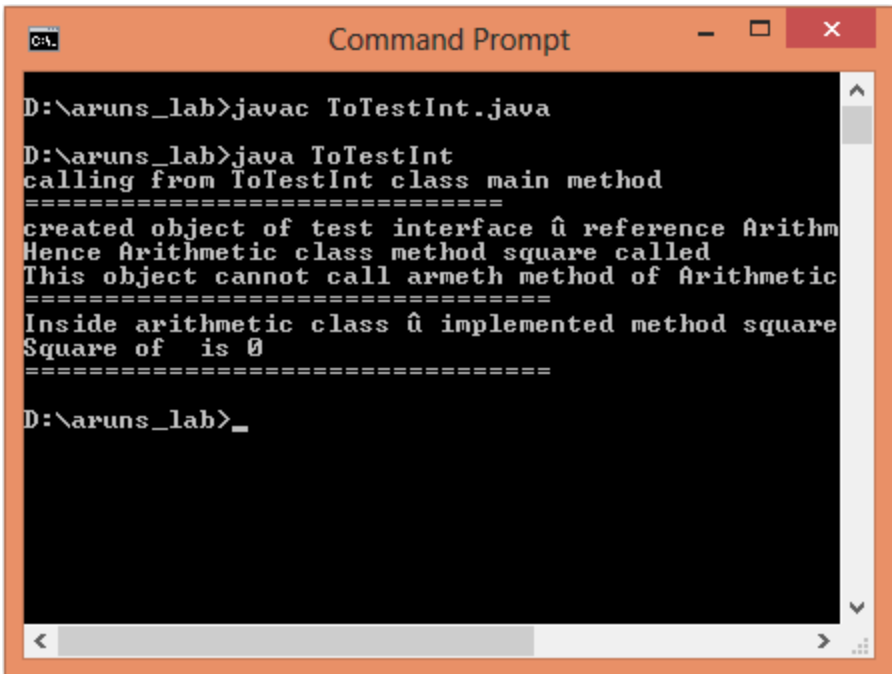
Write a Java Program illustrating multiple inheritance using interfaces

Program:

```
// Interface
 interface Test
{
 public int square(int a);
 }
// Implements
class arithmetic implements Test
 {
 int s = 0;
```

```
 public int square(int b)
 {
System.out.println("Inside arithmetic class – implemented method square");
 System.out.println("Square of " + " is "+s);
 return s;
 }
 void armeth()
 {
 System.out.println("Inside method of class Arithmetic");
}
}

// use the object
class ToTestInt
 {
 public static void main(String a[])
 {
 System.out.println("calling from ToTestInt class main method");
 Test t = new arithmetic();
 System.out.println("==============================");
 System.out.println("created object of test interface – reference Arithmetic class ");
 System.out.println("Hence Arithmetic class method square called");
 System.out.println("This object cannot call armeth method of Arithmetic class");
 System.out.println("==================================");
 t.square(10);
 System.out.println("==================================");
 }
}
```

Output:



0. Program Statement:

Write a Java Program to create a package p1, and implement this package in ex1 class.

