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

Assessment Overview

In this **individual** assessment, you will design and develop two **Windows Forms Applications** 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	25%	13-11-2024 (Wednesday at 4.59 PM)	1, 2
Project 2	35%	27-09-2024 (Friday at 4.59 PM)	1, 2
Theory Examination	30%	18-11-2024 (Monday at 12.00 PM)	1, 2
Classroom Tasks	10%	27-09-2024 (Friday at 4.59 PM)	1, 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, 18 September 2024** 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. Do your best to complete this assessment without using an **AI generative tool**. You need to demonstrate to the course lecturer that you can meet the learning outcome(s) for this assessment.

However, if you get stuck, you can use an **AI generative tool** to help you get unstuck, permitting you to acknowledge that you have used it. In the assessment's repository **README.md** file, please include what prompt(s) you provided to the **AI generative tool** and how you used the response(s) to help you with your work. It also applies to code snippets retrieved from **StackOverflow** and **GitHub**.

Failure to do this may 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/o7D2CGoa. If you do not have not one, create a **.gitignore** and add the ignored files in this resource - https://raw.githubusercontent.com/github/gitignore/main/VisualStudio.gitignore. Create a branch called **project-2**. The latest application files in the **project-2** 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. Extensions will **only** be granted if you are unable to complete the assessment by the due date because of **unforeseen circumstances outside your control**. The length of the extension granted will depend on the circumstances and **must** be negotiated with the course lecturer before the assessment due date. A medical certificate or support letter may be needed. Extensions will not be granted on the due date and for poor time management or pressure of other assessments.

Resits

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

Instructions

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

Functionality - Learning Outcome 1 (50%)

Student Management System

The application needs to open without code or file structure modification in **Visual Studio**.

Milestone One - Due Sunday, 25 August 2024 (Week 5) at 4.59 PM

- **Institution.cs**. **public class Institution** has the following **private** fields: **name** of type **string**, **region** of type **string** and **country** of type **string**. This class has a **public constructor** that takes in the following parameters: **name** of type **string**, **region** of type **string** and **country** of type **string**.

- **Department.cs**. **public class Department** has the following **private** fields: **institution** of type **Institution** and **name** of type **string**. This class has a **public constructor** that takes in the following parameters: **institution** of type **Institution** and **name** of type **string**.

- **Course.cs**. **public class Course** 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**. This class has a **public constructor** that takes in the following parameters: **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**.

- **Utils.cs**. **public static class Utils** 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.

Milestone Two - Due Sunday, 1 September 2024 (Week 6) at 4.59 PM

- **CourseAssessmentMark.cs**. **public class CourseAssessmentMark** has the following **private** fields: **course** of type **Course** and **assessmentMarks** of type **List<int>**. Also, this class has the following **public** methods:

    - **CourseAssessmentMark()** with no return type and two parameters: **course** of type **Course** and **assessmentMarks** of type **List<int>**.
    - **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 **List<int>**. This method returns the highest passing assessment mark(s).
    - **GetLowestMarks()** with the return type of **List<int>**. This method returns the lowest passing assessment mark(s).
    - **GetFailMarks()** with the return type of **List<int>**. This method returns the fail assessment mark(s).
    - **GetAverageMark()** with the return type of **double**. This method returns the average assessment mark rounded to two decimal places.
    - **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.

