

# Faculty of Science and Engineering

Coursework – 2022/23 Academic Year

**Module Code:** COMP1000

**Module Title:** Software Engineering 1

**Module Leader:** Dr Swen Gaudi

**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](mailto:science.engineering@plymouth.ac.uk) immediately so any problems can be rectified.

## 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

### 100% Coursework Comprised of Two Elements

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

**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.

## 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 a fixed number of points 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. **The description for an exercise details the number of points it is worth and what specific rules apply.** 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!***

Each exercise has a separate deadline announced on the DLE. When **claiming ECs** for the exercises you can only claim non-submission for them. This results in you not being able to submit any further exercise and if the module is referred you are able to re-take the exercises as a first attempt. If you pass the module with W2, you cannot re-take the exercises even if you submitted ECs for them as the module has already been achieved.

### WHERE TO BEGIN?

Sign up to the COMP1000 Github classroom through the link from the DLE 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.

## GUIDELINE

- Do **only** modify the required files
- If you are not used to git only use the command line tools for submission
- Test on command line and visual studio but never rely only on Visual Studio.
- Be careful what you submit to the repository. A broken repository results in zero points for an exercise
- Check the score for each exercise by looking at your local score and the online score.

## DELIVERABLES

- Before the deadline, upload onto the COMP1000 Github classroom for each of the assignments separately the required files. Running the given tests on your machine, you should see a score for each of the tests.
- On Github, you should see a score for each of the tests, submitting untested code will result in large penalties.

### Deadlines:

Part	Description	Deadline	Percentage
	Set Exercises: <i>Submit through Github for each assignment a modified C# repository. <b>ONLY</b> modify the files that address the challenge. Each exercise will have its own score and submission.</i>	The specific deadlines will be announced on the dle but there will be a week time to work on one exercise.	

## 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. If a bracket is 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 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 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, reddit) 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 on top 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 enable 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: <i>Upload your project by the deadline to your assignment repo</i> Include the video walkthrough and an entry page (README.MD)	17 <sup>th</sup> of January	
	Individual Project: <i>Upload your Project to the DLE</i> Add a text file with the Link to the video walkthrough	17 <sup>th</sup> 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. If a bracket is 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%)	<p>Software does not compile or execute.</p> <p>Video(Link) is missing.</p> <p>Github classroom submission missing.</p> <p>Software crashes or does not offer basic interaction.</p> <p><b>Simple.Map is not loading using the tests</b></p> <p><b>Player cannot move through test code.</b></p> <p>Tests have been modified.</p>	<p>Basic behaviour is mostly working.</p> <p>Map is updating when player moved.</p> <p>The Github page contains some details.</p> <p>The video contains a demo and basic description of the software and its design.</p> <p>The Init Tests work.</p>	<p>Simple Map can be completed.</p> <p>The Github page is concise and explains usage and structure of the program</p> <p>Video contains subtitles or audible explanations</p> <p>Protocol contains only a minor violation (code changes or interaction changes)</p> <p><b>Submitted zip &amp; doc correctly according to specs.</b></p>	<p>The software does not crash/hang.</p> <p>Advanced behaviour can be enabled disabled.</p> <p>Monsters only move when in advanced mode.</p> <p>Demo video showed good understanding of the code.</p> <p>The player can move without requiring the user to press enter.</p> <p><b>All tests pass.</b></p> <p><b>Video is done correctly.</b></p>	<p>The presented project shows a strong contribution.</p> <p>Advanced Behaviour fully included.</p> <p>The Demo video showed a good understanding of used data structures, algorithms and code integration.</p> <p>In the video, arguments and Evaluation of the topic and the prototype are sound.</p>

Please refer to all the lecture content & further study resources on the [DLE](#).