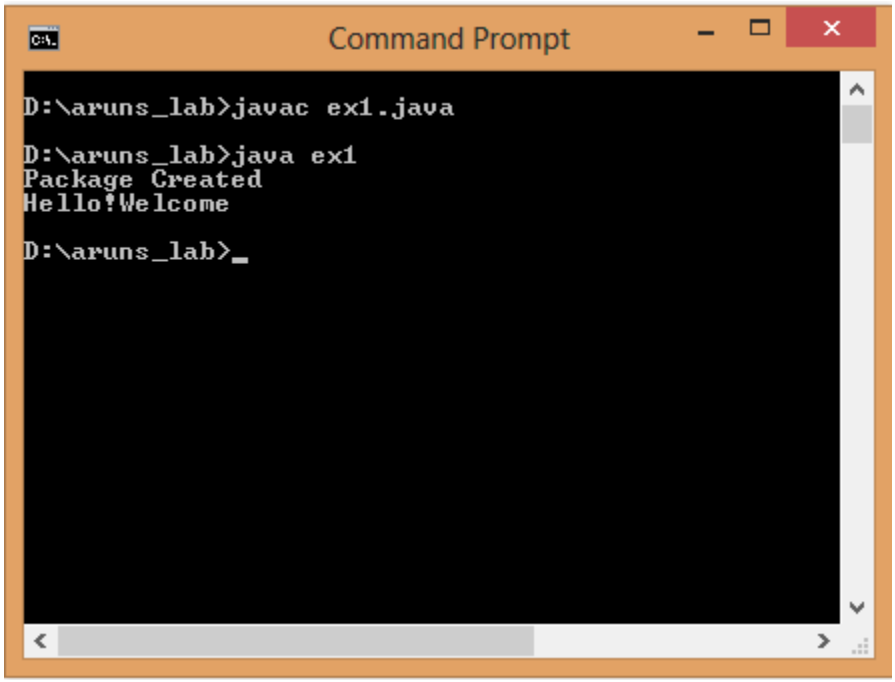
Program: save as ex.java in p1 (created package is sub directory of ur current directory)

```
package p1;
public class ex
{
 public ex()
 {
 System.out.println("Package Created");
 }
 public void display()
 {
 System.out.println("Hello!Welcome");
 }
}
```

Program: save as ex1.java in ur Current Directory

```
import p1.*;
class ex1
{
 public static void main(String[] a)
 {
 ex e=new ex();
 e.display();
 }
}
```

Output:



0. Program Statement:

Write a Java Program to create a package named mypack and import it in circle class.

Program:save as Circle.java in mypack subdirectory

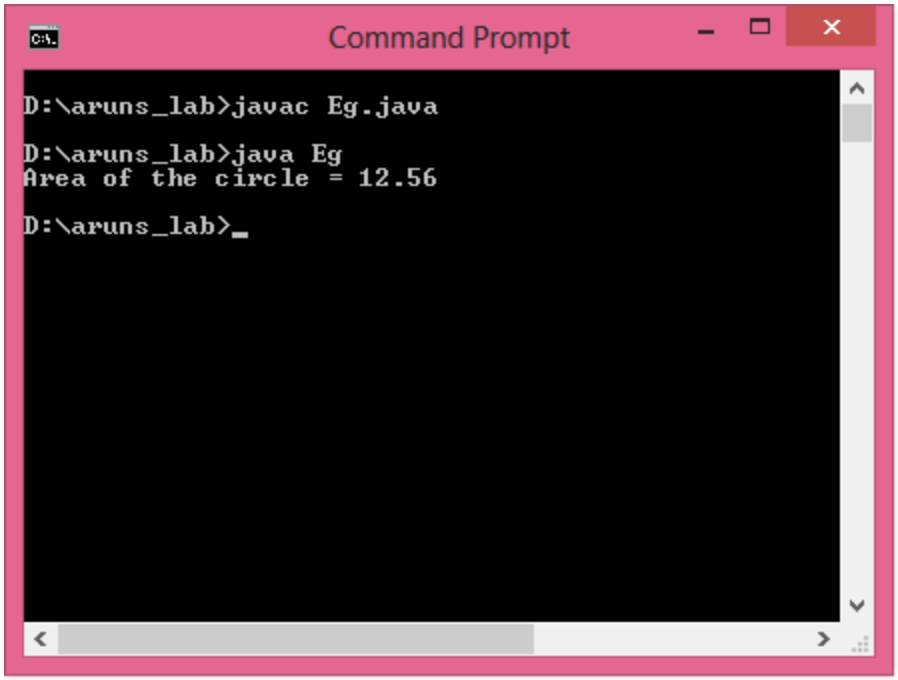
```
/create a package
package mypack;
```

```
public class Circle
{
 double r=2.0;
public  void area()
 {
 System.out.println("Area of the circle = " + (3.14 * r * r));
 }
}
```

