Program

1.write a program to find the number of and sum of all integer number that is greater than 100 and less than 200 that are divisible is 7.

Code:

```
class test
{
        public static void  main(String[] args)
      {
                int count=0,sum=0,i;
                for(i=101;i<200;i++)
                {
                        if(i%7==0)
                        {
                          count++;
                          sum=sum+i;
                        }
                        System.out.println("the tital elements are:"+count);
                        System.out.println("the sum of elements is:"+sum);
        }
}
```

2.write a program to print first n Fibonacci number

```
class test
{
        public static void  main(String[] args)
      {
                int n,a=0,b=1,c,i;
                n=Integer.parseInt(args[0]);
                System.out.println("Fibonacci number are:");
                for(i=1;i<=n;i++)
                {
                        c=a+b;
                        System.out.println(" "+a);
                        a=b;
                        b=c;
                }
        }
}
```

3.write a program that create and initializes a four integer element array. Calculate and display the average of its values.

Code:

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

```
	{
		int a={10,20,30,40};
		double sum=0,avg;
		for(int i=0;i<4;i++)
		{
			sum=sum+a[i];
		}
		avg=sum/4;
		System.out.println("sum= "+sum +"avg="+avg);
	}
}
```

4.Write a program to count occurrence of character in a string.

```
class CountChar
{
	public static void main(String[] args)
	{
		String S="javaisplatformindependent";
		int l=S.length();
		int count=0; char
		c;
		for(int i=0; i<l;i++)
		{
			c=S.charAt(i);
			if(c=='a'||c=='A')
			{
				count++;
			}
		}
		System.out.println("The Occurrence of 'a' is: "+count);
	}
}
```

5.Write a program to check the given string is palindrome or not.

Code:

```
import java.util.*;
class test
{
	public static void main(String s[])
	{
		String str,rev="";
		Scanner sc=new Scanner(System.in);
		str=sc.nextLine();
		int n=str.length();
		for(int i=n-1;i>=0;i--)
		{
			rev=rev+str.charAt(i);
		}
		if(str.equals(rev))
		{
```

```
                        System.out.println("the string is palindrom");
                }
                else
                {
                        System.out.println("the string is not palindrom");
                }
        }
}
```

6.Write a program to read a line from command line and print that line in reverse.

Code:

```
class test
{
        public static void main(String args[])
        {
                String str="",rev="";
                int i;
                for(i=0;i<args.length;i++)
                {
                        str=str+args[i];
                }
                int n=str.length();
                for(i=n-1;i>=0;i--)
                {
                        rev=rev+str.charAt(i);
                }
                System.out.println(rev);
        }
}
```

7) write a program to create circle class with area function to find area of circle.

Code:

```
class circle
{
        double ar,r;
        circle(double x)
        {
                r=x;
        }
        void area()
        {
                ar=3.14*r*r;
                System.out.println("area of circle="+ar);
        }
}
```

```
class test
{
        public static void main(String args[])
        {
                circle c1=new circle(5);
                c1.area();
        }
}
```

8) Design a class named Fan to represent a fan. The class contains:

- Three constants named SLOW, MEDIUM and FAST with values 1, 2 and 3 to denote the fan speed.

- An int data field named speed that specifies the speed of the fan (default SLOW).

- A boolean data field named f_on that specifies whether the fan is on (default false).

- A double data field named radius that specifies the radius of the fan (default 4).

- A data field named color that specifies the color of the fan (default blue).

- A no-arg constructor that creates a default fan.

- A parameterized constructor initializes the fan objects to given values.

- A method named display() will display description for the fan. If the fan is on, the display() method displays speed, color and radius. If the fan is not on, the method returns fan color and radius along with the message "fan is off". Write a test program that creates two Fan objects. One with default values and the other with medium speed, radius 6, color brown, and turned on status true. Display the descriptions for two created Fan objects.

