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

Project 1 (C# Console App): Learner Gradebook

Assessment Overview

In this assessment, you will design & develop a GUI implementation of the classic arcade game **Pong**.

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 & implement user-defined classes using encapsulation, inheritance & polymorphism.

Assessments

Assessment	**Weighting**	**Due Date**	**Learning Outcomes**
Project 1 (C# Console App): Learner Gradebook	25%	26-04-2023 (Wednesday at 4.59 PM)	1 & 2
Project 2 (C# Windows Forms App): Pong	35%	14-06-2023 (Wednesday at 4.59 PM)	1 & 2
Theory Examination	30%	21-06-2023 (Wednesday at 4.45 PM)	1 & 2
Classroom Tasks	10%	07-06-2023 (Wednesday 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 & your progress on this assessment. This assessment will need to be completed by **Wednesday, 26 April 2022** 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 & Resits

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

Submission

You **must** submit all project files via **GitHub Classroom**. Here is the URL to the repository you will use for your submission – https://classroom.github.com/a/eFe1Oh97. Create a **.gitignore** & add the ignored files in this resource - https://raw.githubusercontent.com/github/gitignore/main/VisualStudio.gitignore. The latest project 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 & 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 & achieved a **D grade (40-49%)**. The maximum grade awarded for resubmission will be **C-**.

Resits

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

Instructions

You will need to submit a project & documentation that meet the following requirements:

Functionality - Learning Outcomes 1 & 2 (40%)

- The project **must** open without code or file structure modification in **Visual Studio**.
- Read a text file called **data.txt** which contains information about five learners. This information includes **id**, **first name**, **last name**, three **ID510001: Programming 1 assessment marks** & three **ID511001: Programming 2 assessment marks**. **Note: data.txt must** be located in the **bin/Debug** folder.
- The learners' information is stored in a **List** of **Learner** objects. A **Learner** object **must** have the following fields:
  - **id**
  - **firstName**
  - **lastName**
  - A **List** of **int** called **prog1AssessmentMarks**
  - A **List** of **int** called **prog2AssessmentMarks**
  - A **List** of **int** called **prog1AssessmentGrades**
  -
  - A **List** of **int** called **prog2AssessmentGrades**
  - **prog1OverallGrade**
  - **prog2OverallGrade**
- A grade is calculated using the following 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-	50-54
D	40-49
E	0-39

- When the project is run, display a menu to the user that allows them to:
  - Display a learner's assessment marks, assessment grades & overall grade
  - Display all learners' assessment marks, assessment grades & overall grades
  - Display all assessment's average grade
  - Add a new learner. You should prompt the user for the learner's information & add the learner to the **List** of **Learner** objects. **Note:** The learner's **id must** be unique.
  - Remove a learner. You should prompt the user for the learner's **id** & remove the learner from the **List** of **Learner** objects.

Code Elegance - Learning Outcomes 1 & 2 (45%)

- Adhere to the four principles of **OO**, i.e., encapsulation, abstraction, inheritance & polymorphism.
- Use of intermediate variables, constants & enumerations.
- Idiomatic use of control flow, data structures & in-built functions.
- Efficient algorithmic approach.
- Sufficient modularity.
- Each method & class **must** have a header comment located immediately before its declaration.
- In-line comments where required.
- Project files, i.e., **.cs** files are formatted.
- No dead or unused code.

Documentation & Git Usage - Learning Outcomes 1 & 2 (15%)

- Provide the following in your repository **README.md** file:
  - The project's class diagram created in **Visual Studio**.
  - Known bugs if applicable.
- Commit at least **20** times per week.
- Commit messages **must** reflect the context of each functional requirement change.