College of Engineering, Construction and Living Sciences
Bachelor of Information Technology
IN628: Programming 4
Level 6, Credits 15

In Class Checkpoint 03: File Structure

Assessment Table

Assessment Activity	Weighting	Learning Outcomes	Assessment Grading Scheme	Completion Requirements
In Class Checkpoints	15%	1, 2, 3	CRA	Cumulative
Roguelike	45%	1, 2, 3	CRA	Cumulative
Programming Language Exploration	25%	1, 2, 3	CRA	Cumulative
Theory Exam	15%	3, 4, 5	CRA	Cumulative

Conditions of Assessment

This assessment will need to be completed by Friday, 12 June 2020.

Pass Criteria

This assessment is criterion-referenced with a cumulative pass mark of 50%.

Submission Details

You must submit your program files via **GitHub Classroom**. Here is the link to the repository you will be using for your submission – https://classroom.github.com/a/DigbcYgy. For ease of marking, please submit the marking sheet with your name & student id number via **Microsoft Teams** under the **Assignments** tab.

Authenticity

All parts of your submitted assessment must be completely your work and any references must be cited appropriately.

Policy on Submissions, Extensions, Resubmissions & Resits

The school's process concerning **Submissions, Extensions, Resubmissions and Resits** complies with Otago Polytechnic policies. Students can view policies on the Otago Polytechnic website located at https://www.op.ac.nz/about-us/governance-and-management/policies.

Extensions

Please familiarise yourself with the assessment due dates. If you need an extension, please contact your lecturer before the due date. If you require more than a week's extension, a medical certificate or support letter from your manager may be needed.

Resubmissions

Students may be requested to resubmit an assessment following a rework of part/s of the original assessment. Resubmissions are completed within a short time frame (usually no more than 5 working days) and usually must be completed within the timing of the course to which the assessment relates. Resubmissions will be available to students who have made a genuine attempt at the first assessment opportunity. The maximum grade awarded for resubmission will be C-.

Learning Outcomes

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

1. Program effectively in an industrially relevant programming language.
2. Implement a wide range of intermediate data structures and algorithms to act as modules of larger programs.
3. Use an appropriate integrated development environment to create robust applications.
4. Demonstrate sound programming and software engineering practices independent of the environment or tools used.
5. Explain the theoretical issues surrounding programming language design and development.

Assessment Overview

In this practical, you will complete a series of tasks covering today's lecture. This practical is worth 0.5% of the final mark for the Programming 4 course.

Task 1

In this practical, you will build two classes representing Monsters and Wizards. These two classes have no data members. They both have a constructor, and a single method called Speak(). When a Monster speaks, a message box will pop up containing the following text "I am a Monster... Roar!!! ". When a Wizard speaks, a message box will pop up containing the following text "I am a Wizard... Expelliarmus!!!".

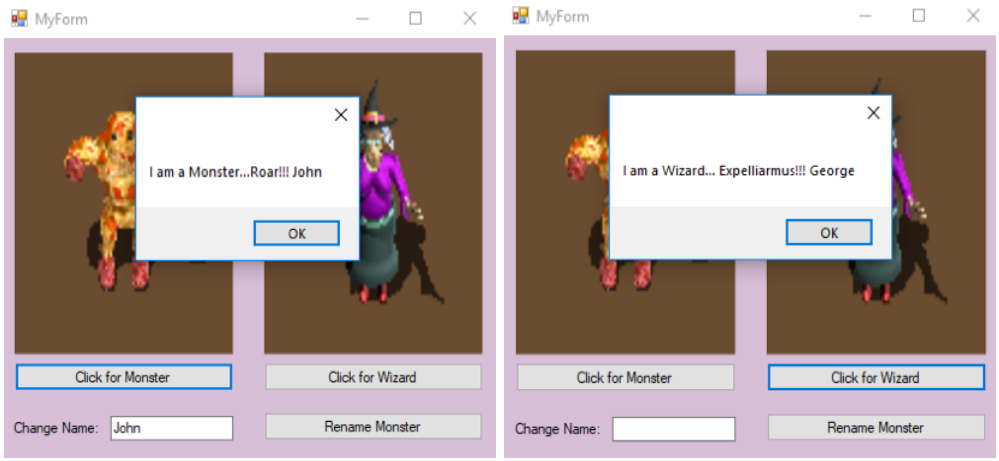
- Create a new C++ CLR project or use the empty project in Microsoft Teams
- Create a Form with two PictureBoxes and two Buttons, as shown in the demo. Use the provided images, or find your own if you prefer
- Implement the two classes:
  - Create a new class via Project− >Add Class
  - Remember to copy the using statements from MyForm.h into each new .h file
  - Delete anything you want from the .h and .cpp files Visual Studio creates for you except the compiler directives – these are the statements that begin with #
  - Place the class definitions in the .h files and place the implementation code in the .cpp files – please don't define Speak() methods inline. Visual Studio has a wizard tool which generates the class definitions
  - Remember to preface each method in the .cpp with the name of the class. For example, I have the following line in my Monster.cpp - void Monster::Speak()
  - Since these classes have no data members and need no initialisation, their constructors will be empty
  - Add a Monster and Wizard object to the MyForm class as data members. Remember to use handlers (managed pointers)
  - Instantiate your Monster and Wizard object in the MyForm_Load event. Your form's name may be different to what is specified here
  - Write the handlers for each of the two buttons so that your application behaves like the demo

Think about: An application's "class architecture" is the collection of classes it contains. Can you think of a better architecture for this program? (Hint: When you have two classes that have nearly identical code, there is almost always a better architecture.)

Task 2

- Add a String data member to the Monster and Wizard class to hold its name. Pass the value into the constructor, and add it to the output of Speak(), as shown in the image below
- Add necessary controls to your MyForm, and modify the code of your Monster class, to allow the user to dynamically change the name of the Monster at run time. Please carefully check for an empty TextBox

Expected Output



Submission

- Create a new branch named 03-checkpoint within your practicals GitHub repository
- Create a new pull request and assign Grayson-Orr to review your submission
- Deadline: Wednesday, 11 March at 5pm

Note: Please don't merge your own pull request.