Program:save as Eg.java in ur own directory

```
//implements the package
import mypack.Circle;
class Eg
{
 public static void main(String a[])
 {
 Circle c = new Circle();
 c.area();
 }
}
```

Output:



0. Program Statement:

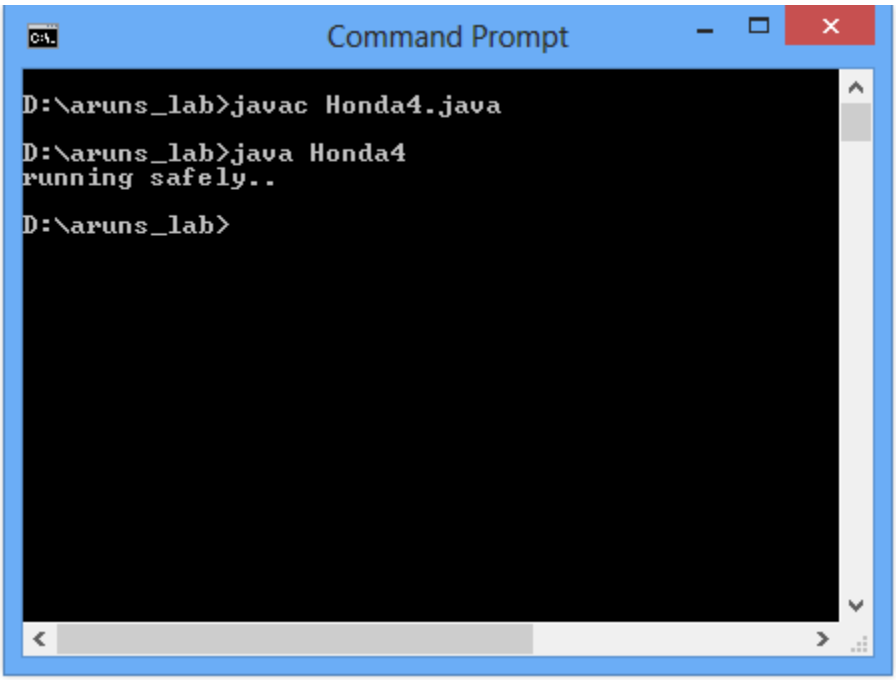
Write a Java Program to give a simple example for abstract class.

Program:

```
abstract class Bike
{
  abstract void run();
}

class Honda4 extends Bike{
void run()
{
System.out.println("running safely..");
}

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

