**Java Assignment no: 1**

1. Write a program that creates & initializes a four element Double array. Calculate & display the average of its values.

**class Ass11**

**{**

**public static void main(String arg[])**

{

double arr[]={5.6,5.7,5.8,5.9};

double sum=0,avg=0;

int i=0;

for(i=0;i<4;i++)

sum+=arr[i];

avg=sum/4;

System.out.println("Avg: " +avg);

}

}

1. Write a program that creates and initializes as 3X2 matrix. The elements are of int type. Calculate & display the average values of each row and column and also the grand average.

**class Ass12**

**{**

**public static void main(String arg[])**

**{**

double arr[][]={{1,2},{3,4},{5,6}};

double sum1=0,sum2=0,sum3=0;

double avg1=0,avg2=0,avg3=0;

for(int i=0;i<3;i++)

{

sum1=0;

for(int j=0;j<2;j++)

{

//sum1=0;

sum1+=arr[i][j];

avg1=sum1/3;

}

System.out.println("Avg of row: " +avg1);

}

for(int j=0;j<2;j++)

{

sum2=0;

for(int i=0;i<3;i++)

{

sum2+=arr[i][j];

avg2=sum2/2;

}

System.out.println("Avg of column: " +avg2);

}

for(int i=0;i<3;i++)

{

sum3=0;

for(int j=0;j<2;j++)

{

sum3+=arr[i][j];

avg3=sum3/6;

}}

System.out.println(" Grand Avg: " +avg3);

}

1. Write a program to compute the hypotenuse of right angle tringle whose sides are 4.5 & 8.9 unit long

**class Ass13**

**{**

**public static void main(String arg[])**

{

double a=4.5, b=8.9;

double h=0;

h=Math.sqrt((a\*a)+(b\*b));

System.out.println("Hypotaneous: " +h);

}

}

1. Write a program that displays the substring formed by the last ten characters of string [use the length() function

**class Ass14**

**{**

**public static void main(String arg[])**

{

String s="Information Technology";

int l;

l=s.length();

String s1=s.substring(l-10,l);

System.out.println("String: " +s1);

}

}

1. A string contains five numbers separated by commas. Write a program that displays the last numbers.

**class ass15**

**{**

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

**{**

int l;

char c[]={'1','2','3','4','5'};

String s=new String(c);

System.out.println(" The string is :"+s);

l=s.length();

String s1=s.substring(l-1,l);

System.out.println(" The character is :"+s1);

}

}

1. Write a program that counts the number of digits and letters in a string and displays these values.

**class Ass16**

**{**

**public static void main(String arg[])**

**{**

String s="a3bc7ef";

int l=s.length();

int h=0,g=0;

char ch;

for(int i=0;i<l; i++)

{

ch=s.charAt(i);

if(Character.isLetter(ch))

h++;

else if(Character.isDigit(ch))

g++;

else

System.out.println("No Digit or letter");

}

System.out.println("No. of Letter "+h);

System.out.println("No. of Digit "+g);

}

}

1. Write a program that determines whether a number is prime or not.

**class Ass17**

**{**

**public static void main(String arg[])**

{

int a=Integer.parseInt(arg[0]);

int j=0;

for(int i=2;i<=a/2;i++)

{

if((a%i)==0)

j++;

}

if(j==0)

System.out.println("A prime no");

else

System.out.println("Not a prime no");

}

}

1. Write a program that prints the numbers between 17 and 100 that can be evenly divided by 17

**class ass18**

**{**

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

{

int a,c=0;

for(int i=17;i<=100;i++)

{

if (i%17==0)

{

System.out.println("The no.s are :"+i);

}

}

}

}

1. Write a program that uses a while to count the number of vowels in a string.

**class ass19**

**{**

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

**{**

int i=0,a,l,f=0;

char c;

String s="beAutiful";

l=s.length();

while(i<l)

{

c=s.charAt(i);

if(c=='a'||c=='A'||c=='E'||c=='e'||c=='I'||c=='i'||c=='o'||c=='O'||c=='u'||c=='u')

f++;

i++;

}

System.out.println("No. of vowels are: "+f);

}

}

1. Write a program that displays all the factors of a number entered by user . For example, if the user entered 8, it would respond with 2 and 4.

**class Ass110**

**{**

**public static void main(String arg[])**

**{**

int a=Integer.parseInt(arg[0]);

int j=0;

for(int i=1;i<=a;i++)

{

if((a%i)==0)

{

System.out.println("Factor is: "+i);

j++;

}

}

System.out.println("Total no of factors are:"+j);

}

}

**Java Assignment no: 2**

**JAVA programming using Constructor**

**Constructor overloading & Method Overloading**

1. Write an application that declares a class name sphere. It should have instance variable to record its radius and the coordinates of its center. This should be of type double. Use the new operator to create sphere object set and display its instance variables

. (i) Use default constructor

(ii) the argument constructor

**class Sphere**

{ double radius;

double cdx,cdy;

**void display()**

**{**

System.out.println ("Instance variables are:\n Radius="+radius+" Coordinates x,y="+cdx+" "+cdy);

}

}

**class ASS21**

**{ public static void main(String arg[])**

{ Sphere s1=new Sphere();

s1.radius=10;

s1.cdx=3;

s1.cdy=6;

s1.display();

}

}

1. In problem 1 add two method findArea() & findVolume() take one variable radius as double to display the surface area & volume of the sphere (using argument constructor).

**class Sphere**

{ double radius;

double cdx,cdy;

**Sphere(double r,double x,double y)**

{ radius=r;

cdx=x;

cdy=y;

}

**void display()**

{

System.out.println("Instance variables are:\n Radius="+radius+" Coordinates x,y="+cdx+" "+cdy);

}

**double findArea()**

{ return 4\*3.14\*radius\*radius;

}

**double findVolume()**

{ return 4/3\*3.14\*radius\*radius\*radius;

}

}

**class ASS22**

**{ public static void main(String arg[])**

{ double area,volume;

Sphere s1=new Sphere(10,3,6);

s1.display();

area=s1.findArea();

volume=s1.findVolume();

System.out.println("Area="+area+" Volume="+volume);

}

}

1. Write a program which simply prints “Wonder of objects” without using any print statement.concrete methods or “.”

**class Wonder**

{ String s;

Wonder()

{ s="Wonders of object";

System.out.println(s);

}

}

**class ASS23**

**{ public static void main(String arg[])**

{ Wonder ob=new Wonder();

}

}

1. Write an application that accepts one command line argument and display its Spanish equivalent. For example, the tokens “UNO”,”dos”, “tres”, “quarto” and “cinco” are the Spanish tokens for the numbers “one ” through “five”. Create a class with static method to accomplish this task.

