College of Engineering, Construction and Living Sciences
Bachelor of Information Technology
ID511001: Programming 2
Level 5, Credits 15

Project 1: Student Management System

Assessment Overview

In this assessment, you will design and develop a student management system **Console Application** using **C#**.

Learning Outcomes

At the successful completion of this course, learners will be able to:

1. Build interactive, event-driven GUI applications using pre-built components.
2. Declare and implement user-defined classes using encapsulation, inheritance and polymorphism.

Assessments

Assessment	Weighting	Due Date	Learning Outcomes
Project 1: Student Management System	35%	20-09-2023 (Wednesday at 4.59 PM)	1 and 2
Project 2: Pong	25%	10-11-2023 (Friday at 04.59 PM)	1 and 2
Theory Examination	30%	15-11-2023 (Wednesday at 12.10 PM)	1 and 2
Classroom Tasks	10%	Multiple Due Dates	1 and 2

Conditions of Assessment

You will complete this assessment during your learner-managed time. However, there will be time during class to discuss the requirements and your progress on this assessment. This assessment will need to be completed by **Wednesday, 20 September 2023** at **4.59 PM**.

Pass Criteria

This assessment is criterion-referenced (CRA) with a cumulative pass mark of **50%** over all assessments in **ID511001: Programming 2**.

Authenticity

All parts of your submitted assessment **must** be completely your work. If you use code snippets from **GitHub**, **StackOverflow** or other online resources, you **must** reference it appropriately using **APA 7th edition**. Provide your references in the **README.md** file in your repository. Failure to do this will result in a mark of **zero** for this assessment.

Policy on Submissions, Extensions, Resubmissions and Resits

The school's process concerning submissions, extensions, resubmissions and resits complies with **Otago Polytechnic — Te Pūkenga** policies. Learners can view policies on the **Otago Polytechnic — Te Pūkenga** website located at https://www.op.ac.nz/about-us/governance-and-management/policies.

Submission

You **must** submit all application files via **GitHub Classroom**. Here is the URL to the repository you will use for your submission – https://classroom.github.com/a/xIHtZr71. Create a **.gitignore** and add the ignored files in this resource - https://raw.githubusercontent.com/github/gitignore/main/VisualStudio.gitignore. The latest application files in the **master** or **main** branch will be used to mark against the **Functionality** criterion. Please test before you submit. Partial marks **will not** be given for incomplete functionality. Late submissions will incur a **10% penalty per day**, rolling over at **5:00 PM**.

Extensions

Familiarise yourself with the assessment due date. Contact the course lecturer before the due date if you need an extension. If you require more than a week's extension, you will need to provide a medical certificate or support letter from your manager.

Resubmissions

Learners may be requested to resubmit an assessment following a rework of part/s of the original assessment. Resubmissions are to be completed within a negotiable short time frame and usually **must** be completed within the timing of the course to which the assessment relates. Resubmissions will be available to learners who have made a genuine attempt at the first assessment opportunity and achieved a **D grade (40-49%)**. The maximum grade awarded for resubmission will be **C-**.

Resits

Resits and reassessments **are not** applicable in **ID511001: Programming 2**.

Instructions

You will need to submit an application and documentation that meet the following requirements:

Functionality - Learning Outcomes 1 and 2 (50%)

- The application **must** open without code or file structure modification in **Visual Studio**.
- The application must contain the following **enums**:

```
public enum Position
{
    LECTURER,
    SENIOR_LECTURER,
    PRINCIPAL_LECTURER,
    ASSOCIATE_PROFESSOR,
    PROFESSOR
}

public enum Salary
{
    LECTURER_SALARY = 85000,
    SENIOR_LECTURER_SALARY = 100000,
    PRINCIPAL_LECTURER_SALARY = 115000,
    ASSOCIATE_PROFESSOR_SALARY = 130000,
    PROFESSOR_SALARY = 145000
}
```