```
 Bike obj = new Honda4();
 obj.run();
}
}
```

Output:



0. Program Statement:

Write a Java Program that describes the life cycle of an applet.

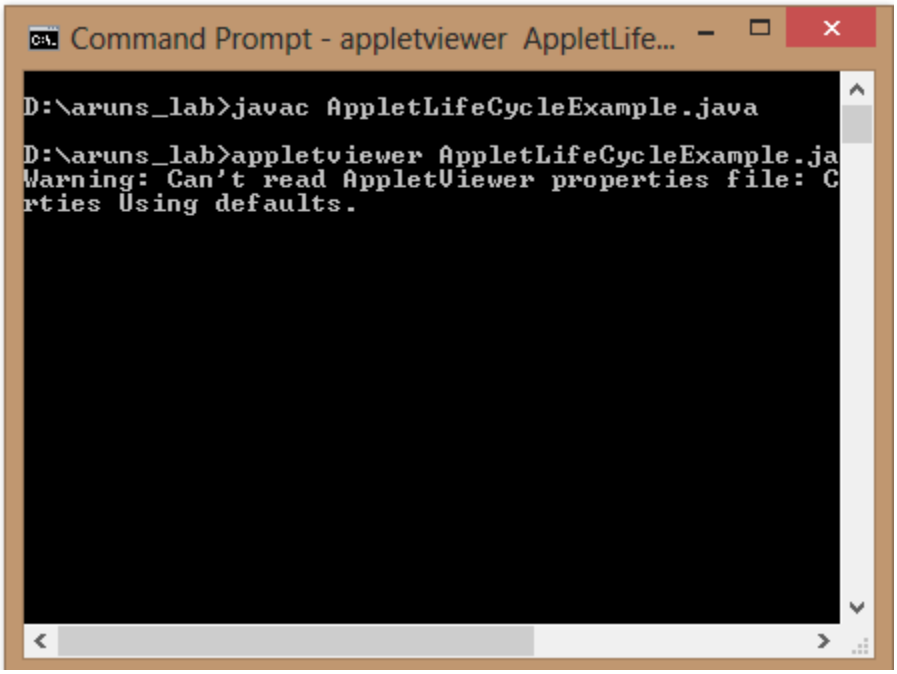
Program:

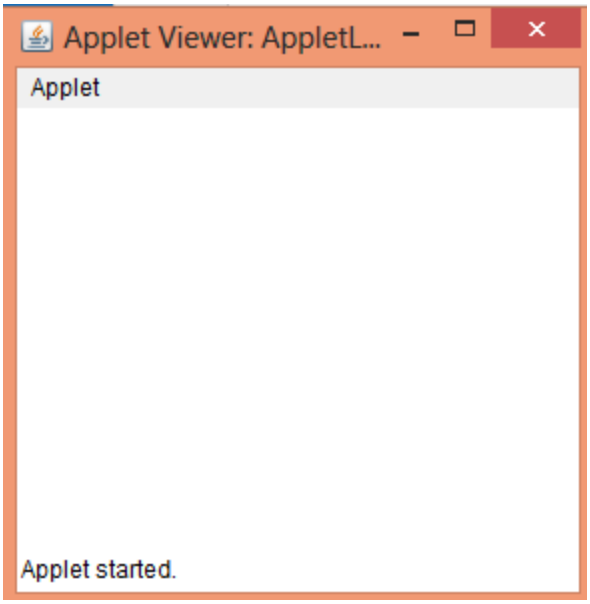
```
import java.applet.Applet;
import java.awt.Graphics;
 /*
<applet code="AppletLifeCycleExample" width=100 height=100>
</applet>
*/
 public class AppletLifeCycleExample extends Applet{
```

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

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

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

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

```
    }

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

}
```

Output:





13. Write a JAVA program using StringTokenizer class, which reads a line of integers and then displays each integer and the sum of all integers.

Program:

```
import java.util.*;


class StringTokenizerDemo {
    public static void main(String args[]) {
        int n;
        int sum = 0;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter integers with one space gap:");
        String s = sc.nextLine();
        StringTokenizer st = new StringTokenizer(s, " ");
        while (st.hasMoreTokens()) {
            String temp = st.nextToken();
            n = Integer.parseInt(temp);
            System.out.println(n);
```

