College of Engineering, Construction and Living Sciences

Bachelor of Information Technology

ID511001: Programming 2

Level 5, Credits 15

Classroom Task: Unit Testing

Assessment Overview

In this assessment, you will **unit test** your 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%	22-09-2023 (Friday 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 Task: Unit Testing	10%	22-09-2023 (Friday at 4.59 PM)	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 **Friday, 22 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. 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 for poor time management or pressure of other assessments.

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 needs to open without code or file structure modification in **Visual Studio**.
- You need to create 20 **unit tests** covering the following:
  - All **properties** in the **Institution** class. **Three** tests are expected.
  - All **properties** in the **Person** class. **Three** tests are expected.
  - All **methods** in the **CourseAssessmentMark** class. **Seven** tests are expected.
  - The number of **Institution**, **Department** and **Course** objects after seeding. **Three** tests are expected.
  - The salary of a **Lecturer**, **Senior Lecturer**, **Principal Lecturer** and **Associate Professor**. **Four** tests are expected.

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

- Appropriate naming of files, variables, methods and classes.
- Idiomatic use of control flow, data structures and in-built functions.
- Efficient algorithmic approach.
- Sufficient modularity.
- Each file needs to have a header comment located at the top of the file.
- In-line comments where required. It should be for code that needs further explanation.
- Formatted code.
- No dead or unused code.

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

- Provide the following in your repository **README.md** file:
  - How to run your unit tests?
- 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.