Code:

```
class fan
{
        final int slow=1;
        final int medium=2;
        final int fast=3;
        int speed;
        boolean f_on;
        double radius;
        String color;
        fan()
        {
                speed=slow;
                f_on=false;
                radius=4;
                color="blue";
        }
        fan(int s,booleanf,doubler,String s1)
        {
                speed=s;
```

```
		f_on=f;
		radius=r;
		color=s1;
	}
	void show()
	{
		if(f_on)
		{
			System.out.println("FAN IS ON");
			System.out.println(speed);
			System.out.println(radius);
			System.out.println(color);
		}
		else
		{
			System.out.println("FAN IS OFF");
			System.out.println(radius);
			System.out.println(color);
		}
	}	}
class test
{
	public static void main(String s[])
	{
		fan f1=new fan();
		fan f2=new fan(2,true,6,"brown");
		f1.show();
		f2.show();
	}
}
```

9) Define the Rectangle class that contains:
Two double fields x and y that specify the center of the rectangle, the data field width and height, A no-arg constructor that creates the default rectangle with (0,0) for (x,y) and 1 for both width and height. 2 A parameterized constructor creates a rectangle with the specified x,y,height and width.
-A method getArea() that returns the area of the rectangle.
-A method getPerimeter() that returns the perimeter of the rectangle.
-A method contains(double x, double y) that returns true if the specified point (x,y) is inside this rectangle.
Write a test program that creates two rectangle objects. One with default values and other with user specified values. Test all the methods of the class for both the objects.

```
class rectangle
{
	double cx,cy,height,width,xleft,xright,yup,ydown;
	rectangle()
	{
		cx=0;
		cy=0;
		height=1;
		width=1;
	}
	rectangle(double x,doubley,doubleh,double w)
	{
		cx=x;
		cy=y;
		height=h;
		width=w;
	}
	double getarea()
	{
		return (height*width);
	}
	double getperimeter()
	{
		return (2*(height+width));
	}
	boolean contains(double x,double y)
	{
		xleft=(cx-(width/2));
		xright=(cx+(width/2));
		yup=(cy+(height/2));
		ydown=(cy-(height/2));
		if((x>xleft&& x<xright) && (y<yup && y>ydown))
		{
			return true;
		}
		else
		{
			return false;
		}

	}
}
class test
{
	public static void main(String s[])
	{
		rectangle r1=new rectangle();
		rectangle r2=new rectangle(10,30,10,20);
		System.out.println(r1.getarea());
		System.out.println(r2.getarea());
		System.out.println(r1.getperimeter());
		System.out.println(r2.getperimeter());
		if(r1.contains(0,0))
		{
			System.out.println("point is inside the rectangle");
		}
		else
		{
			System.out.println("point is outside the rectangle");
		}
		if(r2.contains(250,310))
		{
			System.out.println("point is inside the rectangle");
		}
		else
		{
			System.out.println("point is outside the rectangle");
		}
	}
}
```

Output:

```
1.0
200.0
4.0
60.0
point is inside the rectangle
point is outside the rectangle
```

10) Declare a class called coordinate to represent 3 dimensional Cartesian coordinates(x, y, and z) define following method.
- Constructor
- Display to print values of members
- Add_coordinates, to add three such coordinates object to produce a resultant coordinates object. Generate and hendle exception if x,y and z coordinates of the result are zero
- Main , to show use of above method

Code:

```
class  coordinate extends Exception
{
	double x,y,z;
	coordinate()
	{
		x=0;
		y=0;
		z=0;
	}
	coordinate(double a,double b,double c)
	{
		x=a;
		y=b;
		z=c;
	}
	void  display()
	{
		System.out.println("X="+x);
```

```
                System.out.println("Y="+y);
                System.out.println("Z="+z);
        }
coordinate add_coordinates(coordinate obj1,coordinate obj2,coordinate obj3) throws coordinate
        {
                coordinate obj4=new coordinate();
                obj4.x=obj1.x+obj2.x+obj3.x;
                obj4.y=obj1.y+obj2.y+obj3.y;
                obj4.z=obj1.z+obj2.z+obj3.z;
                if(obj4.x==0 || obj4.y==0 || obj4.z==0)
                {
                        throw new coordinate();
                }
          return obj4;
        }
}
class test
{
        public static void  main(String[] args)
      {
                coordinate c1= new  coordinate();
                coordinate c2= new  coordinate(1,2,3);
                coordinate c3= new  coordinate(4,5,6);
                coordinate c4= new  coordinate(7,8,9);
                try
                {
                        c1=c1.add_coordinates(c2,c3,c4);
                        c1.display();
                }
                catch(coordinate c)
                {
                        System.out.println("exception for zero value");
                }
        }
}
```

11) Define time class with hour and minute. Also define addition method to add two time objects.

