Faculty of Science and Engineering

Coursework – 2022/23 Academic Year

Module Code: **COMP1000**

Module Title: **Software Engineering 1**

Module Leader: **Dr Swen Gaudl**

School: **School of Engineering, Computing and Mathematics**

SUBMISSION INSTRUCTIONS FOR CANDIDATES
Coursework must be submitted electronically using the online submission facility in the DLE by the published deadline. Your submission should be anonymous, i.e. do not write your name on your paper. When you submit your assessment you are stating that it is your own work; Please ensure you are familiar with the regulations regarding Academic Offences. Normal referencing guidelines should be followed when citing other people's work. If you have any queries on submission or in relation to the work, please contact the Faculty Office on 01752 584584 or science.engineering@plymouth.ac.uk immediately so any problems can be rectified.

W2 – Individual Project: Completing a Software Project 70%

DESCRIPTION

This is an individual piece of work with the goal to develop a consistent and homogenous project. This coursework concentrates on implementing coherent pieces of code that integrate into a single piece of software. In addition, **your task is to document and demonstrate how it works** and how the code fits together. The coursework is to complete and extend a given piece of software provided through Github classroom. As part of the software engineering task, we want you to demonstrate the use of learnt principles from the lectures such as OOD and how to write good code. To finalise the coursework a short video demonstration with code walkthrough should be given in the form of a presentation. The video should be uploaded to youtube as unlisted or submitted directly to the dle; it needs to be accessible for at least half a year.

RULES OF ENGAGEMENT

You may not use any other than the **provided codebase which is different from the regular coursework codebase** directly. This includes code from web sources (i.e. “gamedev”, Nvidia, tutorialpoint, youtube, redit) but you may use algorithms and implement them yourself. *If you and another student share the same code both codebase will be seen as a fail.* As part of your documentation on your Github page, you **must** tell us what resources you have used and link the video. If this is not done, you risk failing the assignment.

TASK OF THE PROJECT

The given codebase is a template for a simple 2D command line dungeon crawler. Some parts of the code are missing which need to be completed by you. Other parts are already present can be extended further to make the project more impressive. You may not remove any of the basic structure or ways of interacting with the software. Do not rename existing variables, methods, classes but feel free to add to them.

HANDLING THE GIVEN CODEBASE

You may not change the class structure of the provided code and the way objects interact. You may add functionality but you may not remove or alter any of the provided methods as they will be used for testing as well. The entry point of the software should not be changed but you can extend the protocol of interaction.

REQUIRED BASIC BEHAVIOUR

- **Instead of simply modifying the Console the internal memory needs to be changed as well.**
- The provided tests need to run without any additional changes or input!
- If starting the Dungeon.exe:
    - The user needs to be able to start the project from command line by simply calling the executable name, e.g. “Dungeon.exe”
    - The user needs to be able to pick and load a local map file from the maps folder by typing in “load Simple.Map” followed by **ENTER**
    - The user can play a loaded map by typing in “start” followed by **ENTER**
- Your software needs to be able to load and display the “Simple2.map” on a 2D grid
- The player always starts on “P“.
- Using the **W,A,S,D keys** the user should be able to move and complete the map by entering the “X” tile.
- Using “**Z**” the player can pick up a coin when standing ontop of it.
- Players do not need to press ENTER when making a move.
- The number of steps the player has taken is shown underneath the map.
- Map elements:
- “M” are monsters which the player cannot pass (Monsters do not move),
    - “#” are walls which cannot be passed,
    - ”.” Are empty spaces which can be passed
    - “C” (Coins) can be seen as empty
    - “D” is the exit which upon entering ends the map
    - “@” is the player
    - “P” starting point of the player

ADVANCED BEHAVIOUR

- This extends the basic behaviour.
- Using the command “advanced” before typing in play is used to enabled the advanced mode **only then** are advanced features working.
- Without adding “advanced” the game should be in basic mode without moving monsters
- Using the **“Q”** key the player should be able to attack a position dealing 1 damage to a Monster.
- The advanced map can be loaded and completed by the user.

- When player moves over a tile the tile is hidden and the player symbol is rendered until player leaves the tile.
- Map elements:
    - “M” are monsters (which you can move over empty spaces),
    - “#” are walls which cannot be passed,
    - “C” are gold pieces which can be collected or walked over
- Extras:
    - Monster may have 3 damage point.
    - An attack from the player only deals 1 damage
    - Players heal one damage when they collect a coin and start with 2 health.
    - Monsters may attack
    - Monsters can eat coins to get stronger
    - When Monsters die they drop their coins
    - Upon Entering the exit players get displayed a status message and can either reply or quit the game
    - Allow for a Replay of the map using the command: “replay”

DELIVERABLES

- Create a **single “Dungeon-<student-id>.zip”** archive containing the executable & the README.md and all required files to run your software project. Submit the zip via the DLE electronic submission system. And don’t forget to remove any debug or temporary files and folders.
- Additionally, copy the source code of **CMD-Crawler.cs** into a word document and submit this with the name "**Crawler-<student-id>.doc**" also to the DLE. (*make sure the upload works correctly*)
- When accepting the second assignment, the starting code base will be cloned which you will need to modify and **push back** once you completed the **assignment**. As part of your Github page, add some basic information about your project in the README.md and layout the project page using markdown to create a descriptive entry point to your project. Also include the video as part of the description into the Github page.
    - **The Github page should cover**:
        - How does the user interact with your executable?
        - What resources including books and algorithms is your software using or based on.
        - Which advanced behaviours have been implemented? *(not mentioning them will count as not implemented)*

- Video link/Video iframe

- Record and upload a **video brief** (no shorter than **7** but only up to **10 minutes**) of your prototype to YouTube as unlisted, which should address:
    - Add timestamps on youtube for your main points.
    - How does the user interact with your executable? How do you open and control the software you wrote (exe file)?
    - How does the program code work? How do the classes and functions fit together and who does what?
    - What software engineering paradigms did you use and how?
    - ***Show that it works in your IDE and as executable and compiles!***
    - Are there any software engineering issues or shortcomings of your software?
    - A (brief) evaluation of what you think you have achieved, and what (if anything) you would do differently, knowing what you now know. Draw my attention to anything you are particularly pleased with.
    - Upload the video to **youtube** (unlisted)

Deadlines:

Part	Description	Deadline	Percentage
	Github classroom Submission of project: *Upload your project by the deadline to your assignment repo* Include the video walkthrough and an entry page (README.MD)	14th of January	
	Individual Project: *Upload your Project to the DLE* Add a text file with the Link to the video walkthrough	14th of January	

MARKING RUBRIC

Each rubric covers a bracket from a given percentage to the next. If you complete all elements you move up to the new bracket. Is a bracket not fully completed no elements of the higher bracket count towards the mark. **The tests provided on github map onto the marking rubric so you can check where you are in terms of the code**.

Category	Fail	> 40%	> 50%	> 60%	>70%
Interactive Project (70%)	Software does not compile or execute. Video(Link) is missing. Github classroom submission missing. Software crashes or does not offer basic interaction. **Simple.Map is not loading using the tests** **Player cannot move through test code.** Tests have been modified.	Basic behaviour is mostly working. Map is updating when player moved. The Github page contains some details. The video contains a demo and basic description of the software and its design. The Init Tests work.	Simple Map can be completed. The Github page is concise and explains usage and structure of the program Video contains subtitles or audible explanations Protocol contains only a minor violation (code changes or interaction changes) **Submitted zip & doc correctly according to specs.**	The software does not crash/hang. Advanced behaviour can be enabled disabled. Monsters only move when in advanced mode. Demo video showed good understanding of the code. The player can move without requiring the user to press enter. **All tests pass.** **Video is done correctly.**	The presented project shows a strong contribution. Advanced Behaviour fully included. The Demo video showed a good understanding of used date structures, algorithms and code integration. In the video, arguments and Evaluation of the topic and the prototype are sound.

Please refer to all the lecture content & further study resources on the DLE.