JAVA LAB PROGRAM

Submitted by,

Tamilvanan B.

2018503566.

MO Batch.

09-15-2020.

1. Write a Java program that computes grades for the students in a university based on mode of study and programme. The program should implement the concept of inheritance with the following related classes: a class “Student”, which consists of two subclasses “PartTime” and “FullTime” mode of study. These subclasses further subdivide into still smaller subclasses “Under\_graduate” and “Post\_graduate” programme. Choose appropriate class members for computing the grades. Show variations in grade computations. The key parts of the program are:

· A private static variable ,protected variable

· Constructors among the inherited classes in the hierarchy and the usage of “super” keyword in constructors.

· Overridden method for computing grades. Usage of “super” keyword in method.

Program:

import java.util.Scanner;

class Student{

String name;

int regno;

int[] marks;

protected int grade;

private static int students=0;

Student(String name,int regno){

this.name=name;

this.regno=regno;

students++;

}

void details(int subs){

System.out.println("Name: "+this.name+"\nReg.No: "+this.regno+"\nMarks: ");

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

System.out.println("Subject "+(i+1)+": "+marks[i]);

System.out.println("Grade: "+grade);

}

static int count(){

return students;

}

}

class PartTime extends Student{

private static int students=0;

PartTime(String name,int regno){

super(name,regno);

students++;

}

void setGrade(){

System.out.println("Setting Grade...");

}

static int count(){

return students;

}

}

class FullTime extends Student{

private static int students=0;

FullTime(String name,int regno){

super(name,regno);

students++;

}

void setGrade(){

System.out.println("Setting Grade...");

}

static int count(){

return students;

}

}

class UG\_PartTime extends PartTime{

static int students=0;

private static int subjects=5;

UG\_PartTime(String name,int regno,int[] marks){

super(name,regno);

this.marks=new int[5];

for(int i=0;i<5;i++)

this.marks[i]=marks[i];

students++;

}

void setGrade(){

int i,total=0;

for(i=0;i<5;i++)

total+=marks[i];

grade=total/5;

}

void details(){

super.details(subjects);

}

}

class PG\_PartTime extends PartTime{

static int students=0;

private static int subjects=3;

PG\_PartTime(String name,int regno,int[] marks){

super(name,regno);

this.marks=new int[3];

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

this.marks[i]=marks[i];

students++;

}

void setGrade(){

int i,total=0;

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

total+=marks[i];

grade=total/3;

}

void details(){

super.details(subjects);

}

}

class UG\_FullTime extends FullTime{

static int students=0;

private static int subjects=5;

UG\_FullTime(String name,int regno,int[] marks){

super(name,regno);

this.marks=new int[5];

for(int i=0;i<5;i++)

this.marks[i]=marks[i];

students++;

}

void setGrade(){

int i,total=0;

for(i=0;i<5;i++)

total+=marks[i];

grade=total/5;

}

void details(){

super.details(subjects);

}

}

class PG\_FullTime extends FullTime{

static int students=0;

private static int subjects=3;

PG\_FullTime(String name,int regno,int[] marks){

super(name,regno);

this.marks=new int[3];

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

this.marks[i]=marks[i];

students++;

}

void setGrade(){

int i,total=0;

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

total+=marks[i];

grade=total/5;

}

void details(){

super.details(subjects);

}

}

public class grades {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int i,j,op=1,regno,mode,programme;

String name;

int[] marks=new int[5];

UG\_PartTime[] ugp=new UG\_PartTime[20];

PG\_PartTime[] pgp=new PG\_PartTime[20];

UG\_FullTime[] ugf=new UG\_FullTime[20];

PG\_FullTime[] pgf=new PG\_FullTime[20];

while(op!=0){

System.out.print("Student Name: ");

name=sc.next();

System.out.print("Reg.no: ");

regno=sc.nextInt();

System.out.println("Mode of study:\n1 - Part Time\n2 - Full Time");

mode=sc.nextInt();

