**Practical 1:** Write a method for computing first n terms of Fibonacci sequence. Define method main taking value of n as command line argument and calling the method.

**Solution:**

class Fibonacci

{

public static void main(String[] args)

{

int a=0,b=1,c;

int n=args[0];

if(n==0)

System.out.print(a+" ");

if(n>0)

{

System.out.print(a+" ");

System.out.print(b+" ");

}

for(int i=2;i<=n;i++)

{

c=a+b;

System.out.print(c+" ");

a=b;

b=c;

}

}

}

**Practical 2:** It is required to compute SPI (semester performance index) of n students of a class for their registered subjects in a semester. Assume that all students register for 6 subjects and each subject carry 5 credits. Also, follow GTU convention and method for computation of SPI. Declare a class called student having following data members: id\_no, grades\_obtained and spi. Define constructor, display and calculate\_spi methods. Define main to process data of n students.

**Solution:**

import java.util.\*;

import java.io.\*;

class Students

{

long eroll,sem;

String name;

String Grades[]=new String[6];

int point=0;

int count=0;

Scanner sc=new Scanner(System.in);

public void input()

{

System.out.println("Enter Your Roll No.= ");

eroll=Long.parseLong(sc.nextLine());

System.out.println("Enter your Name= ");

name=sc.nextLine();

System.out.println("Enter Your Semester= ");

sem=Integer.parseInt(sc.nextLine());

System.out.println("Enter Your Grades:{AA,AB,BB,BC,CC,CD,DD}");

for(int i=0;i<6;i++)

{

Grades[i]=sc.nextLine();

switch(Grades[i])

{

case "AA": point=point+10;break;

case "AB": point=point+9;break;

case "BB": point=point+8;break;

case "BC": point=point+7;break;

case "CC": point=point+6;break;

case "CD": point=point+5;break;

case "DD": point=point+4;break;

case "FF": point=point+0;count ++;break;

default:System.out.println("Invalid Grade Input");break;

}

}

}

public void show()

{

System.out.println("Name= "+name+"\tRoll No.= "+eroll+"\tSemester= "+sem+"\n Grades= ");

for(int i=0;i<6;i++)

{

System.out.println(Grades[i]);

}

System.out.println();

for(int i=0;i<50;i++)

{

System.out.print("==");

}

System.out.println();

System.out.println(" SPI= "+((point)\*6)/36);

if(count>0)

{

System.out.println("You are Failed in "+count+" Subject");

}

}

}

public class Spi

{

public static void main(String args[])throws IOException

{

Students s=new Students();

s.input();

s.show();

}

}

**Practical 3:** Create a class called Student. Write a student manager program to manipulate the student information from files by using FileInputStream and FileOutputStream

**Solution:**

**import java.io.\*;**

**import java.util.Scanner;**

**class Student**

**{**

**static Scanner scan=new Scanner(System.in);**

**static void addInfo(FileOutputStream fout) throws IOException**

**{**

**System.out.println("provide information:");**

**System.out.print("Enroll: ");**

**String enroll=scan.next();**

**System.out.print("FirstName: ");**

**String name=scan.next();**

**System.out.print("Sem: ");**

**String sem=scan.next();**

**byte b[]=enroll.getBytes();**

**fout.write(b);**

**fout.write('\_');**

**b=name.getBytes();**

**fout.write(b);**

**fout.write('\_');**

**b=sem.getBytes();**

**fout.write(b);**

**fout.write('\n');**

**}**

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

**{**

**System.out.println("Select from below operations:");**

**System.out.println("1: Add Information");**

**System.out.println("2: View Student Information");**

**int choice=scan.nextInt();**

**switch(choice)**

**{**

**case 1:**

**System.out.print("provide the path of file:(with .doc)");**

**String path=scan.next();**

**FileOutputStream fout=new FileOutputStream(path,true);**

**String s;**

**do{**

**addInfo(fout);**

**System.out.println("\"more\" to add entry || \"stop\" to exit");**

**s=scan.next();**

**}while(s.equals("more"));**

**break;**

**case 2:**

**System.out.print("Provide path of file: (with .doc)");**

**String p=scan.next();**

**File f=new File(p);**

**FileInputStream fin=new FileInputStream(f);**

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

**{**

**System.out.print((char)fin.read());**

**}**

**break;**

**default:**

**System.out.println("Invalid choice");**

**}**

**}**

**}**

**Practical 4:** Refine the student manager program to manipulate the student information from files by using the BufferedReader and BufferedWriter

**Solution:**

**Practical 5:** Implement methods of String class and String buffer class .

**Solution:**

**String class:**

class first

{

public static void main(String args[])