**class ASS24**

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

{

**String s=" ",s1="one",s2="two",s3="three",s4="four",s5="five";**

for(int i=0;i<args.length;i++)

{

s=args[i];

if(s.equalsIgnoreCase(s1))

{ System.out.println("uno");

}

else if(s.equalsIgnoreCase(s2))

{ System.out.println("dos");

}

else if(s.equalsIgnoreCase(s3))

{ System.out.println("tres");

}

else if(s.equalsIgnoreCase(s4))

{ System.out.println("quarto");

}

else if(s.equalsIgnoreCase(s5))

{ System.out.println("cinco");

}

else

{ System.out.println(s);

}

}

}

}

1. Write an application that creates ten rock objects and saves these in an array. Randomly select a mass between 1 and 10 kg for each rock as it is created. After all rocks have been created , display there individual mass and the total mass of all rocks.

**class Rock**

{ double mass;

}

**class ASS25**

**{ public static void main(String arg[])**

{ Rock ob[]=new Rock[10];

double sum=0;

System.out.println("Masses are:");

for(int i=0;i<10;i++)

{ ob[i]=new Rock();

ob[i].mass=10\*Math.random();

System.out.println(ob[i].mass);

sum=sum+ob[i].mass;

}

System.out.println("Total mass:"+sum);

}

}

1. Write a program that has an overloaded method. The first method should accept no arguments. The second method will accept a string and the third method will accept a string and an integer. The first method should display a message **"Welcome to JAVA"** once. The second method should display the message "**Welcome to polymorphism"** twice and third method should display the message **"welcome to overloading "thrice.**

**class Overload**

**{ void printdemo()**

{ System.out.println("WELCOME TO JAVA");

}

**void printdemo(String a)**

{ for(int i=0;i<2;i++)

{ System.out.println(a);

}

}

**void printdemo(String a,int k)**

{ for(int i=0;i<k;i++)

{ System.out.println(a);

}

}

}

**class ASS26**

**{ public static void main(String arg[])**

**{**

Overload obj=new Overload();

String A,B;

A="WELCOME TO POLYMORPHISM";

B="WELCOME TO METHOD OVERLOADING";

obj.printdemo();

obj.printdemo(A);

obj.printdemo(B,3);

}

}

**Java Assignment no: 3**

1. Write an application that illustrates variable hiding .Class S declares an instance variable X of integer type(i.e it is subset of S). Extends S by class T and declares an instance variable named X of string type. Instantiate both of these classes Initialize and display the variable named X with such these objects

**class S**

{ int x=10;

S()

{ System.out.println(x);

}

}

**class T extends S**

{ String x="I am inside class T";

T()

{ System.out.println(x);

}

}

**class ASS31**

{ **public static void main(String arg[])**

{ S a1=new S();

T b1=new T();

}

}

1. Write an application that illustrates how a method can invoke a Superclass method. Class I2 is extended by J2. Class J2 extended by K2. Each of these classes defines a method getDescription() that returns a string which includes a description of the class plus description of each Superclass. Instantiate each of the classes and invoke the getDescription() method.

**class I2**

**{ String getDescription()**

{ return "I am in I2";

}

}

**class J2 extends I2**

**{ String getDescription()**

{ System.out.println("I am in J2");

String s=super.getDescription();

return s;

}

}

**class K2 extends J2**

**{ String getDescription()**

{ System.out.println("I am in K2");

String s=super.getDescription();

return s;

}

}

**class ASS32**

**{ public static void main(String arg[])**

{ I2 a=new I2();

J2 b=new J2();

K2 c=new K2();

**I2 ref;**

**ref=a;**

System.out.println(ref.getDescription());

**ref=b;**

System.out.println(ref.getDescription());

**ref=c;**

System.out.println(ref.getDescription());

}

}

1. Write the following Applications:

Class S1 declares an instance variable named s1 of type integer. Class T1 extends S1 & declares an instance variable t1 of type integer. Class U1 extend T1 and declares an instance variable u1 of type integer . These variables are initialized by the constructor for that class. Each constructor also displays a string to indicate that it has started execution.

**class S1**

**{** int s1;

**S1(int s)**

{ s1=s;

System.out.println("S1 executed");

System.out.println("Integer="+s1);

}

}

**class T1 extends S1**

**{** int t1;

**T1(int a,int b)**

{ super(a);

t1=b;

System.out.println("T1 executed");

System.out.println("Integer="+t1);

}

}

**class U1 extends T1**

{ int u1;

**U1(int a,int b,int c)**

{ super(a,b);

u1=c;

System.out.println("U1 executed");

System.out.println("Integer="+u1);

} }

**class ASS33**

**{ public static void main(String arg[])**

{ S1 x=new S1(10);

T1 y=new T1(20,30);

U1 z=new U1(40,50,60);

}

}

1. The class Jetplane declares one abstract method name numEngines(). There are two concrete subclasses named DC8 & DC10. each of these provides a different implementation of the numEngines()method. The main class instantiates each of these classes and invokes numEngines() method. Also declare a constructor of in the Abstract class Jetplane. This constructor in Abstract class Jetplane This constructor variable to inherited by the subclass DC8 & DC10 (Super Keyword) to represent the sitting capacity of the plane.

**abstract class Jetplane**

{ String s;

Jetplane(String s1)

{ s=s1;

}

**abstract void numEngine();**

}

**class DC8 extends Jetplane**

{ **DC8(String s2)**

{ super(s2);

System.out.println(s);

}

**public void numEngine()**

{ System.out.println("I am in DC8");

}

}

**class DC10 extends Jetplane**

**{ DC10(String s2)**

{ super(s2);

System.out.println(s);

}

**public void numEngine()**

{ System.out.println("I am in DC10");

}

}

**class ASS34**

**{ public static void main(String arg[])**

{ DC8 x=new DC8("Sitting capacity is fair");

DC10 y=new DC10("Sitting capacity is good");

x.numEngine();

y.numEngine();

}

}

1. The abstract **Bird** class has four subclasses named **Pelicans, BrownBooby**, **LittleBittern, Shikra.**Write an application that demonstrates how to establish this class hierarchy. Define a argument constructor of super class **Bird** that has one String variable, which describes the list of the Bird as “**This is a list of the bird species recorded in India. The avifauna of India includes around 1301 species, of which 42 are endemic, 1 has been introduced by humans, and 26 are rare or accidental**.” to be displayed by inheriting from one of it’s subclass. Declare one instance variable of type string that indicates the shape of each subclass of **Bird**. Create and display instances of these objects. Override the toString() method of object to return a string with the description of a Bird and its food