System.out.println("Programme:\n1 - UG\n2 - PG");

programme=sc.nextInt();

System.out.println("Enter marks of each subject: ");

if(programme==1) j=5;

else j=3;

for(i=0;i<j;i++){

System.out.print("Subject "+(i+1)+": ");

marks[i]=sc.nextInt();

}

if(programme==1&&mode==1){

ugp[UG\_PartTime.students]=new UG\_PartTime(name, regno, marks);

ugp[UG\_PartTime.students-1].setGrade();

}

else if(programme==2&&mode==1){

pgp[PG\_PartTime.students]=new PG\_PartTime(name, regno, marks);

pgp[PG\_PartTime.students-1].setGrade();

}

else if(programme==1&&mode==2){

ugf[UG\_FullTime.students]=new UG\_FullTime(name, regno, marks);

ugf[UG\_FullTime.students-1].setGrade();

}

else if(programme==2&&mode==2){

pgf[PG\_FullTime.students]=new PG\_FullTime(name, regno, marks);

pgf[PG\_FullTime.students-1].setGrade();

}

System.out.println("0 Exit\nELSE Proceed");

op=sc.nextInt();

}

sc.close();

System.out.println("\nTotal no.of students: "+Student.count());

System.out.println("No.of Students in Part time: "+PartTime.count());

System.out.println("No.of Students in Full time: "+FullTime.count());

System.out.println("Students in UG Part time: "+UG\_PartTime.students);

for(i=0;i<UG\_PartTime.students;i++){

System.out.println("\nStudent "+(i+1)+": ");

ugp[i].details();

}

System.out.println("\nStudents in PG Part time: "+PG\_PartTime.students);

for(i=0;i<PG\_PartTime.students;i++){

System.out.println("\nStudent "+(i+1)+": ");

pgp[i].details();

}

System.out.println("\nStudents in UG Full time: "+UG\_FullTime.students);

for(i=0;i<UG\_FullTime.students;i++){

System.out.println("\nStudent "+(i+1)+": ");

ugf[i].details();

}

System.out.println("\nStudents in PG Full time: "+PG\_FullTime.students);

for(i=0;i<PG\_FullTime.students;i++){

System.out.println("\nStudent "+(i+1)+": ");

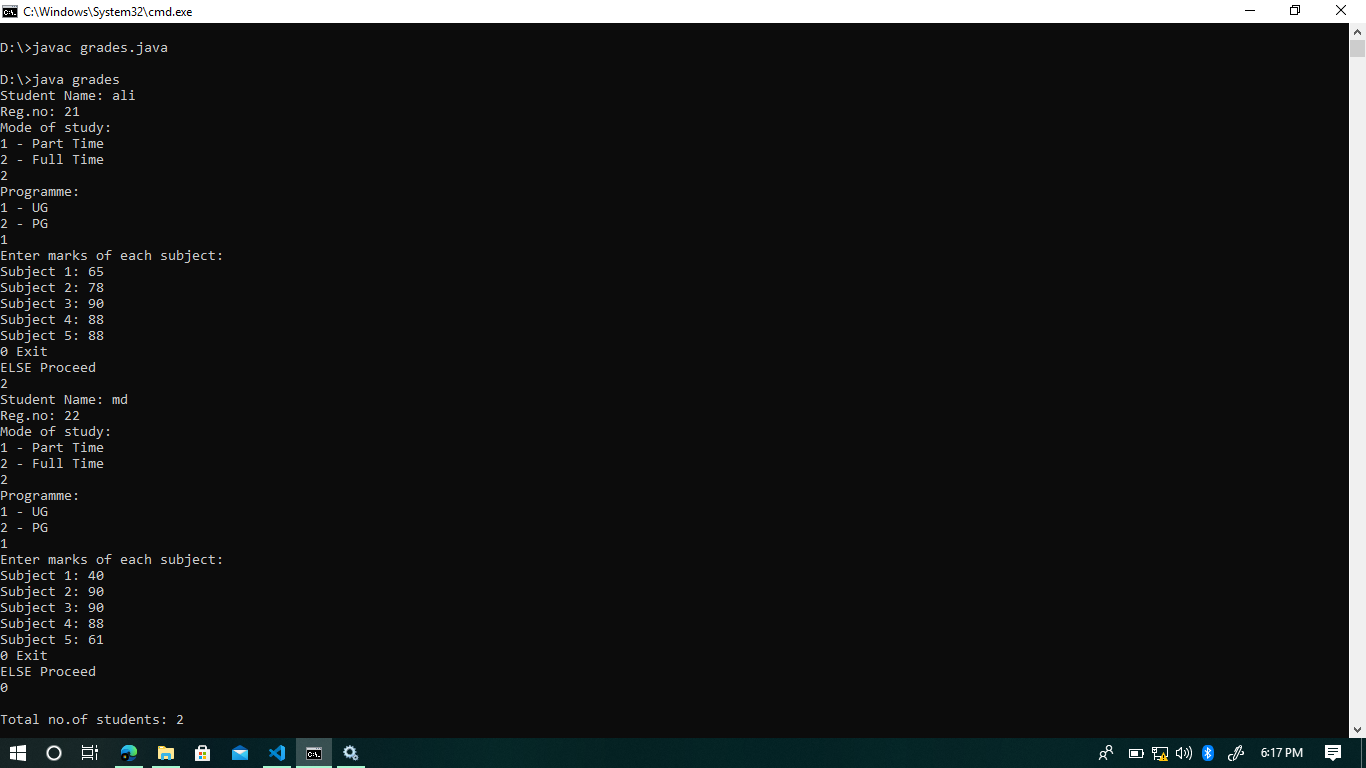
pgf[i].details();