Code:

```
class time
{
        int hour,minute;
        time()
        {
                hour=0;
                minute=0;
        }

        time(int a,int b)
        {
```

```
			hour=a;
			minute=b;
		}
		void add(time obj1,time obj2)
		{
			hour=obj1.hour+obj2.hour;
			minute=obj1.minute+obj2.minute;

			hour=hour+(minute/60);
			minute=minute%60;
		}
		void show()
		{
			System.out.println("hour="+hour);
			System.out.println("minute="+minute);
		}
 }
 class test
 {
		public static void main(String args[])
		{
			time t1=new time(10,20);
			time t2=new time(100,300);
			time t3=new time();
			t3.add(t1,t2);
			t3.show();
		}
 }
```

12) It is required to compute SPI (semester performance index) of n students of your college for their registered subjects in a semester.
Declare a class called student having following data members: id_no, no_of_subjects_registered, subject_code, subject_credits, grade_obtained and spi.
- Define constructor and calculate_spi methods.
- Define main to instantiate an array for objects of class student to process data of n students to be given as command line arguments.
Code:

```
import java.util.Scanner;
class Student
{
		int id_no;
		int no_of_subjects_registered;
		int total_credit=0;
		int sub_code[]=new int[10];
		int sub_credit[]=new int[10];
		int temp[]=new int[10];
		int g_point[]=new int[10];
		String grade_obtained;
		String grade_obt[]=new String[10];
		float spi=0;
```

```
	Student(int id, int no_sub)
	{
		id_no=id;
		no_of_subjects_registered=no_sub;
	}
	void get_subdata(int n,int s_code,int s_credit,String g_obt)
	{
		sub_code[n]=s_code;
		sub_credit[n]=s_credit;
		grade_obt[n]=g_obt;
		if(grade_obt[n].equals("AA"))
		{
			g_point[n]=10;
		}
		else if(grade_obt[n].equals("AB"))
		{
			g_point[n]=9;
		}
		else if(grade_obt[n].equals("BB"))
		{
			g_point[n]=8;
		}
		else if(grade_obt[n].equals("BC"))
		{
			g_point[n]=7;
		}
		else if(grade_obt[n].equals("CC"))
		{
			g_point[n]=6;
		}
		else if(grade_obt[n].equals("CD"))
		{
			g_point[n]=5;
		}
		else if(grade_obt[n].equals("DD"))
		{
			g_point[n]=4;
		}
		else if(grade_obt[n].equals("FF"))
		{
			g_point[n]=0;
		}
	}
	void student_details()
	{
		System.out.println("");
		System.out.println("Student id:"+id_no);
		System.out.println("");
		System.out.println("No of Subjects:"+no_of_subjects_registered);
		System.out.println("\tSub Code\tSub Credit\tGrade obtained");
		for(int i=0;i<no_of_subjects_registered;i++)
		{
	System.out.println("\t"+sub_code[i]+"\t\t"+sub_credit[i]+"\t"+grade_obt[i]);
		}
	}

	void count_spi()
	{
		int ans=0;
		for(int i=0;i<no_of_subjects_registered;i++)
		{
			temp[i]=sub_credit[i]*g_point[i];
			ans=ans+temp[i];
			total_credit=sub_credit[i]+total_credit;
		}

	spi=ans/total_credit;
	System.out.println(ans+""+total_credit);
	System.out.println("Congratulations Your SPI is: "+spi);
	System.out.println("");
	System.out.println("");
	}
}

class test
{
	public static void main(String[] args)
	{
		int num=Integer.parseInt(args[0]);
		Student s[]=new Student[num];
		Scanner sc=new Scanner(System.in);
		for(int i=0;i<num;i++)
		{
			int z=i+1;
			System.out.println("");
			System.out.println("Enter the Details for Student num:"+z);
			System.out.println("");
			System.out.println("Enter Student ID:");
			int id=sc.nextInt();
			System.out.println("Enter Number of Subjects:");
			int sub=sc.nextInt();
			s[i]=new Student(id,sub);
		for(int j=0;j<sub;j++)
		{
			System.out.println("");
			int y=j+1;
			System.out.println("For Subject"+y); System.out.println("");
			System.out.println("Enter Subject Code:");
```

```
                              int s_cod=sc.nextInt();
                              System.out.println("Enter Subject Credit:");
                              int s_cre=sc.nextInt();
                              sc.nextLine();
                              System.out.println("Enter Grade:");
                              String s_gra=sc.next();
                              s[i].get_subdata(j,s_cod,s_cre,s_gra);
                    }
                    s[i].student_details();
                    s[i].count_spi();
                    }
          }
}
```