**abstract class Bird**

{

String s,Shape,food,description;

**Bird(String s1)**

{ s=s1;

}

abstract public String toString();

abstract void Shape(String s2);

}

**class Pelicans extends Bird**

{

**Pelicans(String s1)**

{ super(s1);

System.out.println(s);

}

**public String toString()**

{ return ("The description and food of Pelicans:"+description+" "+food);

}

**public void Shape(String s2)**

{ Shape=s2;

System.out.println("The shape of the Bird:"+Shape);

}

}

**class BrownBobby extends Bird**

**{**

**BrownBobby(String s1)**

{ super(s1);

}

**public String toString()**

{ return ("The description and food of BrownBobby:"+description+" "+food);

}

**public void Shape(String s2)**

{ Shape=s2;

System.out.println("The shape of the Bird:"+Shape);

}

}

**class LittleBittern extends Bird**

**{**

**LittleBittern(String s1**)

{ super(s1);

}

**public String toString()**

{ return ("The description and food of LittleBittern:"+description+" "+food);

}

**public void Shape(String s2)**

{ Shape=s2;

System.out.println("The shape of the Bird:"+Shape);

}

}

**class Shikra extends Bird**

{

**Shikra(String s1)**

{ super(s1);

}

**public String toString()**

{ return ("The description and food of Shikra:"+description+" "+food);

}

**public void Shape(String s2)**

{ Shape=s2;

System.out.println("The shape of the Bird:"+Shape);

}

}

**class ASS35**

**{ public static void main(String arg[])**

{ Pelicans p1=new Pelicans ("This is a list of the bird species recorded in India. It includes around 1301 species of which 42 are endemic.It has been introduced by human and 26 are rare of accident.");

p1.Shape("Small");

p1.description="Small";

p1.food="Fruit";

System.out.println(p1);

BrownBobby b1=new BrownBobby(" ");

b1.Shape("Medium");

b1.description="Medium";

b1.food="Peanuts";

System.out.println(b1);

LittleBittern l1=new LittleBittern(" ");

l1.Shape("Small");

l1.description="Small";

l1.food="Fruit";

System.out.println(l1);

Shikra a1=new Shikra(" ");

a1.Shape("Small");

a1.description="Small";

a1.food="Fruit";

System.out.println(a1);

}}

1. The abstract class Airplane has three subclasses named **B747, B757 , B767** . Each airplane type can transport a different no of passengers , Each airplane object has a unique serial no. Write an application that declares this class hierchy. Instantiate several types of airplane & display them. Override tostring() method of object to return a string with the type serial no. & capacity.

**abstract class Airplane**

{ int nmb;

abstract void Passenger(int n);

abstract public String toString();

}

**class B747 extends Airplane**

**{ public void Passenger(int n)**

{ System.out.println("Accomodation="+n+" passengers");

}

**public String toString()**

{ return ("Serial no.- 00IND\_B747"+" B747 Capacity is fair");

}

}

**class B757 extends Airplane**

**{ public void Passenger(int n)**

{ System.out.println("Accomodation="+n+" passengers");

}

**public String toString()**

{ return ("Serial no.- 00IND\_B757"+" B757 Capacity is average");

}

}

**class B767 extends Airplane**

**{ public void Passenger(int n)**

{ System.out.println("Accomodation="+n+" passengers");

}

**public String toString()**

{ return ("Serial no.- 00IND\_B767"+" B767 Capacity is good");

}

}

**class ASS36**

**{ public static void main(String arg[])**

{ B747 ap4=new B747();

B757 ap5=new B757();

B767 ap6=new B767();

ap4.Passenger(50);

System.out.println(ap4);

ap5.Passenger(100);

System.out.println(ap5);

ap6.Passenger(200);

System.out.println(ap6);

}

}

1. Abstract Class Weight has four concrete subclass named **WeightA, WeightB, WeightC, WeightD**, The mass of them is 4, 1.5, 1.4, 7 respectively each weight object has String that identify its colour. Create six 6 Objects, & store them in a one dimensional array & display its entities & total mass.

**abstract class Weight**

{ String color;

int mass;

**Weight(String a,int b)**

{ color=a;

mass=b;

}

**abstract void display();**

}

**class WeightA extends Weight**

{ **WeightA(String a,int b)**

{ super(a,b)

}

**public void display()**

{ System.out.println("The color is "+color+" Mass is:"+mass);

}

}

**class WeightB extends Weight**

**{ WeightB(String a,int b)**

{ super(a,b)

}

**public void display()**

{ System.out.println("The color is "+color+" Mass is:"+mass);

}

}

**class WeightC extends Weight**

**{ WeightC(String a,int b)**

{ super(a,b)

}

**public void display()**

{ System.out.println("The color is "+color+" Mass is:"+mass);

}

}

**class WeightD extends Weight**

**{ WeightD(String a,int b)**

{ super(a,b)

}

**public void display()**

{ System.out.println("The color is "+color+" Mass is:"+mass);

}

}

**class ASS37**

**{ public static void main(String arg[])**

{ weight WW;

weight m[]=new Weight[6];

m[0]=new WeightA("RED",4);

WW=m[0];

WW.display();

m[1]=new WeightB("GREEN",5);

WW=m[1];

WW.display();

m[2]=new WeightC("YELLOW",4.5);

WW=m[2];

WW.display();

m[3]=new WeightD("VIOLET",10);

WW=m[3];

WW.display();

m[4]=new WeightA("BLUE",8);

WW=m[4];

WW.display();

m[5]=new WeightD("PINK",6.5);

WW=m[5];

WW.display();

}

}

1. The abstract class fruit has four subclass named **apple, orange, strawberry.** Write the following Application that demonstrate how to establish this class hierarchy . Declare one instance variable of type string that indicates the color of fruit. Create & display instances of these objects. **Override the tostring()** method of object to return a string with the name of the fruit & its color

**abstract class fruit**

**{**

String colour;

String name;

**abstract public String toString();**

**fruit(String a, String b)**

{

colour =a;

name =b;

}

}

**class apple extends fruit**

**{**

**apple(String a, String b)**

{

super(a,b);

}

**public String toString()**

{

return "the colour of the fruit is:"+colour +" the fruit name is:"+name;

}

}

**class banana extends fruit**

**{**

**banana(String a, String b)**

{

super(a,b);

}

**public String toString()**

**{**

return "the colour of the fruit is:"+colour +" the fruit name is:"+name;

}

}

**class orange extends fruit**

**{**

**orange(String a, String b)**

**{**

super(a,b);