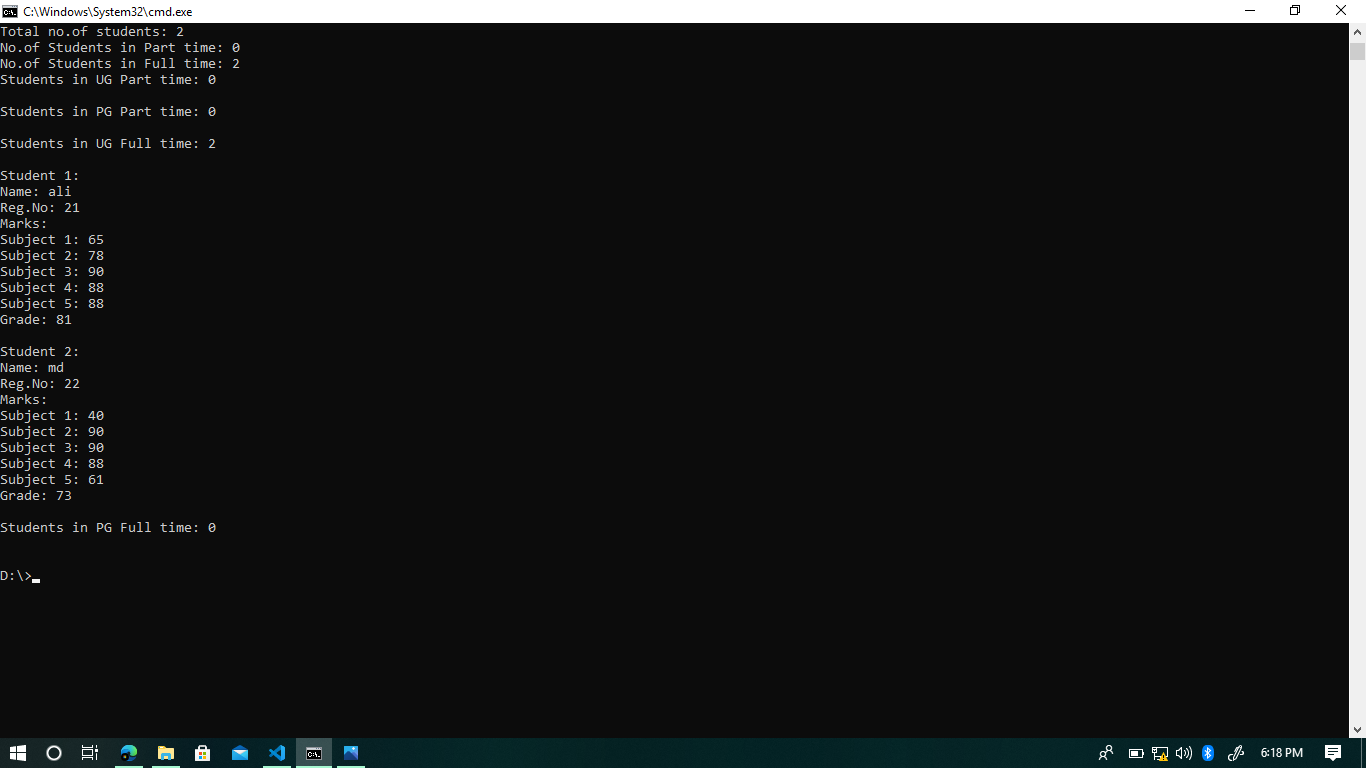
}

System.out.println();

}

}

Output:



2. Write a java program that keeps track of the project details and computes the workload percentage for full-time and contract employees in an organization. Classes “contract” and “Full-time” inherits properties from class “Employee”. The Employee class includes fields that describes the employee and project details. The Employee class holds a method named workloadBilling() that computes the workload percentage of an employee based on the number of hours they work in a project and total number of hours they spent in the organization. The class full-time and contract class should override the workloadBilling() method to add their extra hours. Include a static method workCheck() in both the subclasses to display the project details along with number of hours they work in it. Include a method projectTrack() in Employee class that displays the details of project, number of employees working on it and their details.

Note: Both full-time and contract employee can work in a same project.

Program:

import java.util.\*;

class EMployee{

int id,hours\_on\_project,hours\_on\_org;

EMployee(int a,int b,int c){

id = a;

hours\_on\_project = b;

hours\_on\_org = c;

}

void projecttrack(){

System.out.println("id : " + id);

System.out.println("hours\_on\_project : " + hours\_on\_project);

System.out.println("hours\_on\_org : " + hours\_on\_org);

