**CS1083**

**Assignment #1**

**Daniyal Khan**

**3765942**

**Source Code for Gradable.java:**

*public* *interface* Gradable {

*public* double *calculateGPA*();

*public* String *listCourses*();

}

**Source Code for Student.java:**

*public* *abstract* *class* Student *implements* Gradable {

*private* *static* int ID = 1000;

*private* *final* int STUDENT\_ID;

*private* String name;

*private* CourseMatrix courseMatrix;

*public* *Student*(String name, CourseMatrix courseMatrix) {

STUDENT\_ID = ID;

ID++;

this.*name* = name;

this.*courseMatrix* = courseMatrix;

}

*public* String *getName*() {

*return* name;

}

*public* int *getID*() {

*return* STUDENT\_ID;

}

*public* CourseMatrix *getCourseMatrix*() {

*return* courseMatrix;

}

*public* String *listCourses*() {

CourseMatrix courseMatrix = *getCourseMatrix*();

String[][] courseCode = courseMatrix.*getCourseCodes*();

double[][] gpa = courseMatrix.*getGPA*();

String courseList = "";

*for* (int term = 0; term < courseCode.*length*; term++) {

courseList += "Term: " + (term+1) + "\n";

*for* (int course = 0; course < courseCode[term].*length*; course++) {

courseList += gpa[term][course] + " " + courseCode[term][course] + "\n";

}

}

*return* courseList;

}

}

**Source Code for UndergradStudent.java:**

*public* *class* UndergradStudent *extends* Student {

*private* String degreeProgram;

*public* *UndergradStudent*(String name, CourseMatrix courseMatrix, String degreeProgram) {

super(name, courseMatrix);

this.*degreeProgram* = degreeProgram;

}

*public* double *calculateGPA*() {

String[][] courseCodes = this.*getCourseMatrix*().*getCourseCodes*();

double[][] gpas = this.*getCourseMatrix*().*getGPA*();

int totalCredits = 0;

double totalGradePoints = 0.0;

*for* (int term = 0; term < courseCodes.*length*; term++) {

*for* (int course = 0; course < courseCodes[term].*length*; course++) {

String courseCode = courseCodes[term][course];

double gpa = gpas[term][course];

*if* (!courseCode.*equals*("") && gpa != -1) {

*if*(courseCode.*startsWith*(degreeProgram)) {

totalCredits += 4;

totalGradePoints += (4\*gpa);

} *else* {

totalCredits += 3;

totalGradePoints += (3\*gpa);

}

}

}

}

*if* (totalCredits == 0) {

*return* 0;

}

*return* totalGradePoints/totalCredits;

}

*public* String *toString*() {

*return* "Name: " + this.*getName*() + "\n" + "Degree: " + this.*degreeProgram* + "\n" + "Overall GPA: " + Math.*round*(this.*calculateGPA*()\*100.0)/100.0 + "\n";

}

}

**Source Code for CourseMatrix.java:**

*public* *class* CourseMatrix {

*private* String[][] courseCodes;

*private* double[][] gpa;

*public* *CourseMatrix*(String[][] courseCodes, double[][] gpa){

this.*courseCodes* = courseCodes;

this.*gpa* = gpa;

}

*public* String[][] *getCourseCodes*() {

*return* courseCodes;

}

*public* double[][] *getGPA*() {

*return* gpa;

}

}

**Source Code for Driver.java:**

import *java*.*util*.*ArrayList*;

*public* *class* Driver {

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

String[][] courseCodes1 = {

{"CS1303", "CS1203", "MATH1003", "TME2001", "CS1073"},

{"CS1083", "CS3113", "MATH1013", "CS1103", "ENGL1103"}

};

double[][] gpa1 = {

{3.8, 4.1, 4, 3.4, 3.6},

{2.7, 3.4, 2.3, 4.1, 3.9}

};

String[][] courseCodes2 = {

{"ME1001", "MATH1003", "CS1023", "ME1033", "MATH1503"},

{"ME1365", "MATH1013", "ENGL1103", "ME1830", "MATH3113"},

{"PHYS2001", "ME2033", "ME1013", "CHEM1103", "CS3013"},

{"HIST2103", "ART1203", "MATH2023", "CS4003", "ECON1003"}

};

double[][] gpa2 = {

{3.0, 2.9, 3.6, 3.9, 2.5},

{4.0, 3.9, 3.0, 2.6, 4.1},

{3.8, 4.1, 4, 3.4, -1},

{2.7, 3.4, 2.3, 4.1, 3.9}

};

String[][] courseCodes3 = {

{"EE1303", "CS1023", "MATH1003", "MATH1503", "ENGL1103"},

{"EE2023", "EE1230", "MATH1013", "BIO1103", "CHEM1103"}

};

double[][] gpa3 = {

{3.9, 2.8, 4.2, 3.5, 3.7},

{2.6, 3.2, 3.9, -1, 2.9}

};

String[][] courseCodes4 = {

{"FOR2013", "FOR3013", "GGE2023", "GGE4103", "MATH2103"},

{"GGE1013", "GGE3043", "FOR2203"}

};

double[][] gpa4 = {

{3.5, 4.0, 3.7, 3.3, 3.9},

{2.8, 3.1, 4.2}

};

CourseMatrix courseMatrix1 = *new* *CourseMatrix*(courseCodes1, gpa1);

CourseMatrix courseMatrix2 = *new* *CourseMatrix*(courseCodes2, gpa2);

CourseMatrix courseMatrix3 = *new* *CourseMatrix*(courseCodes3, gpa3);

CourseMatrix courseMatrix4 = *new* *CourseMatrix*(courseCodes4, gpa4);

UndergradStudent undergradStudent1 = *new* *UndergradStudent*("Omar", courseMatrix1, "CS");

UndergradStudent undergradStudent2 = *new* *UndergradStudent*("Said", courseMatrix2, "ME");

UndergradStudent undergradStudent3 = *new* *UndergradStudent*("Gab", courseMatrix3, "MATH");

UndergradStudent undergradStudent4 = *new* *UndergradStudent*("Emily", courseMatrix4, "FOR");

ArrayList<UndergradStudent> arraylist = *new* ArrayList<>();

arraylist.*add*(undergradStudent1);

arraylist.*add*(undergradStudent2);

arraylist.*add*(undergradStudent3);

arraylist.*add*(undergradStudent4);

*for* (UndergradStudent undergrad *:* arraylist) {

System.*out*.*println*(undergrad);

}

System.*out*.*println*("-----------------------------");

System.*out*.*println*("List Courses:");

System.*out*.*println*(undergradStudent1.*getName*() + "\n\n" + undergradStudent1.*listCourses*());

System.*out*.*println*(undergradStudent2.*getName*() + "\n\n" + undergradStudent2.*listCourses*());

System.*out*.*println*(undergradStudent3.*getName*() + "\n\n" + undergradStudent3.*listCourses*());

System.*out*.*println*(undergradStudent4.*getName*() + "\n\n" + undergradStudent4.*listCourses*());

}

}

**UML Diagram:  
  
A paper with writing on it

Description automatically generated**

**Output:**

A computer screen shot of a black background

Description automatically generated

The program lists the name, degree and overall gpa of 4 undergraduate students with different courses and gpa.

Also prints out the list of courses they have taken along with GPA in each course using listCourses() method. -1 means that they did not complete the course.

It works by adding all the undergraduate students to an arraylist and then using a for-each loop to print them.

A black screen with white text

Description automatically generated  
A screenshot of a computer

Description automatically generated