}

**public String toString()**

{

return "the colour of the fruit is:"+colour +" the fruit name is:"+name;

}

}

**class strawberry extends fruit**

**{**

**strawberry(String a, String b)**

**{**

super(a,b);

}

**public String toString()**

{

return "the colour of the fruit is:"+colour +" the fruit name is:"+name;

}

}

**class ass3\_8**

**{**

**public static void main(String arg[])**

**{**

apple a = new apple("red","apple");

banana b = new banana("yellow","banana");

orange o = new orange("orange","orange");

strawberry s = new strawberry("pink","strawberry");

**fruit ref;**

**ref =a;**

System.out.println(ref);

**ref =b;**

System.out.println(ref);

**ref =o;**

System.out.println(ref);

**ref =s;**

System.out.println(ref);

}

}

**JAVA** **Assignment 4**

**INTERFACE:**

* 1. Write the following application an interface **shape2D** that declares a **getarea()** method that calculates returns the area of an enclosed **2D shape.** Interface **shape3D** declares a **getvolume()** method that calculates & returns the volume of an enclosed **3D shape**. The abstract class shape declares an abstract method **display()** is extended by circle and sphere class. Abstract class shape implements both **shape 2D** and **shape 3D** interface. The shape instantiates each of these class.

**interface Shape2D**

**{** double getArea();

}

**interface Shape3D**

{ double getVolume();

}

**abstract class Shape implements Shape2D,Shape3D**

{

double rad;

**Shape(double r)**

{ rad=r;

}

**public double getArea()**

{ return 3.14\*rad\*rad;

}

**public double getVolume()**

{ return 1.33\*3.14\*rad\*rad\*rad;

}

**abstract void display();**

}

**class Circle extends Shape**

{

**Circle(double r1)**

{ super(r1);

System.out.println("Radius of Circle="+ rad);

}

**public void display()**

{ System.out.println("Area of Circle="+getArea());

}

}

**class Sphere extends Shape**

{

**Sphere(double r1)**

{ super(r1);

System.out.println("Radius of Sphere="+ rad);

}

**public void display()**

{ System.out.println("Volume of Sphere="+getVolume());

}

}

**class ASS41**

**{ public static void main(String arg[])**

{ Circle a= new Circle(10);

Sphere b= new Sphere(20);

Shape ref;

ref=a;

ref.display();

ref=b;

ref.display();

}

}

1. Write a program with a class **Teacher** containing two fields **Name and Qualification**. Extends the class to **Department**, which contains **Dept no and Dept name**. An interface named as **College,** which contains one field **Name of the college**. Using the above classes and interface get the appropriate information and display it.

**interface College**

{ String name="HIT";

void display();

}

**class Department implements College**

**{**

int deptno;

String dname;

**Department(int dno,String dnm)**

{

deptno=dno;

dname=dnm;

}

**public void display()**

**{**

System.out.println(name);

}

}

**class Teacher extends Department**

{

String tname,tqual;

**Teacher(int dno,String dnm,String s,String s1)**

{

super(dno,dnm);

tname=s;

tqual=s1;

System.out.println("Dept no:"+deptno+" Department name"+dname);

System.out.println("Name="+tname+" Qualification="+tqual);

}

}

**class ASS42**

**{ public static void main(String arg[])**

{

**College ref;**

Teacher t=new Teacher(15,"information Technology","ABC","M.Tech");

ref=t;

ref.display();

}

}

3. Interface **Material** defines a set of string constants for various materials. Abstract class **MaterialObjects** has one instance variable named **material** of type String . This records the material used to construct that object classes **Ball, Coin and Ring** extend **Material object**. The constructors initialize the **material** variable. Class **MaterialObjects** instantiates these three classes. A different material is passed to each constructor. The material of each object is displayed

**interface Material**

{ String s1="rubber",s2="gold",s3="Silver";

}

**abstract class Materialobj implements Material**

{ String mat;

**Materialobj(String raw)**

{ mat=raw;

}

**abstract void display1();**

}

**class Ball extends Materialobj**

{ String x;

**Ball(String rw,String mtrl)**

{ super(rw);

x=mtrl;

System.out.println("Ball is made from:"+s1);

}

**public void display1()**

{ System.out.println(x+" is made from:"+mat);

}

}

**class Coin extends Materialobj**

{ String y;

**Coin(String rw,String mtrl)**

{ super(rw);

y=mtrl;

System.out.println("Coin is made from:"+s2);

}

**public void display1()**

{ System.out.println(y+" is made from:"+mat);

}

}

**class Ring extends Materialobj**

{ String z;

**Ring(String rw,String mtrl)**

{ super(rw);

z=mtrl;

System.out.println("Ring is made from:"+s3);

}

**public void display1()**

{ System.out.println(z+" is made from:"+mat);

}

}

**class ASS43**

**{ public static void main(String arg[])**

{ Ball b=new Ball("Aluminium","Utensils");

Coin c=new Coin("Almond","Oil");

Ring r=new Ring("Wood","Chair");

b.display1();

c.display1();

r.display1();

}

}

4. Interface **AntiLockBreakers , CruiseControl and PowerSteering** declare optional functionality for an automobile. Each interface declares one method that has the same name as its interface. The abstract Auto class is extended by **Model1, Model2, Model3** classes. Power steering is available for **Model1** Objects. **Antilock breakers and Cruise control** are available for **Model2** objects. **Cruise control** is available for M**odel3** objects. Instantiate each of these classes and exercises its methods.

**interface Antilockbreakers**

{ String Antilockbreakers();

}

**interface CruiseControl**

{ String CruiseControl();

}

**interface PowerSteering**

{ String PowerSteering();

}

**abstract class Auto implements Antilockbreakers,CruiseControl,PowerSteering**

{ public String Antilockbreakers()

{ return "Antilockbreakers is available";

}

**public String CruiseControl()**

{ return "CruiseControl is available";

}

**public String PowerSteering()**

{ return "PowerSteering is available";

}

**abstract void display();**

}

**class Model1 extends Auto**

{ public void display()

{ System.out.println("In Model1, "+PowerSteering());

}

}

**class Model2 extends Auto**

{ **public void display()**

{

System.out.println("In Model2, "+Antilockbreakers()+" and " +CruiseControl());

}

}

**class Model3 extends Auto**

{ **public void display()**

{ System.out.println("In Model3, "+CruiseControl());

}

}

**class ASS44**

**{ public static void main(String arg[])**

{

Auto ref;

Model1 a= new Model1();

Model2 b= new Model2();

Model3 c= new Model3();

ref=a;

ref.display();

ref=b;

ref.display();

ref=c;

ref.display();

}

}

5. Interface **LuminousObject** declares **Lightoff()** and **Lighton()** methods. Class SolidObject is extended by cone and implements **LuminousObject.** Class **LuminousCone** extends **Cone** and implements **LuminousObject.** Class **LuminousCube** extends **Cube** and implements **LuminousObject.** Instantiate the **LuminousCone** and **LuminousCube** classes.. Use interface reference to those objects Invoke the methods of **LuminousObject** interface via the interface reference.