Milestone Three - Due Wednesday, 4 September 2024 (Week 7) at 4.59 PM

- The application needs to contain the following **enums**:

```
public enum EPosition
{
    LECTURER = 0,
    SENIOR_LECTURER = 1,
    PRINCIPAL_LECTURER = 2,
    ASSOCIATE_PROFESSOR = 3,
    PROFESSOR = 4
}

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

- **Person.cs**. **public class Person** has the following **protected** fields: **id** of type **int**, **firstName** of type **string** and **lastName** of type **string**. This class has a **public constructor** that takes in the following parameters: **id** of type **int**, **firstName** of type **string** and **lastName** of type **string**.

- **Learner.cs**. **public class Learner** inherits from **Person** and has one additional **private** field: **courseAssessmentMarks** of type **CourseAssessmentMark**. This class has a **public constructor** that takes in the following parameters: **id** of type **int**, **firstName** of type **string**, **lastName** of type **string** and **courseAssessmentMarks** of type **CourseAssessmentMark**.

- **Lecturer.cs**. **public class Lecturer** inherits from **Person** and has three additional **private** fields: **position** of type **EPosition**, **salary** of type **ESalary** and **course** of type **Course**. This class has a **public constructor** that takes in the following parameters: **id** of type **int**, **firstName** of type **string**, **lastName** of type **string**, **position** of type **EPosition**, **salary** of type **ESalary** and **course** of type **Course**.

Milestone Four - Due Sunday, 8 September 2024 (Week 7) at 4.59 PM

- **Utils.cs**. **public static class Utils** has the following **public static** methods:

  - In the **assessments** > **project 2** directory of the **course materials** repository, you will find the implementation of the following method in the **read-from-file-learners.txt** file. **ReadFromFile()** with no return type and three parameters: **filePath** of type **string**, **learners** of type **List**<**Learner**> and **isAttendance** of type **bool**. This method reads the **learners.txt** file and populates the **learners** parameter. The **learners.txt** file contains the following information:

    ```
    1,John,Doe,0,100,100,95,10,0
    2,Jane,Doe,0,45,35,45,75,65
    3,Grayson,Orr,1,50,60,75,80,55
    4,Joe,Blogs,1,10,20,30,70,80
    5,Bob,Brown,2,75,82,95,55,10
    ```

    What do the numbers represent? The first number is the **id**, the second number is the **course** and the remaining numbers are the **assessment marks 1-5**. **Note:** You will use the second number to access a **Course** object from the **courses** list.

– **ReadFromFile()** 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. The **lecturers.txt** file contains the following information:

```
1,Graydon,Ore,1,100000,0
2,Aidan,Moscow,2,115000,1
3,Jon,Seena,0,85000,2
```

What do the numbers represent? The first number is the **id**, the second number is the **position**, the third number is the **salary** and the fourth number is the **course**. **Note:** You will use the second number to access the enum **EPosition**, the third number to access the enum **ESalary** and fourth number to access a **Course** object from the **courses** list.

Milestone Five - Due Sunday, 15 September 2024 (Week 8) at 4.59 PM

• **Form1.cs**. **public class Form1** manages the user interface. This class needs to account for the following functionality:

  – Seeding the **institutions**, **departments** and **courses** lists.
  – Reading the **learners.txt** and **lecturers.txt** files. When calling the **ReadFromFile()** method, the **isAttendance** parameter needs to be **false**.
  – Displaying course details. This needs to display the following details in a **DataGridView**:
    * course **code** and **name** in this format - **code: name**
    * course **description**
    * course **credits**
    * course **fees**
    * institution **name**, **region** and **country** in this format - **name, region, country**
    * department **name**
  – Displaying all marks. This needs to display the following details in a **DataGridView**:
    * learner **id**
    * learner **first name** and **last name** in this format - **first name last name**
    * course **code** and **name** in this format - **code: name**
    * learner **assessment marks 1-5** in this format - **mark 1, mark 2, mark 3, mark 4, mark 5**
  – Displaying all grades. This needs to display the following details in a **DataGridView**:
    * learner **id**
    * learner **first name** and **last name** in this format - **first name last name**
    * course **code** and **name** in this format - **code: name**
    * learner **assessment grades 1-5** in this format - **grade 1, grade 2, grade 3, grade 4, grade 5**
  – Displaying highest marks. This needs to display the following details in a **DataGridView**:
    * learner **id**
    * learner **first name** and **last name** in this format - **first name last name**
    * course **code** and **name** in this format - **code: name**
    * learner **highest assessment mark(s)**
  – Displaying lowest marks. This needs to display the following details in a **DataGridView**:
    * learner **id**
    * learner **first name** and **last name** in this format - **first name last name**
    * course **code** and **name** in this format - **code: name**
    * learner **lowest assessment mark(s)**

- – Displaying fail marks. This needs to display the following details in a **DataGridView**:
  - * learner **id**
  - * learner **first name** and **last name** in this format - **first name last name**
  - * course **code** and **name** in this format - **code: name**
  - * learner **fail assessment mark(s)**
- – Displaying average marks. This needs to display the following details in a **DataGridView**:
  - * learner **id**
  - * learner **first name** and **last name** in this format - **first name last name**
  - * course **code** and **name** in this format - **code: name**
  - * learner **average assessment mark**
- – Displaying average grades. This needs to display the following details in a **DataGridView**:
  - * learner **id**
  - * learner **first name** and **last name** in this format - **first name last name**
  - * course **code** and **name** in this format - **code: name**
  - * learner **average assessment grade**
- – Displaying lecturer details. This needs to display the following details in a **DataGridView**:
  - * lecturer **id**
  - * lecturer **first name** and **last name** in this format - **first name last name**
  - * lecturer **position**
  - * institution **name**, **region** and **country** in this format - **name, region, country**
  - * department **name**
  - * course **code** and **name** in this format - **code: name**
  - * lecturer **salary**
- – Adding a learner. When adding a learner, the **id** is auto-generated and unique. Prompt the user to enter the following details: **first name**, **last name**, **course** and **assessment marks 1-5**. Implement the following validation:
  - * **first name** and **last name** are not empty or, contain numbers and special characters.
  - * **course** is a valid number.
  - * **assessment marks 1-5** are between 0 and 100.

  If the validation is successful, add the learner to the **learners** list, write the learner to the **learners.txt** file and display a success message. Otherwise, display an error message.
- – Adding a lecturer. When adding a lecturer, the **id** is auto-generated and unique. The **salary** is calculated based on the **position**. Prompt the user to enter the following details: **first name**, **last name**, **position** and **course**. Implement the following validation:
  - * **first name** and **last name** are not empty or, contain numbers and special characters.
  - * **position** and **course** are valid numbers.

  If the validation is successful, add the lecturer to the **lecturers** list, write the lecturer to the **lecturers.txt** file and display a success message. Otherwise, display an error message.
- – Removing a lecturer. When removing a lecturer, prompt the user to enter the **id** of the learner to remove. If the **id** is valid, prompt the user to confirm the removal. If the user confirms the removal, remove the lecturer from the **lecturers** list and **lecturer.txt** file, and display a success message. Otherwise, display an error message.
- – Exiting the application

  If the user enters an invalid option, display an error message and prompt the user to enter a valid option.

Attendance Tracker

The application needs to open without code or file structure modification in **Visual Studio**.

Milestone Six - Due Friday, 27 September 2024 (Week 10) at 4.59 PM

- Add a **Project Reference** to the **Student Management System** application.

- **Learner.cs**. In this class, you will add an additional **private** field: **attendance** of type **int** and a **public constructor** that takes in the following parameters: **id** of type **int**, **firstName** of type **string**, **lastName** of type **string** and **attendance** of type **int**. **Note:** This class should have two **private** fields and two **public constructors**.

- **Form1.cs**. **public class Form1** manages the user interface. This class needs to account for the following functionality:

  – Reading the **attendance.txt** file. **Note:** Uncomment the two lines of code in this file. When calling the **ReadFromFile()** method, the **isAttendance** parameter needs to be **true**.

  – Displaying attendance. This needs to display the following details in a **DataGridView**:
    * learner **id**
    * learner **first name** and **last name** in this format - **first name last name**
    * learner **percentage**

  – Display learners at risk. A learner at risk is someone who has an attendance percentage of less than 50%. This needs to display the following learner **first name** and **last name** in this format - **first name last name** in a **ListBox**. **Note:** You cannot deviate from the required format.

Code Quality and Best Practices - Learning Outcome 2 (45%)

- A **Visual Studio .gitignore** file is used.

- Appropriate naming of files, variables, methods and classes.

- Idiomatic use of object-oriented principles, values, control flow, data structures and in-built functions.

- Efficient algorithmic approach.

- Sufficient modularity.

- Each file has an **XML documentation comment** located at the top of the file. In the **assessment** directory of the **course materials** repository, you will find an **XML documentation comment** example in the **xml-documentation-comment.txt** file.

- Formatted code.

- No dead or unused code.

Documentation - Learning Outcome 2 (5%)

- Provide the following in your repository **README.md** file:

  – A class diagram of your applications.

Additional Information

- Exemplars are available in **assessments** > **project 2** directory of the **course materials** repository.

- You may add additional classes and methods.

- Grade and mark range 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-	50-54 (passing assessment marks)
D	40-49 (fail assessment marks)
E	0-39 (fail assessment marks)

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