- The application must contain the following **classes**:
  - **Institution**. This **public** class has the following **private** fields: **name** of type **string**, **region** of type **string** and **country** of type **string**.
  - **Department**. This **public** class has the following **private** fields: **institution** of type **Institution** and **name** of type **string**.
  - **Course**. This **public** class has the following **private** fields: **department** of type **Department**, **code** of type **string**, **name** of type **string**, **description** of type **string**, **credits** of type **int** and **fees** of type **double**.
  - **CourseAssessmentMark**. This **public** class has the following **private** fields: **course** of type **Course** and **assessmentMarks** of type **List<int>**. Also, this **public** class has the following **public** methods:
    - **GetAllMarks()** with the return type of **List<int>**. This method returns all assessment marks.
    - **GetAllGrades()** with the return type of **List<string>**. This method returns all assessment grades.
    - **GetHighestMarks()** with the return type of **int**. This method returns the highest passing assessment mark(s). .
    - **GetLowestMarks()** with the return type of **int**. This method returns the lowest passing assessment mark(s).
    - **GetFailMarks()** with the return type of **int**. This method returns the fail assessment mark(s).
    - **GetAverageMarks()** with the return type of **double**. This method returns the average assessment mark.
    - **GetAverageGrade()** with the return type of **string**. This method returns the average assessment grade.

    For more information on how to calculate the highest, lowest and fail marks, refer to the **grade table** in the **Additional Information** section below.
  - **Person**. This **parent** and **public** class has the following **protected** fields: **id** of type **int**, **firstName** of type **string** and **lastName** of type **string**.

- **Learner**. This **child** and **public** class inherits from **Person**. However, has one **private** field: **courseAssessmentMarks** of type **CourseAssessmentMark**.
- **Lecturer**. This **child** and **public** class inherits from **Person**. However, has three **private** fields: **position** of type **Position**, **salary** of type **Salary** and **course** of type **Course**.
- **Utils**. This **static** class has the following **public static** methods:
  - **SeedInstitutions()** with the return type of **List<Institution>**. This method seeds a **List<Institution>** with **three Institution** objects.
  - **SeedDepartments()** with the return type of **List<Department>**. This method seeds a **List<Department>** with **three Department** objects.
  - **SeedCourses()** with the return type of **List<Course>**. This method seeds a **List<Course>** with **three Course** objects.
  - **ReadLearnersFromFile()** with no return type and two parameters: **filePath** of type **string** and **learners** of type **List<Learner>**. This method reads the **learners.txt** file and populates the **learners** parameter.
  - **ReadLecturersFromFile()** with no return type and two parameters: **filePath** of type **string** and **lecturers** of type **List<Lecturer>**. This method reads the **lecturers.txt** file and populates the **lecturers** parameter.

Program. This **public** class manages the business logic and user interface of the application. This class must account for the following functionality:

- Displaying course details
- Displaying all marks
- Displaying all grades
- Displaying highest marks
- Displaying lowest marks
- Displaying fail marks
- Displaying average marks
- Displaying average grades
- Adding a learner. When adding a learner, the **id** must be auto-generated and unique. You must prompt the user to enter the following details: **first name**, **last name**, **course** and **assessment marks 1-5**. Also, you must implement the following validation:
  - **first name** and **last name** must not be empty or, contain numbers or special characters.
  - **course** must be a valid number.
  - **assessment marks 1-5** must be between 0 and 100.
- Removing a learner
- Displaying lecturer details
- Adding a lecturer
- Exiting the application

Code Elegance - Learning Outcomes 1 and 2 (40%)

-

Documentation and Git Usage - Learning Outcomes 1 and 2 (10%)

- Provide the following in your repository **README.md** file:
  –
- Commit at least **five** times per week.
- Commit messages reflect the context of each functional requirement change.

Additional Information

- **Do not** rewrite your **Git** history. It is important that the course lecturer can see how you worked on your assessment over time.
- Grade table:

Grade	Mark Range
A+	90-100
A	85-89
A-	80-84
B+	75-79
B	70-74
B-	65-69
C+	60-64
C	55-59
C- (passing assessment marks)	50-54
D	40-49 (fail assessment marks)
E	0-39 (fail assessment marks)