COMP1000 Software Engineering 1

20 CREDIT MODULE / 100% COURSEWORK SUBMISSION

MODULE LEADER: Swen Gaudl
MODULE TUTOR: -

MODULE AIMS

- To familiarise students with the fundamentals of software programming.
- To expose students to software engineering methodologies and good practices.
- To understand widely used software engineering paradigms such as object orientation and functional programming.

ASSESSED LEARNING OUTCOME:

- Employ fundamental programming constructs such as control structures and data types.
- Describe widely used programming paradigms such as object orientation and functional programming.
- Select appropriate software development tools, techniques and environments to aid the implementation of simple software.

OVERVIEW

This module introduces concepts & paradigms for writing software. It covers computational thinking and algorithmic thinking as guiding principles of programming. It provides an introduction to industry relevant skills and tools useful across all Computer Science domains. The module uses a bottom-up approach; **no prior programming knowledge** is needed though it might be beneficial. COMP1000 teaches object oriented design (OOD) using high-level programming languages, ie. C#, as well as introduces other programming paradigms such as functional/procedural programming. Students will gain highly transferrable software engineering skills which will be crucial in the long run.

Lectures, Seminars and Workshops are integrated into the module to introduce concepts and deepen the understanding of topics as well as to guide the project work.

100% Coursework Comprising Two Elements

W1: Set Exercises 30% Working individually on a number of set exercises which, taken together and averaged, count as the first assessment.

W2: Documented Software Project 70% Working individually on a small self-contained project based on the supplied code-base. Record a short video which explains the developed code and how it relates to software paradigms and where and why they have been employed.

MODULE DELIVERY

Delivery format:	Weekly 2hr zoom workshop Monday from 11 am – 1 pm Weekly 2hr zoom workshop Thursday 11 am – 1 pm
Delivery staff:	**Swen Gaudl – swen.gaudl@plymouth.ac.uk**
Duration:	**12 weeks Semester 1**

Assessment Offences:

For this assignment you may be using information from differing sources:

- Books, journal articles
- Course/module materials
- Websites
- Existing OpenSource Projects

It is ***very important*** for you to note that these assignments are *an **individual effort.** It **should make the contributions from the student** and the use of external resources or an initial start base clear.*

Thus, do not simply copy existing sources, i.e. other students work, interspersed with a few lines of code or words of your own. This is paraphrasing, and it is not encouraged, it is not likely to get you a good mark and in some cases it could be seen as plagiarism. In a similar vein, do not simply copy material from elsewhere without citing it properly.

For more **information** on how to write texts, reference source material and plagiarism in general, see:
https://www.plymouth.ac.uk/student-life/your-studies/essential-information/regulations/plagiarism
If you have any doubt as to what constitutes '***an individual effort and in your own words***' then either see your student handbook or see me.

W1 – Set Exercises 30%

DESCRIPTION

This is an individual assignment. It will help us to estimate where you currently are in terms of capabilities.

The ultimate aim is for you to demonstrate that you can understand different software paradigms and solve computational problems. To evaluate your performance, we will use a set of exercises which contain programming tasks and coding puzzles where you have to complete or fix some piece of code and submit your answer to the Github classroom repository.

To complete W1, you need to tackle the given exercises and pass with a score of **at least 4 points** in sum over all exercises. Each exercise is worth one point and each test in an exercise is a fraction of that. The exercises can be tried multiply times but need to be completed by the given deadline. It can take between a couple of minutes to an hour or two to complete an exercise. ***Try to do the exercises as early as possible, late submissions result in zero points for the specific exercise! You do not need to complete all exercises or finish all tests but you should try to complete as many as possible!***

Where to begin?
Sign up to the COMP1000 Github classroom through the link you should have received in an email and pick the first W1 assessment. Do not override the test files or modify any of the files other than the specific exercise files for each of the assignments. Not all assignments will be visible at once, check the lectures for when assignments will be open. You can test against the given test-sets but submit your final solution before each of the deadlines.

DELIVERABLES

- Before the deadline, upload onto the COMP1000 Github classroom for each of the assignments separately the required files. You should see a score for each of the tests.

Deadlines:

Part	Description	Deadline	Percentage
	Set Exercises: *Submit through Github for each assignment a modified C# repository.* ***ONLY*** *modify the files that address the challenge. Each exercise will have its own score and submission.*	The specific deadlines will be announced on the dle.	

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.

Category	Fail	> 40%	> 60%	> 80%
Set Exercises (30%)	Receive a score of no more than 3 over all exercises. Use code available online. Hard-code answers to pass the tests. Modify test files.	Pass with a score of more than 4 over all exercises.	Pass with more than 6 points. Code contains meaningful comments for own code.	Pass with above 7 points. Individual code shows good structure when required to change. Good naming of variables, when required to create new ones.

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

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 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 can either be uploaded to youtube as unlisted or to your onedrive but needs to be accessible for at least half a year.

Rules of Engagement
You may not use any other than the provided codebase directly from web sources (i.e. “gamedev”, Nvidia) but you may use algorithms and implement them yourself. As part of your documentation on your Github page, you **must** tell us what resources you have used. 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 then way objects interact. You may add functionality but you may not remove any as those 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:

- The user needs to be able to start the project from command line by simply calling the executable name, e.g. “Crawler.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 “play” followed by **ENTER**
- Your software needs to be able to load and display the “Simple.map” on a 2D grid
- The player always starts on “S“.
- Using “**WASD**” the user should be able to move and complete the map by entering the “E” tile.
- Map elements:
    - “M” are monsters which the player cannot pass,
    - “#” are walls which cannot be passed,
    - ”.” Are empty spaces which can be passed
    - “G” can be seen as empty space
    - “E” is the exit which upon entering ends the map
    - “@” is the player
    - “S” starting point of the player

Advanced behaviour:

- This extends the basic behaviour.
- Using **SPACE** the player should be able to attack a position dealing 1 damage to a Monster.
- Using “**E**” the player can pick up gold when standing ontop of it.
- Players do not need to press ENTER when making a move.
- 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,
    - “G” are gold pieces which can be collected or walked over
- Extras:
    - Monster may have more than 1 damage point.
    - Monsters may attack
    - Allow for a Replay of the map

DELIVERABLES

- Create a **single “.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.
- 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.
        - Video link/Video iframe

- Record and upload a **video brief** (around **10 minutes**) of your prototype, which should address:
  - 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 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.

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* Share a 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.

Category	Fail	> 40%	> 50%	> 60%	>70%
Interactive Project (70%)	Software does not compile or execute. Video is missing. Github classroom submission missing. Software crashes or does not offer basic interaction. Simple.Map is not loading or showing Player cannot move	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. Protocol contains some violations (code changes or interaction changes).	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 according to specs.	The software does not crash. No violations in the code or behaviour. Some elements of advanced behaviour are included. Demo video showed good understanding of the code. The player can move without requiring the user to press enter.	The presented project shows a strong contribution. Advanced Behaviour 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.