**interface Luminousobj**

**{** String lightOff();

String lightOn();

void display();

}

**class Solidobj implements Luminousobj**

**{ public String lightOff()**

{ return "Light is Off";

}

**public String lightOn()**

{ return "Light is On";

}

**public void display()**

{

System.out.println("Use the Interface reference");

System.out.println(lightOff());

System.out.println(lightOn());

}

}

**class Cone extends Solidobj**

**{**

String s1;

**Cone(String s)**

{

s1=s;

System.out.println(s1);

}

}

**class LuminousCone extends Cone**

**{**

**LuminousCone(String s)**

{

super(s);

System.out.println("In LuminousCone:");

}

}

**class Cube extends Solidobj**

{ String s1;

Cube(String s)

{

s1=s;

System.out.println(s1);

}

}

**class LuminousCube extends Cube**

**{ LuminousCube(String s)**

{

super(s);

System.out.println("In LuminousCube:");

}

}

**class ASS45**

**{ public static void main(String arg[])**

**{**

**Luminousobj ref;**

LuminousCone a = new LuminousCone("I am subclass of Cone");

**ref=a**;

ref.display();

System.out.println();

LuminousCube b = new LuminousCube("I am subclass of Cube" );

**ref=b**;

ref.display();

}

}

6. Interface P is extended by **P1** and **P2**. Interface **P12** inherits both from **P1** & **P2**. Each interface declares one constant and one method. Class **Q** implements **P12.** Instantiate **Q** and invoke each of its methods . each method displays one of the constants.

**interface P**

{

String inP="interface P";

void Pdisp();

}

**interface P1 extends P**

{

String inP1="interface P1";

void P1disp();

}

**interface P2 extends P**

{

String inP2="interface P2";

void P2disp();

}

**interface P12 extends P1,P2**

{

String inP12="interface P12";

void P12disp();

}

**class Q implements P12**

{

**public void Pdisp()**

{

System.out.println(inP);

}

**public void P1disp()**

{

System.out.println(inP1);

**}**

**public void P2disp()**

{

System.out.println(inP2);

}

**public void P12disp()**

{

System.out.println(inP12);

}

}

**class ASS46**

**{**

**public static void main(String arg[])**

**{**

Q q= new Q();

q.Pdisp();

q.P1disp();

q.P2disp();

q.P12disp();

}

}

1. Declare Interface **I1** and **I2** . Interfaces **I3** extends both of these Interfaces. Also declare Interface **I4**. Class **X** implements **I3**. Class **W** extends **X** and implements **I4**. Create an object of class **W.** Use the instanceof operator to test if that object implements each of the Interfaces and is of type **X**.

**interface I1**

**{**

**void display1();**

**}**

**interface I2**

{

void display2();

}

**interface I3 extends I1,I2**

{

void display3();

}

**interface I4**

{

void display4();

}

**class X implements I3**

{

**public void display1()**

{ System.out.println("Invoking interface I1");

}

**public void display2()**

{ System.out.println("Invoking interface I2");

}

**public void display3()**

{ System.out.println("Invoking interface I3");

}

}

**class W extends X implements I4**

{

**public void display4()**

{ display1();

display2();

display3();

System.out.println("Invoking interface I4");

}

}

**class ASS47**

{ **public static void main(String arg[])**

{ W ob=new W();

**I4 ref;**

ref=ob;

ref.display4();

}

}

8. Interface **K1** declares method **K()** and a variable int **K** that is initialized to one. Interface extends K1 declares method **K().** Interface **K3** extends **k2** and declares method **K().** The return type of method **K()** is void in all interfaces. Class **U** implements **K3.** Its version of method **K()** displays the value of int **K** . Instantiate **U** and invoke its method.

**interface K1**

**{**

int k=1;

void K( );

}

**interface K2 extends K1**

{

void K( );

}

**interface K3 extends K2**

**{**

void K( );

}

**class U implements K3**

{

public void K( )

{

System.out.println("The value of the interface K1: "+k);

}

}

**class ass48**

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

{

U u=new U( );

**K3 ref;**

ref=u;

ref.K( );

}

}

**JAVA** **Assignment 5**

**Exception:**

1. Write a program where four types of exception ie. **Arithmetic exception, ArrayIndexOutof bound, string, Null Pointer exception** using different catch block to handle that exception and ensure that program not terminated using final block concept.

**class ASS51**

**{ public static void main(String arg[])**

{ int a=5,b=0;

int c[]={1,2,3};

String s="SHILPI";

int d[]=null;

try{

for(int i=0;i<4;i++)

{ try{

switch(i)

{ case 0:

System.out.println("a= "+a/b);

break;

case 1:

System.out.println(c[5]);

break;

case 2:

System.out.println(s.charAt(8));

break;

case 3:

System.out.println(d[4]);

break;

}

}

catch(ArithmeticException e)

{ System.out.println("Exception: "+e+ " INVALID DIVISION");

}

catch(ArrayIndexOutOfBoundsException e)

{ System.out.println("Exception: "+e+ " CROSSED ARRAY SIZE");

}

catch(StringIndexOutOfBoundsException e)

{ System.out.println("Exception: "+e+ " CROSSED STRING SIZE");

}

catch(NullPointerException e)

{

System.out.println("Exception: "+e+ " INVALID USE OF NULL REFERENCE");

}

}

}

finally

{ System.out.println("Program executing again");

}

System.out.println("QUIT");

}

}

1. Write an program to handle three types of exception **Arithmetic exception, ArrayIndexOut of bound, string Out index** using nested try blocks.

class ASS52

{ public static void main(String arg[])

{ int a=5,b=0;

int c[]={1,2,3};

String s="SHILPI";

**try{**

**try{**

**try{**

System.out.println("a= "+a/b);

**}**

**catch(ArithmeticException e)**

**{**

**System.out.println("Exception: "+e+ " INVALID DIVISION");**

**}**

System.out.println(c[5]);

}

**catch(ArrayIndexOutOfBoundsException e)**

**{**

**System.out.println("Exception: "+e+ " CROSSED ARRAY SIZE");**

**}**

System.out.println(s.charAt(8));

**}**

**catch(StringIndexOutOfBoundsException e)**

**{**

**System.out.println("Exception: "+e+ " CROSSED STRING SIZE");**

**}**

System.out.println("QUIT");

}

}

3. Write a program for user defined exception that checks the internal & external marks. If the internal marks is greater than 30 or less than 0 that raises the exception the invalid internal marks , if the external marks is greater than 70 or less than 0 it raises the exception & display the message the external marks is exceed. Create the above exception & use it in your program.