```
            sum = sum + n;
        }
        System.out.println("sum of the integers is: " + sum);
        sc.close();
    }
}
```

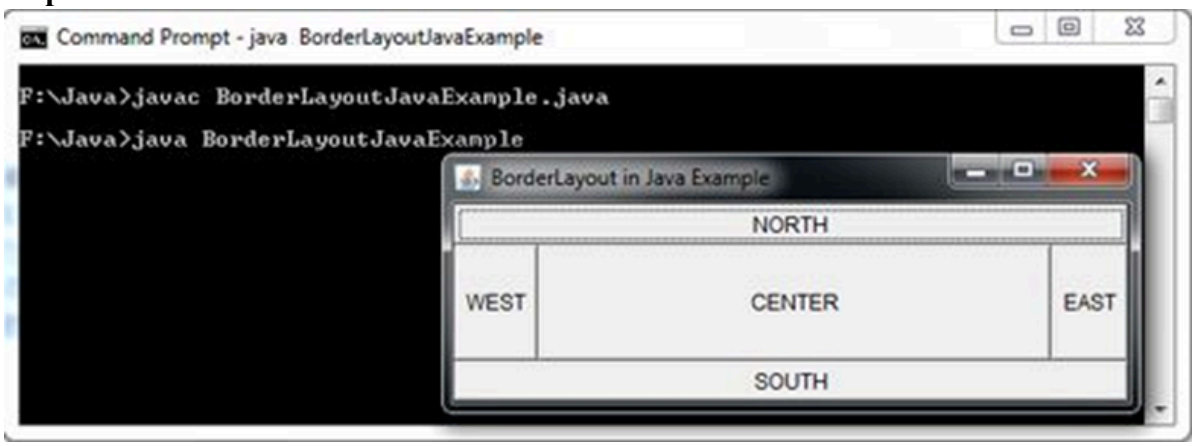
```
F:\Prog>javac StringTokenizerDemo.java

F:\Prog>java StringTokenizerDemo
Enter integers with one space gap:
10 85 45 56 45
10
85
45
56
45
sum of the integers is: 241

F:\Prog>
```

30. Write a JAVA program to create a border layout control.

Program:

```
import java.awt.*;
class BorderLayoutExample extends Frame
{
          BorderLayoutExample()
          {
          setLayout(new BorderLayout());
          add(new Button("NORTH"),BorderLayout.NORTH);
            add(new Button("SOUTH"),BorderLayout.SOUTH);
             add(new Button("EAST"),BorderLayout.EAST);
             add(new Button("WEST"),BorderLayout.WEST);
            add(new Button("CENTER"),BorderLayout.CENTER);
          }
}
  class BorderLayoutJavaExample
{
          public static void main(String args[])
  {
          BorderLayoutExample frame = new BorderLayoutExample();
              frame.setTitle("BorderLayout in Java Example");
          frame.setSize(400,150);
             frame.setVisible(true);
}
}
```

Output:



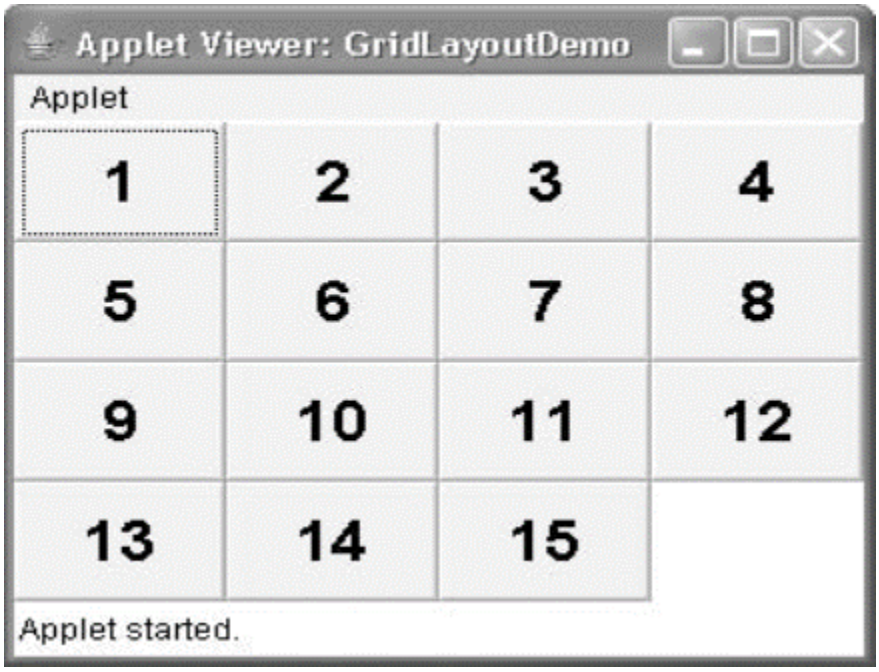
31. Write a JAVA program to create a grid layout control.

Program:

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

```
<applet code="GridLayoutDemo" width=300 height=200>
</applet>
*/
public class GridLayoutDemo extends Applet {
static final int n = 4;
public void init() {
setLayout(new GridLayout(n, n));
setFont(new Font("SansSerif", Font.BOLD, 24));
for(int i = 0; i < n; i++) {
for(int j = 0; j < n; j++) {
int k = i * n + j;
if(k > 0)
add(new Button("" + k));
}
}
}  }
```

Output:



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

Program:

```
import java.awt.*;


import java.awt.event.*;
```

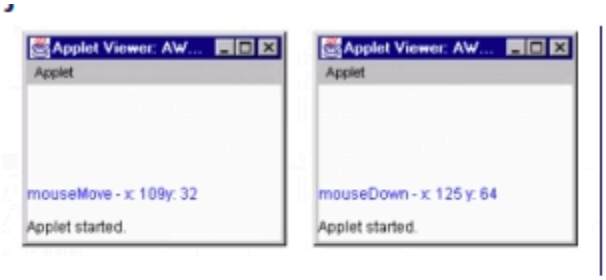
```
import java.applet.Applet;
public class AppletMouse extends Applet implements MouseListener, MouseMotionListener
{
 int x, y;
 String str="";
 public void init()
 {
  addMouseListener(this);
  addMouseMotionListener(this);
 }
                        // override ML 5 abstract methods
 public void mousePressed(MouseEvent e)
 {
  x = e.getX();
  y = e.getY();
  str = "Mouse Pressed";
  repaint();
 }
 public void mouseReleased(MouseEvent e)
 {
  x = e.getX();
  y = e.getY();
  str = "Mouse Released";
  repaint();
 }
 public void mouseClicked(MouseEvent e)
 {
   x = e.getX();
   y = e.getY();
```

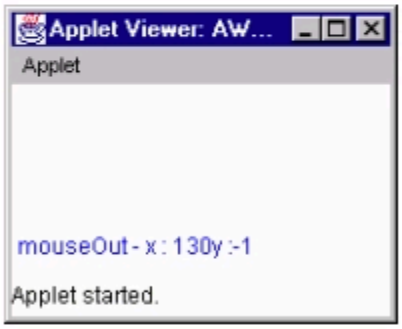
```
  str = "Mouse Clicked";
  repaint();
}
public void mouseEntered(MouseEvent e)
{
  x = e.getX();
  y = e.getY();
  str = "Mouse Entered";
  repaint();
}
public void mouseExited(MouseEvent e)
{
  x = e.getX();
  y = e.getY();
  str = "Mouse Exited";
  repaint();
}
                    // override two abstract methods of MouseMotionListener
public void mouseMoved(MouseEvent e)
{
  x = e.getX();
  y = e.getY();
  str = "Mouse Moved";
  repaint();
}
public void mouseDragged(MouseEvent e)
{
  x = e.getX();
  y = e.getY();
  str = "Mouse dragged";
```

```
    repaint();
  }
                          // called by repaint() method
  public void paint(Graphics g)
  {
    g.setFont(new Font("Monospaced", Font.BOLD, 20));
    g.fillOval(x, y, 10, 10);
    g.drawString(x + "," + y,  x+10, y -10);
    g.drawString(str, x+10, y+20);
    showStatus(str + " at " + x + "," + y);
  }
}
```

OUTPUT:

```
javac AppletMouse.java
Appletviewer Appletmouse.class
```





Write a JAVA program that allows user to draw lines, rectangles and ovals.

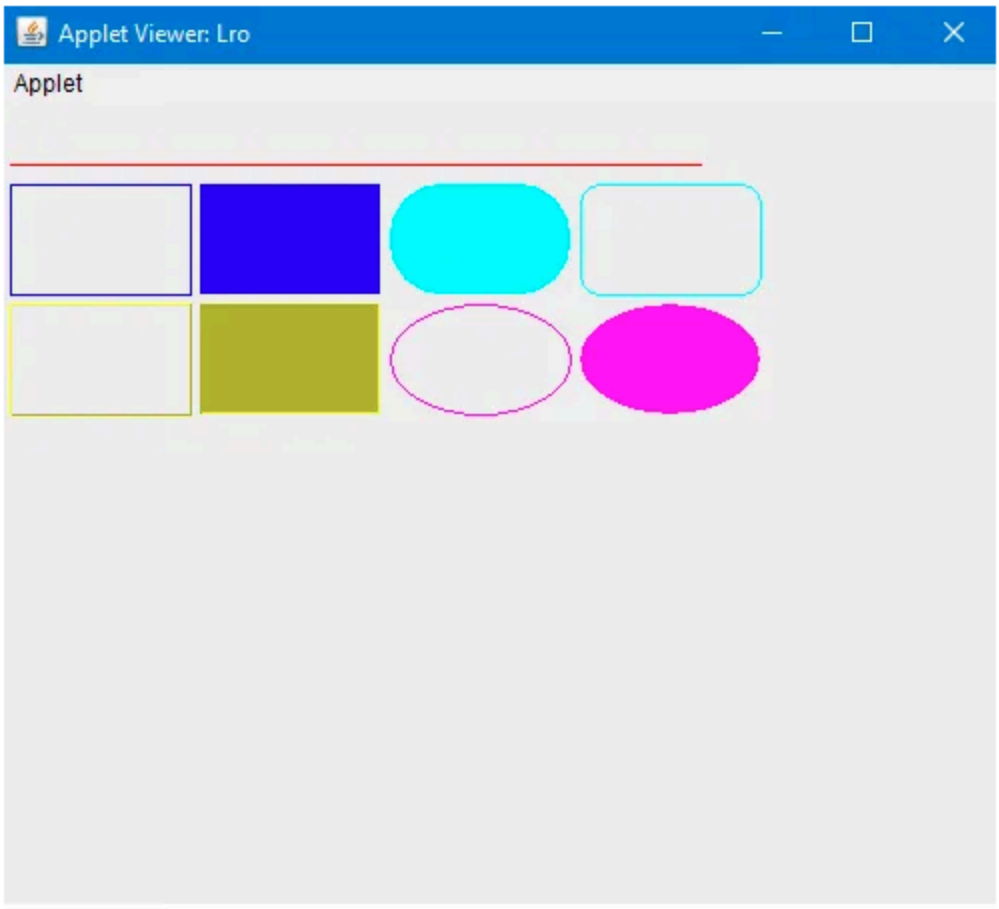
Program:

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

```
    import java.awt.*;
    import javax.swing.*;
/*<applet code="Lro" height="400" width="500" border="1">
    </applet>
    */
    public class Lro extends JApplet
{
     public void paint(Graphics g)
  {
super.paint(g);
g.setColor(Color.red);
g.drawLine(5,30,350,30);
 g.setColor(Color.blue);
g.drawRect(5,40,90,55);
 g.fillRect(100,40,90,55);
 g.setColor(Color.cyan);
g.fillRoundRect(195,40,90,55,50,50);
g.drawRoundRect(290,40,90,55,20,20);
 g.setColor(Color.yellow);
g.draw3DRect(5,100,90,55,true);
g.fill3DRect(100,100,90,55,false);
 g.setColor(Color.magenta);
 g.drawOval(195,100,90,55);
g.fillOval(290,100,90,55);
   }
}
```

OUTPUT:

```
javac Lro.java
Appletviewer Lro.class
```



Applet started.