Inheritance

1) Write a JAVA program to create a super class called figure that storesthe dimensions of a two-dimensional object. It also defines a method called area () that computes the area of an object. The program derives two sub classes from figure. The first is rectangle and the second is Triangle. Each of these subclasses overrides area (), so that it returns the area of a rectangle and a triangle respectively.

```
class figure
{
        double x,y;
        double area()
        {
                return 0;
        }
}
class rectangle extends figure
{
        rectangle(double a,double b)
        {
                x=a;
                y=b;
        }
        double area()
        {
                return (x*y);
        }
}
class triangle extends figure
{
        triangle(double a,double b)
        {
                x=a;
                y=b;
        }
        double area()
        {
                return 2*(x+y);
        }
}
class test1
{
        public static void main(String s[])
        {
                rectangle r1=new rectangle(10,4);
                triangle t1=new triangle(4,5);
                System.out.println("the area of rectangle="+r1.area());
                System.out.println("the area of triangle="+t1.area());
        }
}
```

2) The abstract Vegetable class has three subclasses named Potato, Brinjal and Tomato. Write an application that demonstrates how to establish this class hierarchy. Declare one instance variable of type String that indicates the color of a vegetable. Create and display instances of these objects. Override the toString() method of Object to return a string with the name of the vegetable and its color.

```
Abstract class Vegetable
{
	String vegColour;
	abstract public String toString();
}

class Potato extends Vegetable
{
	public String toString()
	{
		vegColour="Yellow";
		return vegColour;
	}
}

class Brinjal extends Vegetable
{
	public String toString()
	{
		vegColour="Violet";
		return vegColour;
	}
}
class Tomato extends Vegetable
{
	public String toString()
	{
		vegColour="Red";
		return vegColour;
	}
}
class VegetableMain
{
	public static void main(String[] args)
	{
		Vegetable p=new Potato();
		Vegetable b=new Brinjal();
		Vegetable t=new Tomato();
		System.out.println("The Colour of the Potato is: "+p.toString());
		System.out.println("The Colour of the Brinjal is: "+b.toString());
		System.out.println("The Colour of the Tomato is: "+t.toString());

	}
}
```

3)Describe abstract class called Shape which has three subclasses say Triangle,Rectangle,Circle. Define one method area() in the abstract class and override this area() in these three subclasses to calculate for specific object i.e. area() of Triangle subclass should calculate area of triangle etc. Same for Rectangle and Circle.

Code:

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

class Circle extends Shape
{
                    double r;
                    Circle(double R)
                    {
                       r=R;
                    }
                    void Area()
                    {
                       System.out.println("Area of Circle="+(3.14*r*r));
                    }

}
class Triangle extends Shape
{                   int b,h;
                    Triangle(int B,int H)
                    {
                       b=B;
                       h=H;
                    }
                    void Area()
                    {
                       System.out.println("Area of Triangle="+0.5*h*b);
                    }
}
class Rectangle extends Shape
{                   int b,l;
                    Rectangle(int L,int B)
                    {
                       l=L;
                       b=B;
                    }
                    void Area()
                    {
                       System.out.println("Area of Rectangle="+l*b);
                    }
}
class pb6
{
```

```
                public static void main(String args[])
                {
                   Circle c1= new Circle(3.0);
                   Triangle t1= new Triangle(5,6);
                   Rectangle r1 =new Rectangle(9,10);
                   c1.Area();
                   t1.Area();
                   r1.Area();
                }
}
```

4) Write a program to define abstract class, with two methods addition() and subtraction(). addition() is abstract method. Implement the abstract method and call that method using a program(s).

Code:

```
abstract class A
{
        int a=20,b=10,c;
        abstract void addition();
        void subtraction() {  }
}
class demo extends A
{
        void addition()
        {
                c=a+b;
                System.out.println("sum="+c);
        }
        void subtraction()
        {
                c=a-b;
                System.out.println("sub="+c);
        }
}
class test
{
        public static void main(String s[])
        {
                demo d1=new demo();
                d1.addition();
                d1.subtraction();
        }
}
```

5) Write a program that illustrates interface inheritance. Interface P is extended by P1 And P2. Interface P12 inherits from both P1 and P2.Each interface declares one

constant and one method. Class Q implements P12.Instantiate Q and invokes each of its methods. Each method displays one of the constants.