{char ch1[]={'h','e','d','l','o'};

String s=new String(ch1);

System.out.println(s.length());

System.out.println("print string");

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

{

System.out.print(ch1[i]);

}

System.out.println("\nprint reverse string");

for(int i=s.length()-1; i>=0;i--)

{

System.out.print(ch1[i]);

}

//System.out.println("\n"+s+"frnds");

String s1=new String("HELLO");

System.out.println(s.concat(s1));

System.out.println(s.compareTo(s1));

System.out.println(s.equals(s1));

System.out.println(s.equalsIgnoreCase(s1));

System.out.println(s.charAt(2));

System.out.println(s1.indexof('L'));

}

}

**String Buffer:**

public class Test

{

public static void main(String args[])

{

StringBuffer sBuffer = new StringBuffer("test");

sBuffer.append("String Buffer");

System.out.println(sBuffer);

}

}

**Practical 6:** The abstract vegetable class has three subclasses named Potato, Brinjal and Tomato. Write a java prog. That demonstrates how to establish this class hierarchy. Declare one instance variable of type String that indicates the color of a vegetable. Crete and display instances of these objects. Override the toString() method of object to return a string with the name of vegetable and its color

**Solution:**

import java.lang.\*;

abstract class Vegetables

{

String colorVeg;

abstract public String toString();

}

class Potato extends Vegetables

{

public String toString()

{

colorVeg = "Yellow";

return colorVeg;

}

}

class Brinjal extends Vegetable

{

public String toString()

{

colorVeg="Purple";

return colorVeg;

}

}

class Tomato extends Vegetable

{

public String toString()

{

colorVeg="Red";

return colorVeg;

}

}

class abstractClass

{

public static void main(String args[])

{

Potato p = new Potato();

Brinjal b = new Brinjal();

Tomato t = new Tomato();

Vegetable vegref;

vegref = p;

System.out.println("Color of potato:"+vegref.toString());

vegref = b;

System.out.println("Color of potato:"+vegref.toString());

vegref = t;

System.out.println("Color of potato:"+vegref.toString());

}

}

**Practical 7:** Write a program to demonstrate use of multiple inheritance .

**Solution:**

/\*\* Java Program for Student Result Calculation\*/

interface Exam

{

void percent\_cal();

}

class Student

{

String name;

int roll\_no,mark1,mark2;

Student(String n, int r, int m1, int m2)

{

name=n;

roll\_no=r;

mark1=m1;

mark2=m2;

}

void display()

{

System.out.println ("Name of Student: "+name);

System.out.println ("Roll No. of Student: "+roll\_no);

System.out.println ("Marks of Subject 1: "+mark1);

System.out.println ("Marks of Subject 2: "+mark2);

}

}

class Result extends Student implements Exam

{

Result(String n, int r, int m1, int m2)

{

super(n,r,m1,m2);

}

public void percent\_cal()

{

int total=(mark1+mark2);

float percent=total\*100/200;

System.out.println ("Percentage: "+percent+"%");

}

void display()

{

super.display();

}

}

class MultipleInheritance

{

public static void main(String args[])

{

Result R = new Result("Ra.one",12,93,84);

R.display();

R.percent\_cal();

}

}

**Practical 8:** 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.

**Solution:**

/\*\*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 ExceptionHandling

{

public static void main(String args[])

{

int len,i;

int sum=0,t,num;

len=args.length;

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

{

num=Integer.parseInt(args[i]);

try

{

if(num<0)

throw new Exception("Number "+num+" is Negative");

else if(num>1000 && num<2000)

throw new Exception("Number "+num+" is between 1000 and 2000");

else if(num>7000)

throw new Exception("Number "+num+" is greater than 7000");

else

{

int number=num,digit=0,sumOfDigits=0;

while (number != 0)

{

digit = number % 10;

sumOfDigits = sumOfDigits + digit;

number = number / 10;

}

if(sumOfDigits%10==0)

throw new Exception("Number "+num+" is Evenly Divisible by 10");

}

}

catch(Exception e)

{

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

continue;

}

sum=sum+num;

}

System.out.println("Sum of all the number is : "+sum);

}

}

**Practical 9:** Write a complete multi-threaded program to meet following requirements for producerconsumer threads: - Three threads – one producer and two consumers to be instantiated in the method main. - At a time, the producer produces one integer information along with consumer\_id to represent id of a consumer that will consume produced information. - Information and consumer\_id are stored in a shared buffer. - The information produced is to be consumed by appropriate consumer only, as specified by the producer. - The producer thread produces total 6 information.

**Solution:**