**COMP2130 Assignment 1**

**Due: Sunday, October 17th, 2021**

**11:00 P.M.**

**This is an individual assignment**

This Assignment is designed to take you through creating classes, aggregation, and manipulating arrays of objects.

**Scenario:** A University likes to have a simple system to keep track of all the students (graduate and undergrads). You have to create a menu-driven program for the user to use the system through the console. The following classes are needed for this object-oriented database.

1. Create a class called Student that has the following stored properties:

**Student**

* StudentID : Integer
* stdFirstName: String
* stdLastName: String
* stdMarks : Double []
* stdAddress: Address

\*\* Class Student should have set/get properties, constructor and have following methods:

Average() - that returns the average grade for students

toString() method that returns the above information as a String

1. Create a class called Address which can be **aggregated** into the class student

Address

* streetInfo: String
* city: String
* postalCode: String
* province: String
* country: String

\*\* Class Address should have set/get properties, constructor and following method:

toString() method that returns the above information as a String

1. Create a class called UndergraduateStudent that **inherits** from Student and has the following members:

**Undergrad Student**

* subject: String
* yearOfEntry :Integer

\*\*\*Class UndergraduateStudent should have set/get properties, constructor and following method:

Graduate() – Boolean that returns true if the Student is eligible to graduate when the average of their marks is greater than 50.

toString() method that returns the above information as a String

1. Create a class called GraduateStudent that **inherits** from Student and has the following members:

**Graduate Student**

* subject : String
* yearOfEntry :Integer
* thesisTopic: String

Class GraduateStudent should have set/get properties, constructor and following method:

Graduate() – Boolean that returns true if the Student is eligible to graduate when the average of their marks is greater than 70.

toString() method that returns the above information as a String

**Summary of Operations**

**System Menu:**

1. Add undergraduate student
2. Add graduate student
3. View all the students
4. View only eligible students for graduation
5. exit

Overview:

* You may use array or ArrayList to store **all your students**(graduate and undergrad) into **one** array of objects.
* If you use an Array, you may assume the user does not enter more than ten students in total into the system

1 –**Add undergraduate Student:** this menu should accept all the necessary parameters for undergraduate students and create an instance from undergraduate class and store it into students array.

2 -**Add graduate student:** this menu should accept all the necessary parameters for graduate students. It should create an instance from the graduate class and store it in the students' array.

3- **View all the students:** view all the relevant information of students (graduate and undergraduate) from students array

4- **View only eligible students for graduation:** view all the relevant information (graduate and undergraduate) from the students' array only if they are eligible to graduate.

5 – **Exit:** exit the running menu (program)

**Submission Requirements:**

* Submission: On Blackboard.
* Required files:
  + You may upload all the java files, including main or/and zip the project and upload.
  + If you use online compilers, you may copy-paste your code into this document.

CAREFUL NOTE:

- Please safeguard your code work.

- If two or more assignments are the same (or very much alike) they will all get 0 marks, so be cautious not to share your application with others.

- Note: **Mobi-Help** members are **NOT** supposed to do or help you with your assignment code.

MAIN

package universitysystem\_asgmt1;

import java.util.Scanner;

/\*\*

\*

\* @author Stevanella Marco

\* id: 101307949

\* ASGMT 1 - COMP 2130

\* Professor Hesam Akbari

\* Notes: I have decided to use an array of 10 as I would like to first deepen my knowledge on ArrayList before using them. Thank you for spending time on this and marking it.

\*/

public class UniversitySystem\_asgmt1 {

static Student [] students = new Student [10];

static Scanner input = new Scanner (System.in);

static int indexStudentsCounter = 0;

public static void main(String[] args) {

WelcomeMessage();

boolean isUserActive = true;

while(isUserActive){

Menu();

int selection = input.nextInt();

switch (selection){

case 1:

AddUndergradStudent();

break;

case 2:

AddGraduateStudent();

break;

case 3:

ViewStudents();

break;

case 4:

ViewEligibleStudents();

break;

case 5:

System.out.println("Exit");

isUserActive = false;

break;

default:

System.out.println("We think you entered a wrong number, digit a valid number.");

}

}

}

// Control Flow Methods

public static void WelcomeMessage(){

System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\n"

+ "Hi, Welcome to the University Control System for STAFF.\r\n"

+ "This System will allow you to create new students, view them and see who is elegible for graduation.\r\n"

+"-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\n");

}

public static void Menu (){

System.out.println("Select one of the following options:\n\r"

+ "1) Add Undergraduate Student\n\r"

+ "2) Add Graduate Student\n\r"

+ "3) View all the Students\n\r"

+ "4) View only elegible Student\n\r"

+ "5) exit\n\r");

}

public static void AddGraduateStudent(){

char selection;

System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\n"

+ "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=- ADD GRADUATE STUDENT -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\n"

+ "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\n");

boolean userWantsToCreate = true;

while(userWantsToCreate){

System.out.println("\r\nWould you like to create a new graduate student? Yes (Y) No (N)");

selection = input.next().charAt(0);

if(selection == 'Y' || selection == 'y'){

System.out.print("ID: ");

int studentID = input.nextInt();

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

String studentFirstName = input.next();

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

String studentLastName = input.next();

System.out.print("Num of Marks to record: ");

int numMarksToRecord = input.nextInt();

double [] marks = new double [numMarksToRecord];

for ( int j = 0 ; j < marks.length ; j++){

System.out.print("Mark " + (j+1) + ": "); // we should display to the user the right number

marks[j] = input.nextDouble();