```
interface P
{
        int c=10;
        void display();

}
interface P1 extends P
{
        int c1=11;
        void display();
}
interface P2 extends P
{
        int c2=12;
        void display();
}
interface P12 extends P1,P2
{
        void display();
}
class Q implements P12
{
        public void display()
        {
                System.out.println(c);
                System.out.println(c1);
                System.out.println(c2);
        }
}
class pb1
{
        public static void main(String args[])
        {
                P p;
                p=new Q();
                p.display();
        }
}
```

6) The Transport interface declares a deliver() method. The abstract class Animal is the superclass of the Tiger, Camel, Deer and Donkey classes. The Transport interface is implemented by the Camel and Donkey classes. Write a test program that initialize

an array of four Animal objects. If the object implements the Transport interface, the deliver() method is invoked.

Code:

```
interface Transport
{
void deliver();
}
abstract class Animal
{
 void show() { }
}
class Tiger extends Animal
{
        public void show ()
        {
                System.out.println("This is the Tiger Method!!!");
        }
}
class Camel extends Animal implements Transport
{
        public void show ()
        {
                System.out.println("This is the Camel Method!!!");
        }
        public void deliver()
        {
                System.out.println("Hey Camel! We need to Deliver you to your Owner!!!");
        }
}
class Deer extends Animal
{
        public void show ()
        {
                System.out.println("This is the Deer Method!!!");
        }
}
class Donkey extends Animal implements Transport
{
        public void show ()
        {
                System.out.println("This is the Donkey Method!!!");
        }
        public void deliver()
        {
                System.out.println("Hey Donkey! We also need to Deliver you to your
Owner!!!");
        }
}
class AnimalMain
{
        public static void main(String[] args)
        {
                Animal a1[]=new Animal[4];
                a1[0]=new Tiger();
                a1[1]=new Camel();
                a1[2]=new Deer();
                a1[3]=new Donkey();
                for(int i=0;i<4;i++)
                {
                        a1[i].show();
                }
                Camel c1=new Camel();
                Donkey d1=new Donkey();
                c1.deliver();
                d1.deliver();
        }
}
```

7) Write a program to demonstrate combination of both types of inheritance as shown in figure 1. i.e. hybrid inheritance.

$$\begin{tikzcd} A \arrow[dr] & & B \arrow[dl] \\ & C \arrow[d] & \\ & D & \end{tikzcd}$$

Code:

```
class A
{
        int a=10;
}
interface B
{
        int b=20;
}
class C extends A implements B
{
        void show()
        {
                System.out.println(a);
                System.out.println(b);
        }
}
```

```
class D extends C
{
	void add()
	{
		int c;
		c=a+b;
		System.out.println("sum="+c);
	}
}
class test
{
	public static void main(String s[])
	{
		C c1=new C();
		c1.show();
		D d1=new D();
		d1.add();
	}
}
```

8) Write a program to demonstrate the multipath inheritance for the classes having relations as shown in figure 2 A-> (B, C) ->D.

Code:

```
interface A
{
	int a=10;
}
interface B extends A
{
	int b=a+20;
}
class C implements A
{
	public void show()
	{
		System.out.println(a);
	}
}
class D extends C implements B
{
	public void show()
	{
		System.out.println(a);
		System.out.println(b);
	}
}
class test
{
	public static void main(String s[])
	{
		C c1=new C();
		c1.show();
		D d1=new D();
		d1.show();
	}
}
```

9) Declare a class called book having author_name as private data member. Extend book class to have two sub classes called book_publication&paper_publication. Each of these classes have private member called title. Write a complete program to show usage of dynamic method dispatch (dynamic polymorphism) to display book or paper publications of given author. Use command line arguments for inputting data

Code:

```
class Book
{
	private String authorName;
	public void setName(String authorName)
	{
		this.authorName = authorName;
	}
	public String getName()
	{
	  return this.authorName;
	}
	public void setTitle(String title)
	{
		//Will override by Sub classes
	}
	public void display()
	{
		//Will override by Sub classes
	}
}
class BookPublication extends Book
{
	private String title;
	public String getTitle()
	{
		return title;
	}
	public void setTitle(String title)
	{
		this.title = title;
	}
	public void display()
	{
	  System.out.println("author name is : " + getName() + " book title is : " + this.title);
	}
}
```

```
class PaperPublication extends Book
{
	private String title;
	public String getTitle()
	{
		return title;
	}
	public void setTitle(String title)
	{
		this.title = title;
	}
	public void display()
	{
	   System.out.println("author name is : " + getName() + " paper title is : " +this.title);
	}
}
class BookPaperPublicationDemo
{
	public static void main(String a [])
	{
		Book b = null;
		BookPublication bp = new BookPublication();
		PaperPublication pp = new PaperPublication();
		b = bp;
		b.setName(a[0]);
		b.setTitle(a[1]);
		b.display();
		b = pp;
		b.setName(a[2]);
		b.setTitle(a[3]);
		b.display();
	}
}
```

10) Declare a class called employee having employee_id and employee_name as members. Extend class employee to have a subclass called salary having designation and monthly_salary as members. Define following: - Required constructors - A method to find and display all details of employees drawing salary more than Rs. 20000/-. - Method main for creating an array for storing these details given as command line arguments and showing usage of above methods.

```
class employee
{
	int eid;
	String ename;
}
class salary extends employee
{
	String des;
	int msalary;
	salary(String s1,String s2,String s3,String s4)
	{
		eid=Integer.parseInt(s1);
		ename=s2;
		des=s3;
		msalary=Integer.parseInt(s4);
	}
	void display()
	{
		if(msalary>20000)
		{
			System.out.println(eid);
			System.out.println(ename);
			System.out.println(des);
			System.out.println(msalary);
		}
	}
}
class test
{
	public static void main(String args[])
	{
		salary s1[]=new salary[5];
		int i,n=0;
		for(i=0;i<2;i++)
		{
			s1[i]=new salary(args[n],args[n+1],args[n+2],args[n+3]);
			n=n+4;
		}
		for(i=0;i<2;i++)
		{
			s1[i].display();
		}
	}
}
```

Output: V:\j>java test 1 ankit assi 23000 2 patel prof 12000

```
1
ankit
assi
23000
```

Exeception

1) Write an application that generates custom exception if first argument from command line argument is 0.

```
class Zero extends Exception
{
                Zero()
                {
                        System.out.println("Zero Exception Generated");
                }
}
class pb3
{
                public static void main(String args[])
                {
                        System.out.println("***Exception Demo***");
                        try
                        {
                          if (Integer.parseInt(args[0])==0)
                                throw new Zero();
                        }
                        catch(Zero z)
                        {
                                System.out.println("Please give some another number");
                        }
                }
}
```

2) Write an application that generates custom exception if any value from its command line arguments is negative.

```
class Negative extends Exception
{
        Negative()
        {
                System.out.println("Negative Exception generated");
        }
}
class pb4
{
        public static void main(String args[])
        {
         try{
                for(int i=0;i<args.length;i++)
                {
                                if(Integer.parseInt(args[i]) < 0)
                                {
                                        throw new Negative();
                                }
                }
           }
```

```
          catch(Negative n)
          {
               System.out.println("Any one or more of your command line args is negative");

          }
     }
}
```

3) Write a method for computing x^y by doing repetitive multiplication. x and y are of type integer and are to be given as command line arguments. Raise and handle exception(s) for invalid values of x and y. Also define method main. Use finally in above program and explain its usage.

```
class RepeatativeMultiplication
{
     public static void main(String[] args)
     {
          int x,y,z;
          String s1=args[0], s2=args[1];
          try
          {
               x=Integer.parseInt(s1);
               y=Integer.parseInt(s2);
               z=1;
               for(int i=0; i<y;i++)
               {
                    z=z*x;
               }
               System.out.println(z);
          }
          catch(NumberFormatException e)
          {
               System.out.println("Number Format Exception:"+e);
          }
          finally
          {
               System.out.println("This will Execute Every Time!!!");
          }
     }
}
```

4) Write a program to create user define exception MyException. Define a class ExceptionDemo that has a method named compute( ) which throws a MyException object, when compute( )'s integer parameter is greater than 10.

```
class myexception extends Exception
{
     myexception()
     {
          System.out.println("Custom Exception");
     }
}
```

```
class exceptiondemo
{
        void compute(int n)
        {
                try{
                        if(n>10)
                        {
                          throw new myexception();
                        }
                }
                catch(myexception e)
                {
                        System.out.println("no is greater than 10");
                }
        }
        public static void main(String s[])
        {
                exceptiondemo d1=new exceptiondemo();
                d1.compute(11);
        }
}
```

5) Write a complete program to accept N integer numbers from the command line. Raise and handle exceptions for following cases :
- when a number is –ve
- when a number is evenly divisible by 10
- when a number is greater than 1000 and less than 2000
- when a number is greater than 7000
Skip the number if an exception is raised for it, otherwise add it to find total sum.