}

void WorkloadBilling(){

System.out.println("workload Percentage ");

System.out.println(((double)hours\_on\_project/hours\_on\_org) \* 100);

}

void add(int val){

hours\_on\_org += val;

hours\_on\_project += val;

}

}

class contract extends EMployee{

contract(int a,int b,int c){

super(a,b,c);

}

void WorkloadBilling(){

Scanner input = new Scanner(System.in);

System.out.println("enter the extra hours");

int val = input.nextInt();

super.add(val);

input.close();

}

void Workcheck(){

System.out.println("no. of hours in org : " + super.hours\_on\_org);

System.out.println("no. of hours in pro : " + super.hours\_on\_project);

}

void cal(){

super.WorkloadBilling();

}

}

class Full\_time extends EMployee{

Full\_time(int a,int b,int c){

super(a,b,c);

}

void WorkloadBilling(){

Scanner input = new Scanner(System.in);

System.out.println("enter the extra hours");

int val = input.nextInt();

super.add(val);

input.close();

}

void Workcheck(){

System.out.println("no. of hours in org : " + super.hours\_on\_org);

System.out.println("no. of hours in pro : " + super.hours\_on\_project);

}

void cal(){

super.WorkloadBilling();

}

}

public class emp {

public static void main(String args[]){

Scanner input = new Scanner(System.in);

contract a = new contract(11,78,152);

System.out.println("contract class is created . . . ");

a.cal();

a.WorkloadBilling();

a.cal();

a.projecttrack();

input.close();