**class MyException extends Exception**

**{ MyException(String msg)**

{ super(msg);

}

}

**class Check**

{ int a,b;

**Check(int imarks,int emarks)**

{ a= imarks;

b= emarks;

}

**public void display1() throws MyException**

{ if(a<0||a>30) throw new MyException("INVALID INTERNAL MARKS");

else

{ System.out.println("Internal Marks given: "+a);

}

}

**public void display2() throws MyException**

{ if(b<0||b>70) throw new MyException("INVALID EXTERNAL MARKS");

else

{ System.out.println("External Marks given: "+b);

}

}

}

**class ASS53**

**{ public static void main(String arg[])**

{ Check c=new Check(45,65);

**try{**

**try{**

c.display1();

}

**catch(MyException e)**

{ System.out.println(e.getMessage());

}

**try{**

c.display2();

}

**catch(MyException e)**

{ System.out.println(e.getMessage());

}

}

**finally{**

System.out.println("END OF PROGRAM");

}

}

Assignment 6

Multi Threading

1. a. Write an application which creates two threads one child and another is main that is running simultaneously in the system by extending the thread class. The child thread has to display to nos from 1 to 10 and the main thread has to display the number of the child thread multiplied by 5 . Put the name of the child thread and to display its name and overall execution.
   1. Write above program by implently runnable interface that overrides the run method by referring through start method().

**class First extends Thread**

**{**

**public void run()**

**{**

**try{**

for(int i=1;i<=10;i++)

{

System.out.println("Thread i:"+i);

Thread.sleep(100);

}

}

**catch(Exception e)**

**{ }**

System.out.println("Child Thread is over");

}

}

**class Ass61**

**{**

**public static void main(String arg[])**

**{**

System.out.println("Main begins");

System.out.println("Go to other thread");

First f=new First();

f.setName("Mythread\_1");

f.start();

System.out.println("Come back in main thread");

**try{**

for(int k=1;k<=10;k++)

{

System.out.println("Main :"+k\*5);

Thread.sleep(500);

}

}

**catch(Exception e)**

{ }

System.out.println("Main ends"+f.getName());

}

}

**Ans b.**

**class First implements Runnable**

**{**

**public void run()**

**{**

**try{**

for(int i=1;i<=10;i++)

{

System.out.println("Thread i:"+i);

Thread.sleep(100);

}

}

**catch(InterruptedException e)**

{ }

System.out.println("Child Thread is over");

}

}

**class Ass61b**

**{**

**public static void main(String arg[])**

{

System.out.println("Main begins");

System.out.println("Go to other thread");

First f=new First();

Thread t1=new Thread(f);

t1.setName("Mythread\_1");

t1.start();

System.out.println("Come back in main thread");

**try{**

for(int k=1;k<=10;k++)

{

System.out.println("Main :"+k\*5);

**Thread.sleep(500);**

}

}

**catch(Exception e)**

**{** }

System.out.println("Main ends"+t1.getName());

}

}

1. Write an application to implement the three threads by extending thread class.I mplement the method by computing the sum of natural nos in thread1. Sum of square of natural nos. in thread2 and sum of cube of natural nos in thread3. use is Alive () method to test whether the thread is still running or not.

**class First extends Thread**

**{**

**public void run()**

**{**

int s=0;

**try{**

for(int i=1;i<=10;i++)

{

s=s+i;

Thread.sleep(100);

}

System.out.println("Sum of Natural no :"+s);

}

**catch(InterruptedException e)**

{ }

System.out.println(" Child Thread 1 is over");

}

}

**class Second extends Thread**

**{**

**public void run()**

{

int s1=0;

**try{**

for(int i=1;i<=10;i++)

{

s1=s1+i\*i;

Thread.sleep(100);

}

System.out.println("Sum of Square of Natural no:"+s1);

}

**catch(InterruptedException e)**

**{ }**

System.out.println("Child Thread 2 is over");

}

}

**class Third extends Thread**

**{**

**public void run()**

**{**

int s2=0;

**try{**

for(int i=1;i<=10;i++)

{

s2=s2+i\*i\*i;

Thread.sleep(100);

}

System.out.println("Cube of natural no:"+s2);

}

**catch(InterruptedException e)**

{ }

System.out.println("Child Thread 3 is over");

}

}

**class Ass62**

**{**

**public static void main(String arg[])**

**{**

System.out.println("Main begins");

System.out.println("Go to other thread");

First f=new First();

f.setName("Mythread\_1");

Second f1=new Second();

f1.setName("Mythread\_2");

Third f2=new Third();

f2.setName("Mythread\_3");

f.start();

f1.start();

f2.start();

System.out.println("Mythread\_1 is alive:"+f.isAlive());

System.out.println("Mythread\_2is alive:"+f1.isAlive());

System.out.println("Mythread\_3 is alive:"+f2.isAlive());

System.out.println("Come back in main thread");

**try{**

for(int k=1;k<=10;k++)

{

System.out.println("Main :"+k);

Thread.sleep(500);

}

}

**catch(Exception e)**

{ }

System.out.println("Main ends"+f.getName()+" "+f1.getName()+" "+f2.getName());}

System.out.println("Mythread\_1 is alive:"+f.isAlive());

System.out.println("Mythread\_2is alive:"+f1.isAlive());

System.out.println("Mythread\_3 is alive:"+f2.isAlive());

}

1. Write an application evaluating two threads implementing runnable interface to compute the table of 4 & 5 in two threads respectively . Use set priority method to lie the different priority . level5 of two threads and to demonstrate the behavior of its execution.

**class First implements Runnable**

**{**

**public void run()**

**{**

**try{**

for(int i=1;i<=10;i++)

{

System.out.println("4x"+i+"="+4\*i);

Thread.sleep(100);

}

}

**catch(InterruptedException e)**

{ }

System.out.println("Child Thread1 is over");

}

}

**class Second implements Runnable**

**{**

**public void run()**

**{**

**try{**

for(int i=1;i<=10;i++)

{

System.out.println("5x"+i+"="+5\*i);

Thread.sleep(100);

}

}

**catch(InterruptedException e)**

{

}

System.out.println("Child Thread2 is over");

}

}

**class Ass63**

**{**

**public static void main(String arg[])**

**{**

System.out.println("Main begins");

System.out.println("Go back to other thread");

First f=new First();

Thread t1=new Thread(f);

t1.setName("Mythread\_1");

t1.start();

**try**

**{**

t1.join();

}

**catch(InterruptedException e)**

{

}

System.out.println("Mythread\_1 is alive:"+t1.isAlive());

Second f1=new Second();

Thread t2=new Thread(f1);

t2.setName("Mythread\_2");

t2.start();

System.out.println("Mythread\_2is alive:"+t2.isAlive());

**try{**

t2.join();

}

**catch(InterruptedException e)**

**{**

**}**

System.out.println("Mythread\_1 is alive:"+t1.isAlive());

System.out.println("Mythread\_2 is alive:"+t2.isAlive());

System.out.println("Come back in main thread");

**try{**

for(int k=1;k<=10;k++)

{

System.out.println("6x"+k+"="+6\*k);

Thread.sleep(100);

}

}

**catch(Exception e)**

**{ }**

System.out.println("Main ends "+t1.getName()+" " +t2.getName());

}

}

1. Use all the deprecated methods ie .stop(),resume(),suspend() in Problem No3 to demonstrate the behavior of each thread.

**class X extends Thread**

**{**

**public void run()**

**{**

System.out.println("Child Thread 1 is started");

**try{**

for(int i=1;i<=10;i++)

{

System.out.println("X(i): "+i);

Thread.sleep(100);

}

}

**catch(InterruptedException e)**

{ }

System.out.println("Child Thread1 is over");

}

}

**class Y extends Thread**

**{**

**public void run()**

{

System.out.println("Child Thread 2 is started");

**try{**

for(int i=1;i<=10;i++)

{

System.out.println("Y(i): "+i);

Thread.sleep(2000);

}

}

**catch(InterruptedException e)**

**{**

**}**

System.out.println("Child Thread2 is over");

}

}

**class Ass64**

**{**

**public static void main(String arg[])**

**{**

System.out.println("Main begins");

System.out.println("Go back to other thread");

X ThreadX =new X();

ThreadX.setName("Mythread\_1");

Y ThreadY =new Y();

ThreadY.setName("Mythread\_2");

ThreadX.start();

ThreadY.start();

**try{**

Threadx.sleep(500);

Threadx.suspend();

System.out.println("X is suspended");

Threadx.sleep(500);

Threadx.resume();

System.out.println("X is resumed");

Thready.suspend();

System.out.println("y is suspended");

Thready.stop();

System.out.println("y is killed");

}

**catch(Exception e)**

**{ }**

System.out.println("Exit main");

System.out.println("Main ends "+ThreadX.getName()+" " +ThreadY.getName());

}

}

5. Write an application with 3 threads & main thread such that all the threads will be synchronized & ensure that the main thread will finish last after completion of all the three threads. The first child thread will evaluate the table of 2 and the second thread will evaluate table 3 & table 4 respectively. While main thread will display the no from 1 to 10.

**class First extends Thread**

{

int j;

First(int k)

{

j=k;

}

**public void run()**

**{**

**call();**

**}**

**synchronized void call()**

**{**

**try{**

for(int i=1;i<=10;i++)

{

System.out.println(j+"X"+i+"="+i\*j);

Thread.sleep(100);

}

}

**catch(Exception e)**

**{ }**

}

}

**class Ass65**

**{**

**public static void main(String arg[])**

{

System.out.println("Main begins");

System.out.println(" Go back to other thread" );

First t1=new First(2);

First t2=new First(3);

First t3=new First(4);

**try{**

t1.start();

t1.join();

t2.start();

t2.join();

t3.start();

t3.join();

}

**catch(InterruptedException e)**

**{ }**

System.out.println("Come back in main thread");

**try{**

for(int k=1;k<=10;k++)

{

System.out.println("Main"+k);

Thread.sleep(100);

}

}

**catch(Exception e)**

{ }

System.out.println("Exit main");

}

}

**Assignment 7**

**Applet Programming**

1. Write an application which displays 13 vertical colour bars.

import java.applet.\*;

import java.awt.\*;

/\*

<applet code ="Ass81" width=500 height=100>

</applet>

\*/

**public class Ass71 extends Applet**

**{**

String msg ="This is my applet";

**public void init()**

**{**

setBackground(Color.white);

setForeground(Color.red);

}

**public void start()**

**{}**

**public void paint(Graphics g**)

{

g.setColor(Color.red);

g.fillRect(10,10,60,50);

g.setColor(Color.blue);

g.fillRect(70,10,60,50);

g.setColor(Color.cyan);

g.fillRect(130,10,60,50);

g.setColor(Color.yellow);

g.fillRect(190,10,60,50);

g.setColor(Color.pink);

g.fillRect(250,10,60,50);

g.setColor(Color.red);

g.fillRect(310,10,60,50);

g.setColor(Color.black);

g.fillRect(370,10,60,50);

g.setColor(Color.darkGray);

g.fillRect(430,10,60,50);

g.setColor(Color.white);

g.fillRect(490,10,60,50);

g.setColor(Color.green);

g.fillRect(550,10,60,50);

g.setColor(Color.orange);

g.fillRect(610,10,60,50);

g.setColor(Color.magenta);

g.fillRect(670,10,60,50);

g.setColor(Color.lightGray);

g.fillRect(730,10,60,50);

g.drawString(msg,10,30);

}

}

1. Write an application that plots f(x) sin(x)+ ½ sin(2x)+ 1/3 sin(3x) + ¼ sin (4x)

import java.applet.\*;

import java.awt.\*;

/\*<applet code="Ass82" width=500 height=100>

</applet>

\*/

**public class Ass72 extends Applet**

**{**

String msg="this is my String";

**public void init()**

**{**

setBackground(Color.cyan);

setForeground(Color.red);

}

**public void start()**

**{}**

**public void paint(Graphics g)**

{

double x,y;

int x1,y1;

for(x=1;x<=50;x++)

{

y=Math.sin(x)+(1/2)\*Math.sin(2\*x)+(1/3)\*Math.sin(3\*x)+(1/4)\*Math.sin(4\*x);

y=10\*y;

x1=(int)x\*10;

y1=(int)y;

g.drawString("\*",x1,y1+100);

g.drawString(msg,10,30);

}

}

}

1. Write an application that displays the background color to cyan & draw a circle of radius 50 pixels centered in the middle of the applets . The dimension of the applet should 120x120 pixels. Fill the circle with magenta color.

import java.applet.\*;

import java.awt.\*;