}

System.out.print("Street: ");

String streetInfo = input.next();

System.out.print("City: ");

String cityInfo = input.next();

System.out.print("Postal Code: ");

String postalCode = input.next();

System.out.print("Province: ");

String province = input.next();

System.out.print("Country: ");

String country = input.next();

System.out.print("Subject: ");

String subject = input.next();

System.out.print("Entry year: ");

int entry = input.nextInt();

System.out.print("Thesis: ");

String thesis = input.next();

GraduateStudent graduateStud = new GraduateStudent(studentID,studentFirstName,studentLastName,marks,new Address (streetInfo,cityInfo,postalCode,province,country),subject,entry,thesis);

students [indexStudentsCounter] = graduateStud;

indexStudentsCounter++;

System.out.print(graduateStud);

}

else if (selection == 'N' || selection == 'n'){

userWantsToCreate = false;

System.out.print("\r\n-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\nYou are being redirected to the main menu...\r\n-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\n");

}

else{

System.out.print("You must have entered another key, ");

userWantsToCreate = true;

}

}

}

public static void ViewStudents(){

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

if(students[i] == null){

System.out.print("\r\nEmpty Slot at " + i + " for eventual student\r\n");

}

else{

System.out.print("\r\n=-=-=-=-=- STUDENT -=-=-=-=-=\r\n" );

System.out.print(students[i].toString() + "\r\n");

System.out.print("\r\n-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\n" );

}

}

}

public static void ViewEligibleStudents(){

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

if(students[i] instanceof UndergraduateStudent && students[i].Average() >= 50){

System.out.print(students[i].toString() + "\r\n");

}

else if (students[i] instanceof GraduateStudent && students[i].Average() >= 70)

System.out.print(students[i].toString() + "\r\n");

}

}

public static void AddUndergradStudent (){

char selection;

System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\n"

+ "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=- ADD UNDERGRAD STUDENT -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\n"

+ "-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\n");

boolean userWantsToCreate = true;

while(userWantsToCreate){

System.out.println("\r\nWould you like to create a new undergrad student? Yes (Y) No (N)");

selection = input.next().charAt(0);

if(selection == 'Y' || selection == 'y'){

System.out.print("ID: ");

int studentID = input.nextInt();

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

String studentFirstName = input.next();

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

String studentLastName = input.next();

System.out.print("Num of marks to record: ");

int numMarksToRecord = input.nextInt();

double [] marks = new double [numMarksToRecord];

for ( int j = 0 ; j < marks.length ; j++){

System.out.print("Mark " + (j+1) + ": ");

marks[j] = input.nextDouble();

}

System.out.print("Street: ");

String streetInfo = input.next();

System.out.print("City: ");

String cityInfo = input.next();

System.out.print("Postal Code: ");

String postalCode = input.next();

System.out.print("Province: ");

String province = input.next();

System.out.print("Country: ");

String country = input.next();

System.out.print("Subject: ");

String subject = input.next();

System.out.print("Entry Year: ");

int entry = input.nextInt();

UndergraduateStudent undergradStud = new UndergraduateStudent(studentID,studentFirstName,studentLastName,marks, new Address(streetInfo,cityInfo,postalCode,province,country),subject,entry);

students[indexStudentsCounter] = undergradStud;

indexStudentsCounter++;

System.out.print(undergradStud);

}

else if (selection == 'N' || selection == 'n'){

userWantsToCreate = false;

System.out.print("\r\n-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\nYou are being redirected to the main menu...\r\n-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\r\n");

}

else{

System.out.print("You must have entered another key, ");

userWantsToCreate = true;

}

}

}

}

**Marking Scheme:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trait** | **Excellent (85-100)** | **Good (70-85)** | **Satisfactory(50-70)** | **Unsatisfactory (< 50)** |
| **Delivery**  **(5 marks)** | * Submitted on time and in the correct format. * Completed 90 - 100 percent of the program requirements | * Submitted on time and in the correct format. * Completed 75 -90 percent of the program requirements | * Submitted on time and in the correct format. * Completed 70 -80 percent of the program requirements | * Submitted late or in the wrong format. * Completed less than 70% of the program requirements |
| **Coding Standards and Documentation (10 marks)** | * Includes name, date and assignment number. * Excellent variable names used (no global variables, or vague naming). * Useful documentation descriptions. * All functions commented. * Indented to standard. | * Includes name, date and assignment number. * Appropriate variable names used (little use of global variables, or vague naming). * Useful documentation descriptions. * Most functions commented. Indented to standard | * Includes name, date and assignment number. * Appropriate variable names used (a few use of global variables, or vague naming). * Basic documentation descriptions including purpose for functions. * Mostly indented well | * No name, program description included * Poor or misleading variable names used. Little or no indentation. * Regular use of global variables |
| **Specification and Runtime (75 marks)** | * The program meets all of the specifications required and works. * No errors in output. Output is formatted excellently. * All requirements met | * No errors in output. Output is formatted. * All requirements met. * It also meets most of the other specifications. | * No errors in output. * Output has basic formatting or meets core specifications only. | * Does not run due to errors, data read incorrectly. * Little or no requirement met. * Output is poorly formatted or does not follow specifications. |
| **Efficiency**  **(10 marks)** | * Algorithm is easy to understand and efficient. * Can be maintained or modified with minimal changes | * Algorithm is easy to understand and efficient | * Algorithm is easy to understand and but inefficient (excessive use of variables, loops or conditionals) | * Algorithm is hard to understand and very inefficient (excessive use of variables, loops or conditionals) |