Program -01

Preparing and practice – Installation of Java software, study of any Integrated development environment, sample programs on operator precedence and associativity, class and package concept, scope concept, control structures, constructors and destructors. Learn to compile, debug and execute java programs

Operator precedence

```
public class Precedence {
   public static void main(String[] args) {
      System.out.println( 3 + 3 * 2 );
      System.out.println( 3 * 3 - 2 );
      System.out.println( 3 * 3 / 2 );
      System.out.println("--");

      System.out.println( 1 * 1 + 1 * 1);
      System.out.println( 1 + 1 / 1 - 1);
      System.out.println( 3 * 3 / 2 + 2);
      System.out.println("--");

      int x = 1;
      System.out.println( x++ + x++ * --x );

      x = 1;
      System.out.println( x << 1 * 2 >> 1);

      x = 0xf;
      System.out.println( 0xf & 0x5 | 0xa );
   }
}
```

Output:

```
# java Precedence
9
7
4
--
2
1
6
--
5
2
15
```

package concept

1. //save by A.java
2.
3. package pack;
4. public class A{
5. public void msg(){System.out.println("Hello");}
6. }

1. //save by B.java
2. package mypack;
3. import pack.A;

```
4.
5.  class B{
6.    public static void main(String args[]){
7.     A obj = new A();
8.     obj.msg();
9.    }
10. }
```

```
Output:Hello
```

Example of Object and Class

```
1.  class Student1{
2.   int id;//data member (also instance variable)
3.   String name;//data member(also instance variable)
4.
5.   public static void main(String args[]){
6.    Student1 s1=new Student1();//creating an object of Student
7.    System.out.println(s1.id);
8.    System.out.println(s1.name);
9.   }
10. }
```

Output:0 null

Variable Scope in Java Programming

```
// Demonstrate block scope.
class Scope {
  public static void main(String args[])
  {
  int n1; // Visible in main

  n1 = 10;

  if(n1 == 10)
   {
   // start new scope
   int n2 = 20; // visible only to this block

   // num1 and num2 both visible here.
   System.out.println("n1 and n2 : "+ n1 +""+ n2);
   }
   // n2 = 100; // Error! y not known here

   // n1 is still visible here.
    System.out.println("n1 is " + n1);
  }
}
```

Output :

```
n1 and n2 : 10 20
n1 is 10
```

Program -02

Write Java program(s) on use of inheritance, preventing inheritance using final, abstract classes.

Java program on use of inheritance

```
class Parent
{
    public void p1()
    {
        System.out.println("Parent method");
    }
}
public class Child extends Parent {
    public void c1()
    {
        System.out.println("Child method");
    }
    public static void main(String[] args)
    {
        Child cobj = new Child();
        cobj.c1();   //method of Child class
        cobj.p1();   //method of Parent class
    }
}
```

Output :

```
Child method
Parent method
```

Preventing inheritance using final

```
final class DataV1 {


}

class DataV2 extends DataV1{


}

public class Javaapp {

    public static void main(String[] args){

        DataV2 obj = new DataV2();
   }
}
```

Abstract classes

```
abstract class A
{
 abstract void callme();
}
class B extends A
{
 void callme()
 {
  System.out.println("this is callme.");
 }
 public static void main(String[] args)
 {
  B b = new B();
  b.callme();
 }
}
```

Output :

```
this is callme.
```

Program -03

Write Java program(s) on dynamic binding, differentiating method overloading and overriding.

Dynamic binding

```
1. class Animal{
2.  void eat(){System.out.println("animal is eating...");}
3. }
4.
5. class Dog extends Animal{
6.  void eat(){System.out.println("dog is eating...");}
7.
8.  public static void main(String args[]){
9.   Animal a=new Dog();
10.  a.eat();
11. }
12.       }
13.
```

Output:dog is eating...

Differentiating Method Overloading And Overriding.

```
public class OverloadingOverridingTest {
public static void main(String[] args) {
//
Example of method overloading in Java
//
Loan cheapLoan = Loan.createLoan("HSBC");
Loan veryCheapLoan = Loan.createLoan("Citibank");
//
Example of method overriding in Java
Loan personalLoan = new
PersonalLoan();
personalLoan.toString();
System.out.println(" loan by citi bank");
}
}
class Loan {
private double interestRate; private String
customer; private String lender;
public static Loan createLoan(String lender) {
Loan loan = new Loan();
```

```
loan.lender = lender; return loan;
}
public static Loan createLoan(String lender, double interestRate) {
Loan loan = new Loan();
loan.lender = lender;
loan.interestRate = interestRate;
return loan;
}
public String toString() {
return "This is
Loan by Citibank";
}
}
class PersonalLoan extends Loan {
public String toString() {
return "This is Personal Loan by Citibank";
}
}
```

Output :Loan by citi bank

Program -04

Write Java program(s) on ways of implementing interface.

Interface Implementation

```
interface MyInterface
{
   public void method1();
   public void method2();
}
class XYZ implements MyInterface
{
  public void method1()
  {
      System.out.println("implementation of method1");
  }
  public void method2()
  {
      System.out.println("implementation of method2");
  }
  public static void main(String arg[])
  {
      MyInterface obj = new XYZ();
      obj. method1();
  }
}
```

Output:

```
implementation of method1
```

Program -05

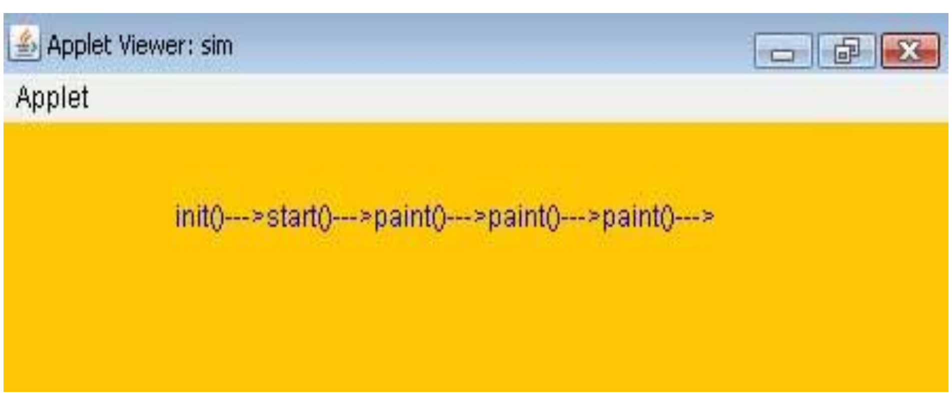
Write a program for the following

- **Develop an applet that displays a simple message.**
- **Develop an applet for waving a Flag using Applets and Threads**.

5(a) Java applet program that displays a simple message

```
import java.awt.*;
import java.applet.*;
/*
 <applet code="sim" width=300 height=300>
 </applet>
 */
 public class sim extends Applet
 {
               String msg=" ";
               public void init()
               {
                              msg+="init()--->";
                              setBackground(Color.orange);
               }
               public void start()
               {
                       msg+="start()--->";
                       setForeground(Color.blue);

               }
                public void paint(Graphics g)
               {
                       msg+="paint()--->";
                       g.drawString(msg,200,50);
               }
 }
```

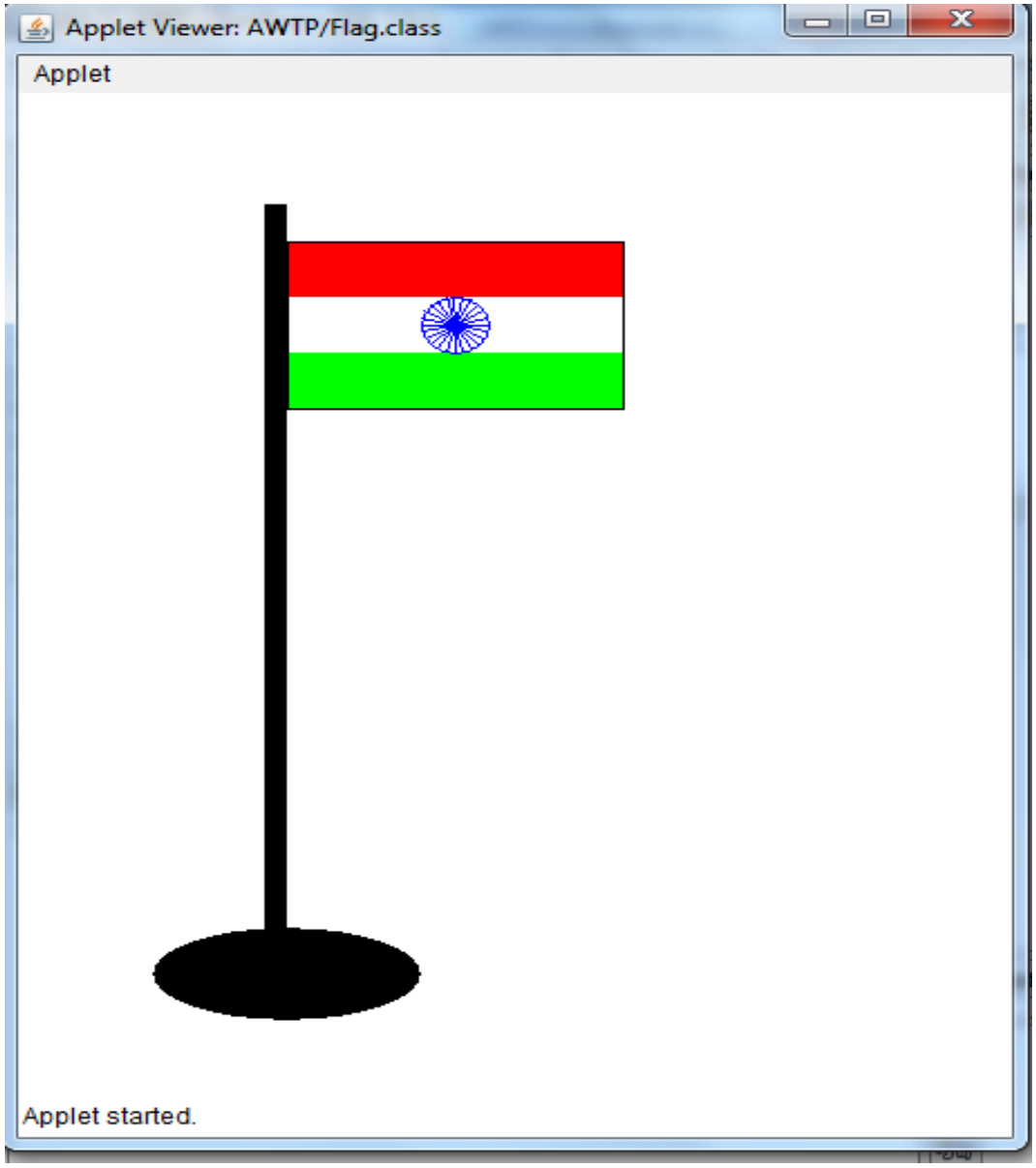
Output:



5(b) Java Code For Indian National Flag

```
package AWTP;

import java.applet.*;
import java.awt.*;
public class Flag extends Applet
{
   @Override
public void paint(Graphics g)
{
g.fillOval(60,450,120,50);
g.fillRect(110,60,10,400);
g.setColor(Color.red);
g.fillRect(120,80,150,30);
g.setColor(Color.white);
g.fillRect(120,110,150,30);
g.setColor(Color.green);
g.fillRect(120,140,150,30);
g.setColor(Color.black);
g.drawRect(120,80,150,90);
g.setColor(Color.blue);

g.drawOval(180,110,30,30);
int t=0;
int x=195,y=125;
double x1,y1;
double r=15;
double d;
for(int i=1;i<=24;i++)
{
d=(double)t*3.14/180.0;
x1=x+(double)r*Math.cos(d);
y1=y+(double)r*Math.sin(d);
g.drawLine(x,y,(int)x1,(int)y1);
t+=360/24;
}
}
}
```

OUTPUT :



Program -06

Write Java program(s) which uses the exception handling features of the language, creates exceptions and handles them properly, uses the predefined exceptions, and create own exceptions

EXCEPTION HANDLING PROGRAM

```
class Javaapp {

    public static void main(String[] args) {

        int a=1/0;
        System.out.println("Execution Complete");
    }
}
```

Program Output

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Javaapp.main(Javaapp.java:6)
Java Result: 1
```

MODIFIED CODE FOR THE ABOVE PROGRAM TO HANDLE THE EXCEPTION

```
class Javaapp {

    public static void main(String[] args) {

        try{

            int a=1/0;

        }catch(ArithmeticException ex)
        {
            System.out.println("Division by zero");

        }
        System.out.println("Execution Complete");
    }
}
```

Program Source : Without User-defined Exception

```
class Javaapp {

    public static void main(String[] args) {

        int a=1/0;
        System.out.println("Execution Complete");
    }
}
```

Program Source : With User-defined Exception(own exception)

```
class Javaapp {

    public static void main(String[] args) {

        try{
            int a=1/0;
        }catch(ArithmeticException ex)
        {
            System.out.println("Division by zero");
        }
        System.out.println("Execution Complete");
    }
}
```

Predefined exceptions

The three try, catch, and throw blocks

```
class Ex1
{
    public static void main (String [] args)
    {
        try
        {
            String s1=args[0];
            String s2=args[1];
            int n1=Integer.parseInt (s1);
            int n2=Integer.parseInt (s2);
            int n3=n1/n2;
            System.out.println ("DIVISION VALUE = "+n3);
        }
        catch (ArithmeticException Ae)
        {
            System.out.println ("DONT ENTER ZERO FOR ENOMINATOR...");
        }
        catch (NumberFormatException Nfe)
        {
            System.out.println ("PASS ONLY INTEGER VALUES...");
        }
        catch (ArrayIndexOutOfBoundsException Aioobe)
        {
            System.out.println ("PASS DATA FROM COMMAND PROMPT...");
        }
        finally
        {
            System.out.println ("I AM FROM FINALLY...");
        }
    }
}
```

Program -07

Write java program that inputs 5 numbers, each between 10 and 100 inclusive. As each number is read display it only if it's not a duplicate of any number already read. Display the complete set of unique values input after the user enters each new value.

```
public static void main(String[] args) {
Scanner input = new Scanner(System.in);
   int sid[] = new int[5];

   int count = 0;
   int x = 0;
   int num = 0;

   while (x < sid.length)
       {
       System.out.println("Enter number: ");
       num = input.nextInt();

       if ((num >= 10) && (num <= 100)) {
           boolean digit = false;
           x++;

           for (int i = 0; i < sid.length; i++)
            {   if (sid[i] == num)
                digit = true;
            }

       if (!digit) {

               sid[count] = num;

               count++;

       }

           else

           System.out.printf("the number was entered before \n");

                 }

         else
          System.out.println("number must be between 10 and 100");


       for (int i =0;  i < x; i++) {
           System.out.print(sid[i] +" ");

       }

       System.out.println();
   }

   }
}
```

Output

```
{
        Enter number:
            11
            11
            Enter number:
            21
            11 21
            Enter number:
            34
            11 21 34
            Enter number:
            11
            the number was entered before
            11 21 34 0
            Enter number:
            44
            11 21 34 44 0

}
```

Program -08

Write Java program(s) on creating multiple threads, assigning priority to threads, synchronizing threads, suspend and resume threads.

Creating a thread

Java defines two ways by which a thread can be created.

- By implementing the **Runnable** interface.
- By extending the **Thread** class.

Implementing the Runnable Interface

```
public void run()

class MyThread implements Runnable
{
 public void run()
 {
  System.out.println("concurrent thread started running..");
 }
}

class MyThreadDemo
{
 public static void main( String args[] )
 {
  MyThread mt = new MyThread();
  Thread t = new Thread(mt);
  t.start();
 }
}
```

Output :

```
concurrent thread started running..
```

To call the **run()** method, **start()** method is used. On calling start(), a new stack is provided to the thread and run() method is called to introduce the new thread into the program.

Extending Thread class

This is another way to create a thread by a new class that extends **Thread** class and create an instance of that class. The extending class must override **run()** method which is the entry point of new thread.

```
class MyThread extends Thread
{
 public void run()
 {
  System.out.println("Concurrent thread started running..");
 }
}

classMyThreadDemo
{
 public static void main( String args[] )
 {
  MyThread mt = new  MyThread();
  mt.start();
 }
}
```

Output :

```
concurrent thread started running..
```

Java-Thread Priority ---

```
class NewThread extends Thread{

    public long count;

    NewThread(int priority)
    {
        setPriority(priority);
        start();
    }

    public void run()
    {
        for(;;)
        {
            count++;
        }
    }
}

public class Javaapp {

    public static void main(String[] args) {

        NewThread th1 = new NewThread(Thread.MIN_PRIORITY);
        NewThread th2 = new NewThread(Thread.NORM_PRIORITY);
        NewThread th3 = new NewThread(Thread.MAX_PRIORITY);
        try{
            Thread.sleep(3000);
        }catch(InterruptedException ie)
        {
            System.out.println("Main Thread Interrupted");
        }
        th1.stop();
        th2.stop();
        th3.stop();
        System.out.println("Thread th1.count = "+th1.count);
        System.out.println("Thread th2.count = "+th2.count);
        System.out.println("Thread th3.count = "+th3.count);
    }
}
```

Program Output

```
Thread th1.count = 396572
Thread th2.count = 932295731
Thread th3.count = 1602284196
```

Example of priority of a Thread:

```
1. class TestMultiPriority1 extends Thread{
2.  public void run(){
3.   System.out.println("running thread name is:"+Thread.currentThread().getName());
4.   System.out.println("running thread priority is:"+Thread.currentThread().getPriority());
5.
6.  }
7.  public static void main(String args[]){
8.   TestMultiPriority1 m1=new TestMultiPriority1();
9.   TestMultiPriority1 m2=new TestMultiPriority1();
10.  m1.setPriority(Thread.MIN_PRIORITY);
11.  m2.setPriority(Thread.MAX_PRIORITY);
12.  m1.start();
13.  m2.start();
14.
15. }
16. }
```

Output:

```
running thread name is:Thread-0
 running thread priority is:10
 running thread name is:Thread-1
 running thread priority is:1
```

Using Synchronised block

```
class First
{
 public void display(String msg)
 {
  System.out.print ("["+msg);
  try
  {
   Thread.sleep(1000);
  }
  catch(InterruptedException e)
  {
   e.printStackTrace();
  }
  System.out.println ("]");
 }
}

class Second extends Thread
{
 String msg;
 First fobj;
 Second (First fp,String str)
 {
  fobj = fp;
  msg = str;
```

```
  start();
 }
 public void run()
 {
  synchronized(fobj)        //Synchronized block
  {
   fobj.display(msg);
  }
 }
}

public class Syncro
{
 public static void main (String[] args)
 {
  First fnew = new First();
  Second ss = new second(fnew, "welcome");
  Second ss1= new second (fnew,"new");
  Second ss2 = new second(fnew, "programmer");
 }
}
```

Output :

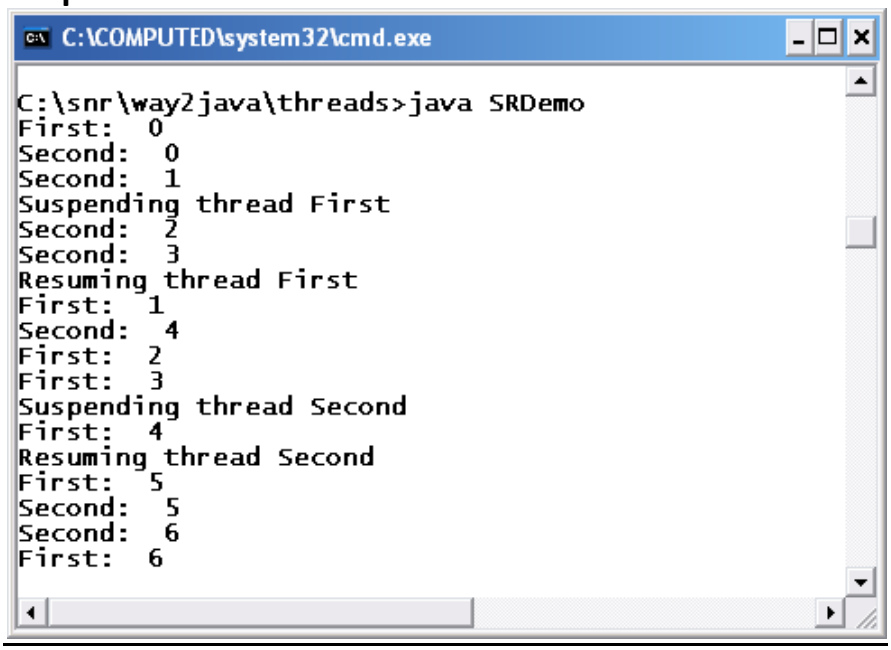
```
[welcome]
[new]
[programmer]
```

Suspend and resume threads

```
public class SRDemo extends Thread
{
 public static void main( String args[ ] )
 {
  SRDemo srd1 = new SRDemo();
  SRDemo srd2 = new SRDemo();
  srd1.setName("First");
  srd2.setName("Second");
  srd1.start();
  srd2.start();
  try
  {
   Thread.sleep( 1000 );
   srd1.suspend();
   System.out.println("Suspending thread First");
   Thread.sleep( 1000 );
   srd1.resume();
   System.out.println("Resuming thread First");

   Thread.sleep(1000);
   srd2.suspend();
   System.out.println("Suspending thread Second");
```

```
    Thread.sleep(1000);
    srd2.resume();
    System.out.println("Resuming thread Second");
   }
   catch(InterruptedException e)
   {
    e.printStackTrace();
   }
 }
 public void run()
 {
  try
  {
   for(int i=0; i<7; i++)
   {
    Thread.sleep(500);
    System.out.println( this.getName() + ":  " + i );
   }
  }
  catch(InterruptedException e)
  {
   e.printStackTrace();
  }
 }
}
```

Output



Program -10

Write a java program to split a given text file into n parts. Name each part as the name of the original file followed by .part<n> where n is the sequence number of the part file.

```
Import java.io.*;
Import java.util.Scanner;
Public class Split {
Public static  void main(String args[]) {
try
{
// Reading file and getting no. of files to be generated
String inputfile = "test.txt"; // Source File Name.
// No. of lines to be split and saved in each output file.
Double nol = 5.0;
File file = new File(inputfile);
Scanner scanner = new Scanner(file);
Int count = 0;
while(scanner.hasNextLine())
{
scanner.nextLine();
count++;
}
// Displays no. of lines in the input file.
System.out.println("Lines in the file: " +count);
Double temp = (count/nol);
Int temp1=(int)temp;
int
nof=0;
if(temp1==temp)
{
nof=temp1;
}
else
{
nof=temp1+1;
}
System.out.println("No. of files to be generated :"+nof);
// Actual splitting of file into smaller files
BufferedReader br = newBufferedReader(newFileReader(inputfile));
String strLine;
for(Int j=1;j<=nof;j++)
{
// Destination File Location
FileWriter fw= newFileWriter("File"+j+".txt");
for(inti=1;i<=nol;i++)
{
strLine = br.
```

```
readLine();
if(strLine!= null)
{
strLine=strLine+"\r\n";
fw.write(strLine);
}
}
fw.close();
}
br.close();
}
catch(Exception e) {
System.err.println("Error: " + e.getMessage());
}
}
}
```

OUTPUT:

The program assumes a file called test.txt.
Program executed successfully and every 5 lines from test.txt have been written to file1, file2, file3 ..... respectively

Program -11

Write a java program to create a super class called Figure that receives the dimensions of two dimensional objects. It also defines a method called area that computes the area of an object. The program derives two subclasses from Figure. The first is Rectangle and second is Triangle. Each of the sub classes override area() so that it returns the area of a rectangle and triangle respectively.

```
importjava.io.*;
// Using abstract methods and classes.
Abstract class Figure {
Double dim1;
Double dim2;
Figure(doublea, doubleb)
 {
dim1 = a;
dim2 = b;
}
// area is now an abstract method
abstractdoublearea();
}
classRectangle extendsFigure {
Rectangle(doublea, doubleb) {
super(a, b);
}
// override area for rectangle
Double area() {
System.out.println("Inside Area for Rectangle.");
return
dim1 * dim2;
}
}
classTriangle extendsFigure {
Triangle(doublea, doubleb) {
super(a, b);
}
// override area for right triangle
doublearea() {
System.out.println("Inside Area for Triangle.");
Return dim1 * dim2 / 2;
}
}
Class AbstractAreas {
Public static void main(String args[]) throwsIOException{
BufferedReader br=new BufferedReader(newInputStreamReader(System.in));
System.out.println("Enter leght and breadth of Rectangle");
doublelen=Double.parseDouble(br.readLine());
double bdt=Double.parseDouble(br.readLine());
System.out.println("Enter height and side of Triangle");
Double ht=Double.parseDouble(br.readLine());
```

```
Double sd=Double.parseDouble(br.readLine());
Rectangle r = new Rectangle(len, bdt);
Triangle t = new Triangle(ht, sd);
Figure figref; // this is OK, no object is created
figref = r;
System.out.println("Area is " + figref.area());
figref = t;
System.out.println("Area is " + figref.area());
}
}
```

O/P 1:

```
Enter length and breadth of Rectangle
4
2
Enter height and side of Triangle
4
2
Inside Area for Rectangle.
Area is 8.0
Inside Area for Triangle.
Area is 4.0
```

O/P 2:

```
 Enter length and breadth of
Rectangle
50
20
Enter height and side of Triangle
20
15
Inside Area for Rectangle.
Area is 1000.0
Inside Area for Triangle.
Area is 150.0
```

Program -12

Write a Java program that creates three threads. First thread displays “Good Morning” every one econd, the second thread displays “Hello” every two seconds and the third thread displays Welcome”every three seconds

```
classA implements Runnable
{
Thread thrd;
// Construct a new thread.
A(String name) {
thrd = newThread(this, name);
thrd.start(); // start the thread
}
synchronizedpublicvoidrun()
{
try
{
while(true)
{
Thread.sleep(1000);
System.out.println("good morning");
}
}
catch(Exception e)
{
System.out.println(thrd.getName() + " interrupted.");
}
}
}
classB implementsRunnable]
{
Thread thrd;
// Construct a new thread.
B(String name) {
thrd = newThread(this, name);
thrd.start(); // start the thread
}
synchronizedpublicvoidrun()
{
try
{
while(true)
{
Thread.sleep(2000);
System.out.println("hello");
}
}
catch(Exception e)
{
System.out.println(thrd.getName() + " interrupted.");
```

```
}
}
}
classC implementsRunnable
{
Thread thrd;
// Construct a new thread.
C(String name) {
thrd = newThread(this, name);
thrd.start(); // start the thread
}
synchronizedpublicvoidrun()
{
try
{
while(true)
{
Thread.sleep(3000);
System.out.println("welcome");
}
}
catch(Exception e){
System.out.println(thrd.getName() + " interrupted.");
}
}
}
classThreadDemo
{
publicstaticvoidmain(String args[])
{
At1=newA("thread1");
B t2=newB("thread2");
C t3=newC("thread3");
}
}
```

O/P:

The above program gives the following output continuously

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

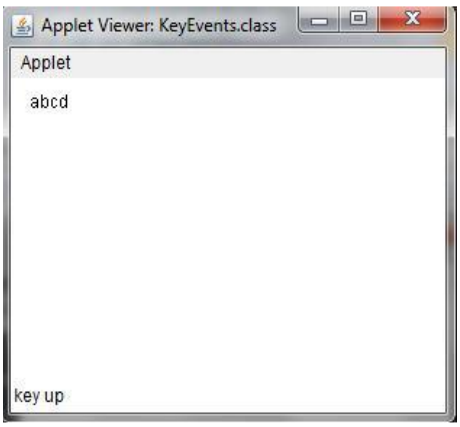
Program -14

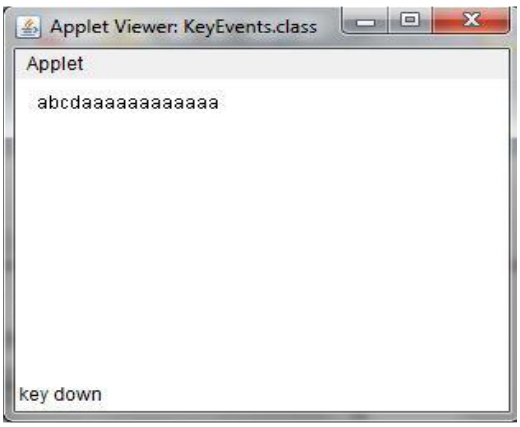
Write a java program to handle mouse events

```
import java.applet.*;
import java.awt.event.*;
import java.awt.*;
/* <applet code="KeyEvents.class" width=300 height=200> </applet> */
public class KeyEvents extends Applet implements KeyListener
{
String msg =" ";
int x=10,y=20;
public void init()
{
addKeyListener(this);
requestFocus();
}
public void keyPressed(KeyEvent k)
{
showStatus("key down");
}
public void keyReleased(KeyEvent k)
{
showStatus("key up");
}
public void keyTyped(KeyEvent k)
{
msg +=k.getKeyChar();
repaint();
}
public void paint(Graphics g)
{
g.drawString(msg,x,y);
}
}
```

**********OUTPUT**********

```
C:\jdk1.6.0_26\bin>javac Keyevents.java
C:\jdk1.6.0_26\bin>appletviewer Keyevents.java
```





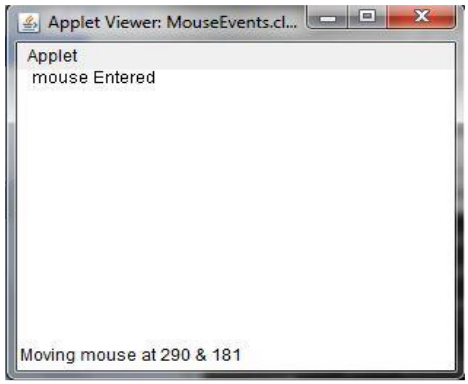
Program -15

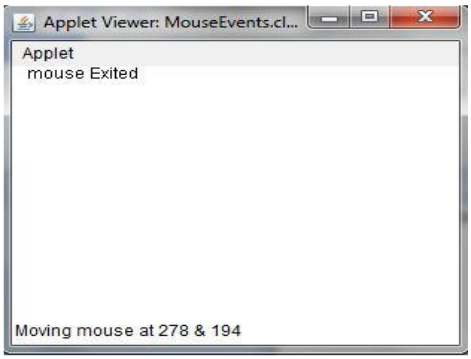
Write a Java Program to demonstrate Mouse events

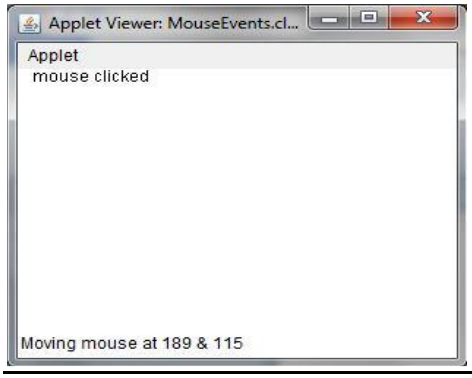
```
import java.applet.*;
import java.awt.event.*;
import java.awt.*;
/* <applet code="MouseEvents.class" width=300 height=200> </applet> */
public class MouseEvents extends Applet implements MouseListener,MouseMotionListener
{
String msg =" ";
int x=0,y=0;
public void init()
{
addMouseListener(this);
addMouseMotionListener(this);
}
public void mouseClicked(MouseEvent m)
{
x=10;
y=10;
msg ="mouse clicked";
repaint();
}
public void mouseEntered(MouseEvent m)
{
x=10;
y=10;
msg ="mouse Entered";
repaint();
}
public void mouseExited(MouseEvent m)
{
x=10;
y=10;
msg ="mouse Exited";
repaint();
}
public void mousePressed(MouseEvent m)
{
x=m.getX();
y=m.getY();
msg ="Down";
repaint();

}
public void mouseReleased(MouseEvent m)
{
x=m.getX();
y=m.getY();
msg ="Up";
repaint();
```

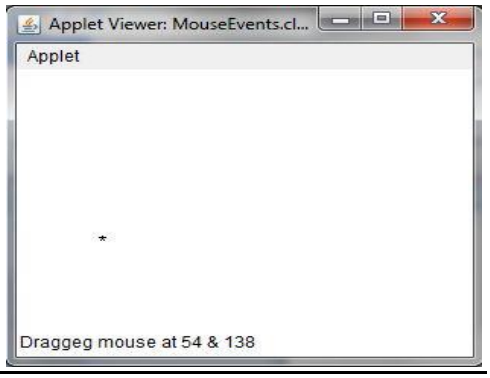
```
}
public void mouseDragged(MouseEvent m)
{
x=m.getX();
y=m.getY();
msg ="*";
showStatus("Draggeg mouse at " +x+ " & "+y);
repaint();
}
public void mouseMoved(MouseEvent m)
{
showStatus("Moving mouse at " +m.getX()+ " & "+m.getY());
}
public void paint(Graphics g)
{
g.drawString(msg,x,y);
}
}
```

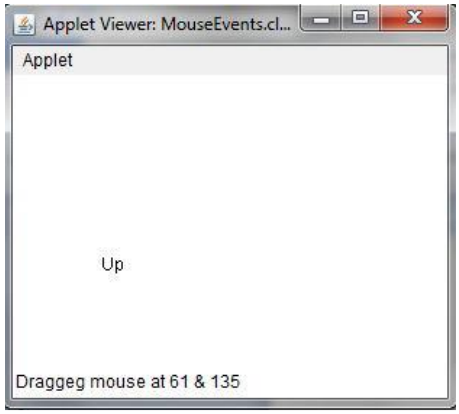
OUTPUT

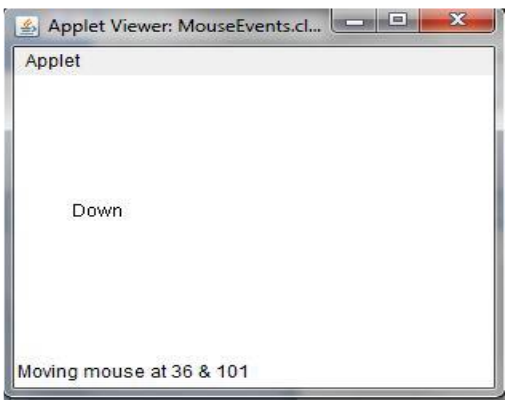












Program -19

Write a Java program that correctly implements producer consumer problem using the concept of inter thread communication

```
class Q {
        int n;
        boolean valueSet = false;

        synchronized int get() {
                if(!valueSet)
                        try {
                                wait();
                        }
                        catch(InterruptedException e) {
                                System.out.println("InterruptedException caught");
                        }

                        System.out.println("Got: " + n);
                        valueSet = false;
                        notify();
                        return n;
        }

        synchronized void put(int n) {
                if(valueSet)
                        try {
                                wait();
                        }
                        catch(InterruptedException e) {
                                System.out.println("InterruptedException caught");
                        }

                        this.n = n;
                        valueSet = true;
                        System.out.println("Put: " + n);
                        notify();
        }
}

class Producer implements Runnable {
        Q q;
        Producer(Q q) {
                this.q = q;
                new Thread(this, "Producer").start();
        }
        public void run() {
                int i = 0;
                while(true) {
                        q.put(i++);
                }
        }
}
```

```
class Consumer implements Runnable {
        Q q;
        Consumer(Q q) {
                this.q = q;
                new Thread(this, "Consumer").start();
        }

        public void run() {
                while(true) {
                        q.get();
                }
        }
}

class ProducerConsumer {
        public static void main(String args[]) {
                Q q = new Q();
                new Producer(q);
                new Consumer(q);
                System.out.println("Press Control-C to stop.");
        }
}
```

```
O/P:
Put: 0
Press Control-C to stop.
Got: 0
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
```

Program -20

Write a java program to find and replace pattern in a given file

```
*public static void main(String[] args) {
    System.out.print("Phase: ");
    Scanner sp = new Scanner(System.in);
    String p;
    p = sp.nextLine();

    System.out.print("Database: ");
    Scanner sd = new Scanner(System.in);
    String d;
    d = sd.nextLine();

    System.out.print("IP address: ");
    Scanner sip = new Scanner(System.in);
    int ip = sip.nextInt();

    {
     try
         {
         File file = new File("C://users//James//Desktop//newcommand.txt");
         BufferedReader reader = new BufferedReader(new FileReader(file));
         String line = "", oldtext = "";
         while((line = reader.readLine()) != null)
             {
             oldtext += line + "\r\n";
         }
         reader.close();

         String phase  = oldtext.replaceAll("phase", "" + p);
         String database = oldtext.replaceAll("db", "" + d);
         String ips = oldtext.replaceAll("IP", "" + ip);

         FileWriter writer = new FileWriter("C://users//James//Desktop//newcommand.txt");
         writer.write(phase + ips + database);


         writer.close();

     }
     catch (IOException ioe)
         {
         ioe.printStackTrace();
     }
 }
```

Program-21

Use inheritance to create an exception super class called ExceptionA and exception sub classes ExceptionB and ExceptionC, where ExceptionB inherits from ExceptionA and ExceptionC inherits from ExceptionB. Write a java program to demonstrate that the catch block for type ExceptionA catches exception of type ExceptionB and ExceptionC.

```
*public class ExceptionA extends Exception {

        public ExceptionA(String message){
            super(message);
        }
    }
```

Create exception `ExceptionB` and define require constructors and methods:

```
    public class ExceptionB extends ExceptionA {

        public ExceptionB(String message){
            super(message);
        }
    }
```

Create exception `ExceptionC` and define require constructors and methods:

```
    public class ExceptionC extends ExceptionB {

        public ExceptionC(String message){
            super(message);
        }
    }
```

```
    * public class TestException {

        public static void main(String[] args){

            try{
                getExceptionB();
            }catch(ExceptionA ea){
                ea.printStackTrace();
            }

            try{
                getExceptionC();
            }catch(ExceptionA ea){
                ea.printStackTrace();
            }

        }

        public static void  getExceptionB() throws ExceptionB{
            throw new ExceptionB("Exception B");
        }

        public static void  getExceptionC() throws ExceptionC{
            throw new ExceptionC("Exception C");
        }

    }
```

Program-23

Write a Java program to create a URLConnection and use it to examine the documents properties and content.

```
import java.net.*;
import java.io.*;
import java.util.Date;
class UCDemo
{
public static void main(String args[]) throws Exception {
int c;
URL hp = new URL("http://www.java-samples.com/j2me/");
URLConnection hpCon = hp.openConnection();
System.out.println("Date: " + new Date(hpCon.getDate()));
System.out.println("Content-Type: " +
hpCon.getContentType());
System.out.println("Expires: " + hpCon.getExpiration());
System.out.println("Last-Modified: " +
new Date(hpCon.getLastModified()));
int len = hpCon.getContentLength();
System.out.println("Content-Length: " + len);
if (len > 0) {
System.out.println("=== Content ===");
InputStream input = hpCon.getInputStream();
int i = len;
while (((c = input.read()) != -1) && (—i > 0)) {
System.out.print((char) c);
}
input.close();
} else {
System.out.println("No Content Available");
}
}
}
```

Program-24

Write a Java program which uses TCP/IP and Datagrams to communicate client and server.

Program for the server

```
import java.net.*;
import java.io.*;

class tcpip_server
{
public static void main(String args[]) throws IOException
{
ServerSocket n1=null;
try
{
n1=new ServerSocket(98);
}
catch(IOException e)
{
System.err.println("Port 98 could not be found");
System.exit(1);
}
Socket c=null;
try
{
c=n1.accept();
System.out.println("Connection from "+c);
}
catch(IOException e)
{
System.out.println("Accept failed");
System.exit(1);
}
PrintWriter out=new PrintWriter(c.getOutputStream(),true);
BufferedReader in=new BufferedReader(new InputStreamReader(c.getInputStream()));
String n;
BufferedReader sin=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Ready to type now");
while((n=sin.readLine())!=null)
{
out.println(n);
}
out.close();
c.close();
n1.close();
}
}
```

Program for the client

```
import java.net.*;
import java.io.*;
class tcpip_client
{
public static void main(String args[]) throws IOException
{
Socket s=null;
BufferedReader b=null;

try
{
s=new Socket(InetAddress.getLocalHost(),98);
b=new BufferedReader(new InputStreamReader(s.getInputStream()));
}

catch(UnknownHostException u)
{
System.err.println("I don't know host");
System.exit(0);
}
String inp;
while((inp=b.readLine())!=null)
{
System.out.println(inp);
}
b.close();
s.close();
}
}
```

Program-25

Create an interface for stack with push and pop operations. Implement the stack in two ways: fixed size stack and Dynamic stack (stack size is increased when stack is full).

Program for the fixed size stack

```
interface Stack<T> {
Stack<T> push(T ele);
T pop();
}

*public class StackArray<T> implements Stack<T> {

private T[] arr;

private int total;

public StackArray()
{
arr = (T[]) new Object[2];
}
private void resize(int capacity)
{
T[] tmp = (T[]) new Object[capacity];
System.arraycopy(arr, 0, tmp, 0, total);
arr = tmp;
}

public StackArray<T> push(T ele)
{
if (arr.length == total) resize(arr.length * 2);
arr[total++] = ele;
return this;
}

public T pop()
{
if (total == 0) throw new java.util.NoSuchElementException();
T ele = arr[--total];
arr[total] = null;
if (total > 0 && total == arr.length / 4) resize(arr.length/2);
return ele;
}
@Override
public String toString()
{
return java.util.Arrays.toString(arr);
}
```

```
}
```

Linked-List implementation

```
interface Stack<T> {
Stack<T> push(T ele);
T pop();
}
*public class StackLinkedList<T> implements Stack<T> {

private int total;

private Node first;

private class Node {
private T ele;
private Node next;
}

public StackLinkedList() { }

public StackLinkedList<T> push(T ele)
{
Node current = first;
first = new Node();
first.ele = ele;
first.next = current;
total++;
return this;
}

public T pop()
{
if (first == null) new java.util.NoSuchElementException();
T ele = first.ele;
first = first.next;
total--;
return ele;
}

@Override
public String toString()
{
StringBuilder sb = new StringBuilder();
Node tmp = first;
while (tmp != null) {
sb.append(tmp.ele).append(", ");
tmp = tmp.next;
}
return sb.toString();
```

```
}
}
```

26) Create multiple threads to access the contents of a stack. Synchronize thread to prevent simultaneous access to push and pop operations.

In this section, you create your first multithreaded program by creating a subclass of Thread and then creating, initializing, and starting two Thread objects from your class. The threads will execute concurrently and display Java is hot, aromatic, and invigorating. to the console window.

The source code of the ThreadTest1program.

```
class ThreadTest1
{
public static void main(String args[])
{
MyThread thread1 = new MyThread("thread1: ");
MyThread thread2 = new MyThread("thread2: ");
thread1.start();
thread2.start();
boolean thread1IsAlive = true;
boolean thread2IsAlive = true;
do {
if (thread1IsAlive && !thread1.isAlive()) {
thread1IsAlive = false;
                        System.out.println("Thread 1 is dead.");
}
if (thread2IsAlive && !thread2.isAlive()) {
thread2IsAlive = false;
System.out.println("Thread 2 is dead.");
}
} while(thread1IsAlive || thread2IsAlive);
}
}
class MyThread extends Thread
{
      static String message[] =
            { "Java", "is", "hot,", "aromatic,", "and", "invigorating."};
public MyThread(String id)
{
super(id);
}
public void run()
{
String name = getName();
for (int i=0;i<message.length;++i) {
randomWait();
System.out.println(name + message[i]);
}
}
void randomWait()
{
try {
```

```
sleep((long)(3000*Math.random()));
} catch (InterruptedException x) {
System.out.println("Interrupted!");
} } }
```