/\*<applet code="Ass83" width=120 height=100>

</applet>

\*/

**public class Ass83 extends Applet**

**{**

String msg="this is a CIRCLE";

**public void init()**

**{**

setBackground(Color.cyan);

setForeground(Color.red);

}

**public void start()**

{ }

**public void paint(Graphics g)**

**{**

g.drawString(msg,10,30);

g.setColor(Color.magenta);

g.fillOval(30,30,60,60);

}

}

1. Write an application that contains one button initialize the label of the button to “start”. When the user press the button change the label to “stop”. Toggle the button label between these 2 values each time button is pressed.

import java.awt.\*;

import java.applet.\*;

import java.awt.event.\*;

**public class Ass74 extends Applet implements ActionListner**

**{**

String msg="BUTTON DEMO";

Button b;

**public void init()**

**{**

b=new Button("START");

add(b);

b.addActionListner(this);

}

**public void Start(){}**

**public void actionPerformed(ActionEvent ae)**

{

String str=ae.getActionCommand();

if(str.equalsIgnoreCase("START"))

{

b.setLabel("STOP");

msg="You Pressed Start";

}

if(str.equalsIgnoreCase("STOP"))

{

b.setLabel("START");

msg="You Pressed Stop";

}

repaint();

}

public void paint(Graphics g)

{

g.drawString(msg,10,100);

}

}

1. Write an application that contains three check box & a canvas . The three checkbox should be label “red”,”green”,”blue ”. The selection of checkbox determines the color of the canvas.

import java.awt.\*;

import java.applet.\*;

import java.awt.event.\*;

**public class Ass75 extends Applet implements ItemListner**

**{**

String msg=" ";

Checkbox cred,cgreen,cblue;

**public void init()**

**{**

cred=new Checkbox();

add(cred);

cgreen=new Checkbox();

add(cgreen);

cblue=new Checkbox();

add(cblue);

cred.addItemListner(this);

cgreen.addItemListner(this);

cblue.addItemListner(this);

}

**public** **void start(){}**

**public void itemStateChanged(ItemEvent ie)**

{

repaint();

}

**public void paint(Graphics g)**

{

Canvas c=new Canvas();

Graphics gc=c.getGraphics();

if(cred.getState())

{

setBackcolor(Color.red);

}

if(cgreen.getState())

{

setBackground(Color.green);

}

if(cblue.getState())

{

setBackground(Color.blue);

}

if(cred.getState() && cgreen.getstate())

{

setBackground(new Color(255,255,0);

}

if(cgreen.getState() && cblue.getstate() )

{

setBackground(new Color(0,255,255);

}

if(cred.getState() && cblue.getstate() )

{

setBackground(new Color(255,0,255));

}

if(cred.getState() && cgreen.getstate() && cblue.getstate())

{

setBackground(new Color(255,255,255));

}

if(cred.getState() && cgreen.getstate() && cblue.getstate())

{

setBackcolor(new Color(0,0,0));

}

} }

1. Write an application that contains three horizontal scrolls bars and a canvas . The scroll bars adjust the red green & blue components of the canvas color.

import java.applet.\*;

import java.awt.\*;

import java.awt.event.\*;

**public class Ass76 extends Applet implements AdjustmentListner**

**{**

Scrollbar sb1,sb2,sb3;

**public void init()**

**{**

sb1=new Scrollbar(Scrollbar.HORIZONTAL,0,1,0,255);

sb2=new Scrollbar(Scrollbar.HORIZONTAL,0,1,0,255);

sb3=new Scrollbar(Scrollbar.HORIZONTAL,0,1,0,255);

add(sb1);

add(sb2);

add(sb3);

sb1.addAdjustmentListener(this);

sb2.addAdjustmentListener(this);

sb3.addAdjustmentListener(this);

}

**public void adjustmentValueChanged(AdjustmentEvent ae)**

**{**

repaint();

}

**public void Paint(Graphics g)**

**{**

Canvas c=new Canvas();

setBackground(new Color(sb1.getValue()));

setBackground(new Color(sb2.getValue()));

setBackground(new Color(sb3.getValue()));

}

}

1. Write an application which displays two rectangle initialize with cyan & magenta. Place 2 pairs of button labeled with red & blue. Under the 2 rectangle pressing of each button will change the color of corresponding rectangles.

import java.applet.\*;

import java.awt.\*;

import java.awt.event.\*;

**public class Ass77 extends Applet implements ActionListener**

**{**

Button str;

Button b1,b2,b3,b4;

**public void init()**

{

b1=new Button("Red");

b2=new Button("Blue");

b3=new Button("Red");

b4=new Button("Blue");

add(b1);

add(b2);

add(b3);

add(b4);

b1.addActionListener(this);

b2.addActionListener(this);

b3.addActionListener(this);

b4.addActionListener(this);

}

**public void start(){}**

**public void actionPerformed (ActionEvent ae)**

**{**

str=(Button)ae.getSource();

repaint();

}

**public void paint(Graphics g)**

{

if(str==b1)

{

g.setColor(Color.red);

g.fillRect(5,5,50,30);

}

if(str==b2)

{

g.setColor(Color.blue);

g.fillRect(5,5,50,30);

}

if(str==b3)

{

g.setColor(Color.red);

g.fillRect(5,100,50,30);

}

if(str==b4)

{

g.setColor(Color.blue);

g.fillRect(5,100,50,30);

}

}

}

1. Write an application that contains label with the string “Name” with 15 character text field “Address” with 20 character text field and “Password” with 8 character text field which is initially empty. Allow the user to change either of these fields , the each character . The each character for the password field should be \*.

import java.applet.\*;

import java.awt.\*;

import java.awt.event.\*;

**public class Ass78 extends Applet implements ActionListener**

**{**

TextField name,pass,adds;

**public void init()**

**{**

Label namep=new Label("Name",Label.RIGHT);

Label passp=new Label("Password",Label.RIGHT);

Label addp=new Label("Address",Label.RIGHT);

name=new TextField(20);

pass=new TextField(20);

adds=new TextField(20);

pass.setEchoChar('\*');

add(namep);

add(name);

add(passp);

add(pass);

add(addp);

add(adds);

name.addActionListener(this);

pass.addActionListener(this);

adds.addActionListener(this);

}

**public void start(){}**

**public void actionPerformed(ActionEvent ae)**

**{**

repaint();

}

**public void paint(Graphics g)**

**{**

g.drawString("Name:"+name.getText(),6,60);

g.drawString("Pass:"+pass.getText(),6,100);

g.drawString("Selected:"+name.getText(),6,90);

g.drawString("Address:"+adds.getText(),6,180);

}

}