```
class negative extends Exception
{
        negative()
        {
                System.out.println("Nagative Exception");
        }
}
class divbyten extends Exception
{
        divbyten()
        {
                System.out.println("Divide by 10 Exception");
        }
}
class between extends Exception
{
        between()
        {
                System.out.println("No between 1000 and 2000 Exception");
        }
}
```

```
class greater extends Exception
{
	greater()
	{
		System.out.println("No greater than 7000 Exception");
	}
}
class test
{
	public static void main(String args[])
	{
		int i,n=0,sum=0;
		for(i=0;i<args.length;i++)
		{
			try
			{
				n=Integer.parseInt(args[i]);
				if(n<0)
				{
				  throw new negative();
				}
				else if(n>1000 && n<2000)
				{
				  throw new between();
				}
				else if(n>7000)
				{
				  throw new greater();
				}
				else if(n%10==0)
				{
				  throw new divbyten();
				}
				else
				{
				  sum=sum+n;
				}
			}
			catch(negative obj)
			{
				System.out.println("no is ="+n);
			}
			catch(divbyten obj)
			{
				System.out.println("no is ="+n);
			}
			catch(between obj)
			{
				System.out.println("no is ="+n);
			}
			catch(greater obj)
			{
				System.out.println("no is ="+n);
			}
		}
		System.out.println("sum="+sum);
	}
}
```

6) It is required to maintain and process the status of total 9 resources. The status value is to be stored in an integer array of dimension 3x3. The valid status of a resource can be one of the 2 followings:
free: indicated by integer value 0
occupied: indicated by integer value 1
inaccessible: indicated by integer value 2
Declare a class called ResourcesStatus, having data member called statusRef, referring to a two dimensional array (3x3) of integers to be used to refer to the above mentioned status values.Define a member method called processStausCount that counts and displays total number of free resources, total number of occupied resources and total number of inaccessible resources. The exception to be raised and handled if total number of occupied resources exceeds total number of free resources. The handler marks status of all inaccessible resources as free.
Accept initial status values from command line arguments and initialize the array. Raise and handle user defined exception if invalid status value given.

```
class invalid extends Exception
{
	invalid()
	{
		System.out.println("custon exception");
	}
}
class exceed extends Exception
{
	exceed()
	{
		System.out.println("custon exception");
	}
}

class resstat
{

	int resref[][]=new int[3][3];
	int free=0,ocp=0,inacc=0;

	resstat()
	{
	}
	resstat(String s1[])
	{
		int n=0;
		for(int i=0;i<3;i++)
		{
			for(int j=0;j<3;j++)
			{
				resref[i][j]=Integer.parseInt(s1[n]);
				n++;
			}
		}
	}
	void prostatcnt ()
	{
		for(int i=0;i<3;i++)
		{
			for(int j=0;j<3;j++)
			{
				if(resref[i][j]==0)
				{
					free++;
				}
				else if(resref[i][j]==1)
				{
					ocp++;
				}
				else if(resref[i][j]==2)
				{
					inacc++;
				}
				else
				{
					try
					{
						throw new invalid();
					}
					catch(invalid e)
					{
						System.out.println("invalis status value");
					}
				}
			}
		}
		try
		{
			if(ocp>free)
			{
				throw new exceed();
			}
		}
```

```
				catch(exceed e)
				{
						System.out.println("total no of occupied resources exceeds total no of
free resources");
						free=free+inacc;
						inacc=0;
				}
				System.out.println("free resourse ="+free);
				System.out.println("occupied resourse= "+ocp);
				System.out.println("inaccecible resourse= "+inacc);
		}
}
class test
{
		public static void main(String s[])
		{
				resstat r1=new resstat(s);
				r1.prostatcnt();
		}
}
```

OUTPUT:

```
V:\j>java test 1 0 0 1 1 2 2 0 0
free resourse =4
occupied resourse= 3
inaccecible resourse= 2


V:\j>java test 1 0 0 1 1 2 2 0 1
custon exception
total no of occupied resources exceeds total no of free resources
free resourse =5
occupied resourse= 4
inaccecible resourse= 0


V:\j>java test 1 0 0 1 1 2 2 0 4
custon exception
invalis status value
free resourse =3
occupied resourse= 3
inaccecible resourse= 2
```