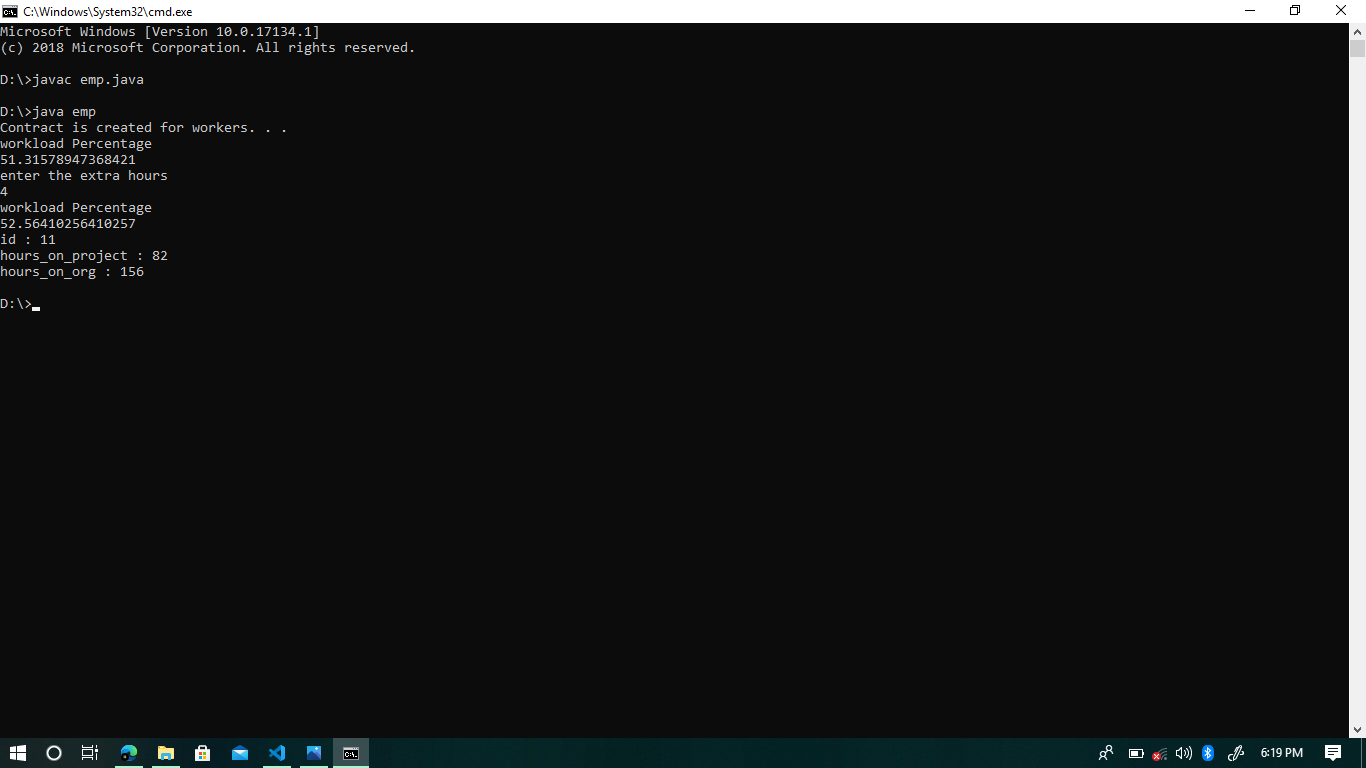
}

}

Output:

Challenge question

1. Create a Java program that finds the number of four digit numbers greater than 4000 that can be formed using the digits 1, 2, 3, 4, 5, 6 and 7 with no digit repeating.

Program:

public class challenge{

public static void main(String[] args) {

int arr[]={1,2,3,4,5,6,7};

int n=7;

int i=1;

int fact=1;

for(i=n-1;i>=1;i--){

fact=fact\*i;

}

int fact2=1;

for(i=3;i>=1;i--){

fact2=fact2\*i;

}

System.out.println("The maximum number of 4 digit numbers that can be created are " + 4\*(fact/fact2));

}

}

Output:

