CM10228 / Programming 1B:

Assignment 2 (2016): Graphical User interface

Introduction:

Assignment Date: 10th March 2016

Due date: Wednesday 6th April 2016 @ 5pm

Overall, your mark in Programming IB is composed of

a. 50% coursework,
b. 50% exam.

The coursework component (part a., above) is made up of three exercises (CW1, CW2 and CW3) each of which will build upon the last. Those three exercises ('courseworks') are as follows

CW1. Dungeon of Dooom: Extending prewritten code to allow multiple concurrent, networked agents in one dungeon (Java)
CW2. Dungeon of Dooom: Adding a GUI (Java)
CW3. Dungeon of Dooom: Extending game logic (C)

This document provides requirements for the second coursework (CW2). CW2 is worth 1/2 of the coursework component (i.e. 25% of your total mark for the unit). This means that it is worth more than CW1 and CW3 but may take you more time to complete.

Core Requirements:

This assignment asks you to make further extensions to the **Dungeon of Dooom**. More specifically, it asks you to plan, build and assess your own Graphical User Interface (GUI).

To pass, you must

- Write a specification and test plan (see below)
- Make at least one GUI that allows users to interact with your dungeon (again, see below)
- Document and analyse your work

For high marks, you should

- Make this GUI elaborate
- Build several different GUIs for different tasks and/or make your AI more elaborate.

For most students, we expect that you will be extending/amending the code that you produced in Assignment 1 (CW1). If, however, you would prefer not to extend your code, you can also extend from our code (i.e. the code that we gave out with the specification for assignment 1) and still pass.

Note: Starting from our code will reduce the number of options available to you later in this coursework but you can still pass.

More Detail:

The coursework will be marked out of 100. Marks will be awarded in the following areas:

1. Informal specification & test plan (max 10 marks)
2. Fully commented, working program implementing your design (max 80 marks)
3. Documentation and analysis (max 10 marks)

Total 100 marks

See below for more details on each of these points.

1. Informal Specification and Test Plan (max 10 marks):

Write an informal specification / architecture for your application ***before*** you start anything else. Include

i. Diagrams that describe the
   a. UI that you want to build
   b. Classes that you will use / implement
ii. A test plan
iii. A feature table, which describes the features of your GUI that will gain you marks. Your feature table should look something like this:

Intended Features	Maximum Points Available (From Specification)	Self Evaluation (Does the intended feature work)	Special Instructions
Basic GUI	15	Yes	none
Additional 'World' GUI	15	Yes, pretty much. The only exception is …	You have to use a special command line argument to get this running. It should look like this ….
…			
and so on			

2.a. Develop a Basic GUI (Max 20 marks):

The most basic objective, which must be completed for a passing mark is

Creating a GUI for human users

This part is mandatory, and is worth **20** points.

At a minimum, this must:

- Operate in a window,
- Have a control panel consisting of buttons
- Have a pane that shows the user the outcome of their actions.

More specifically, you should to build a User Client GUI application with some labels, a large pane for showing feedback from the server, and a number of buttons. ***In the rest of the instructions below, this will be called the "Player GUI", because you may choose to build a second GUI later.***

Please note:

- Buttons (or other widgets of your choice) should let you control a character in your dungeon.
- One button should let you quit the application.
- Use labels to make it clear how to use your application.

Please also note:

- Ideally, the Player GUI should allow you to connect to the client code that you produced in Assignment 1. If that is the case and you want it to connect with other people's servers, you should also provide editable text fields so that your user can enter IP addresses and port numbers for connecting to them.
- If more than one person is playing in the dungeon, or an AI is also playing in the dungeon, then some things might happen without the user doing anything. However, you can pass this coursework if you just make a window allowing a single human user to play the game by themselves.

2.b. Add TWO 'Advanced' Features (Max 25 marks each / 50 marks total for functionality plus 10 marks for comments, clarity and style):

In order to get full marks on the coursework, you will need to add **two** 'advanced' features to your GUI/game. The list below describes the choices that you have when choosing these advanced features. **You should not try to do everything!** We will only mark two advanced features**.** Please, do save some time for your other courses.

When choosing from the list, below, please do select the options that you think will be the most fun to work on. You may also want to think about demoing this program for someone who might give you a summer job.

Please note:

- Advanced features will be marked differently from the basic GUI. It will be easy to get 20 out of 20 for the basic GUI by following the instructions in this document. For the advanced features, however, we're expecting you to go over and above what's in the spec. **Creativity will be rewarded!**
- Functionality alone doesn't get you high marks! There should be good code layout, good comments, good documentation, good testing and good exception handling
- When you are filling in the Feature Table as part of your specification (see below) remember that you can also get some marks for things even if you haven't got them working in time for the hand in. This is particularly true in the specification and analysis sections. So be optimistic in your initial specification. Have fun!

List of advanced features (Each worth 25 marks):

Advanced Feature 1

Add a graphic pane to your Player GUI to display what is going on without resorting to scrolling text.

- Update this pane for each player move
- Update this pane for each bot move
- Show player/bot progress towards winning the game

Advanced Feature 2

Have your Player GUI display update in response to events from the server. That is, let the server send an event to the Player GUI and have that GUI update even though the player hasn't done anything. For example, an AI bot may have walked into the room.

- You can only do this if you use a client / server architecture, which you will know about after doing CW2.
- The events will come from AI or other people's clients. But your GUI should only be hearing about the world from the sever that contains the game engine.

Advanced Feature 3

Create another GUI (the Game Engine GUI) for the game engine when it's a server (**requires** client / server architecture).

- Create a "god's eye view" pane that lets you watch everything that happens in the dungeon. This is unlike the player's view, which only shows what the player can know. Note that this can help you debug your AI, so if you are going to work on AI you should probably do this first.
- Report the IP address for your server.
- Use radio buttons to turn the actual serving for clients (the listening) on and off. Allow a new port to be selected, but disable this if the server is currently serving / listening.

Advanced Feature 4

Create another GUI (the Game Engine GUI) that allows you to watch what is going on in your dungeon when it's **not** a server (**not** using client / server).

- Create a "god's eye view" pane that lets you watch everything that happens in the dungeon. This is unlike the player's view, which only shows what the player can know. Note that this can help you debug your AI, so if you are going to work on AI you should probably do this first.
- Make this GUI the main GUI that launches when you start the program.
- Make sure it allows you to open & close the GUI for the human user.
- Make sure you allow the user to "blank" the "god's eye view" panel so they can play the game without cheating if they want to.

Advanced Feature 5

Add more types of interactions between bots and/or players.

- After coursework 1, you probably have bots and players that compete against each other
- If you select this option, however, you should make bots that
  - help (e.g. donate each other gold, guide each other.)
  - hurt (e.g. hunt and destroy each other)
  - compete as before
- You may want to add more possible items into the dungeon too (e.g. weapons, cameras, mobile phones).

Advanced Feature 6

Make multiple kinds of bots for your dungeon that don't interact.

- Create two or three different bot types with different strategies.
- In your write up, show that these bots work differently, e.g. that one strategy can win the game faster than another.
- **10** points for a variety of simple bots, or **25** points for substantially different planning systems or representations and experiments to show they are different.

Advanced Feature 7

Make another GUI for launching AI bots, or add a pane to your Game Engine GUI to do this.

- If you have more than one type of bot, this should let you pick which one to launch.
- Add a slider for each bot to control how fast or slow it goes (the "sleep" statement introduced in CW 2).
- Report how many bots are currently alive in the dungeon.
- This may be combined with the Game Engine GUI if you want to, or it can be its own application / window.

Advanced Feature 8

Add a chat capacity to your Player GUI

- Again, you can only do this if you are using a client / server architecture.
- Add an editable text pane that allows you to type words to other players, and another that reports what they type.
- You will want to have names to identify different participants.
- Note the text will have to go through a server.
- A basic chat will get you 10 points. To get the full 25 you will need to add more features.

Documentation and Analysis (10 marks)

Finally, include a short **(2 to 3 pages)** write-up documenting your program. This should include

- A screenshot of your GUI.
- Program documentation / HOWTO / README. Make sure anyone could run your program from your documentation. Try your non-CS friends, or family if they are online.
- Critical analysis of your program:
  - What did you do right?
  - What did you do wrong?
  - What would you do differently if you did it over?

Be sure to draw attention to any work that you think was particularly clever or difficult – we have to check a lot of projects, but we don't want to miss out on giving you credit for something cool.

How you will be marked

This coursework must run on the Java installed on LCPU, which is not necessarily the most recent one. Check this before you submit!

What follows is a rough guide to how you might expect to be marked. That rough guide provides an indicative description of what might achieve particular degree categories.

	Specific Criteria	Maximum Marks
Information Specification	Requirements/Architecture	5
	Test Plan	5
Code	Commenting, Formatting and Clarity of Code	10
	Basic GUI	20
	Advanced Feature 1	25
	Advanced Feature 2	25
Results and Analysis	Documentation/Instructions	5
	Analysis and Suggested Next Steps (Improvements)	5
Total		100

Threshold for Pass (40%+)

- Brief discussion of requirements, design. A test plan.
- Simple program. Basic level of commenting. Code layout is mostly correct. Minimal exception handling necessary for a compile.
- Minimal functionality:
- A basic GUI for a user playing the **Dungeon of Dooom**.
- Minimal documentation so that the system can be run without making the marker hack it.

Good Pass (~55%)

- Specification covers all aspects of the problem at an appropriate level of detail. Good program design. Good plan for implementation that includes sensible testing.
- Program is a good implementation of design, or possibly better than the design. Code is concise and comprehensible. Layout of code is correct and clear. Some reasonable exception handling, e.g. reasonable messages or steps taken on likely exceptions.
- The GUI or GUIs are clean and make the way they should be used apparent.
- Basic functionality:
- Everything in the threshold pass, and
- Has at least 15 points worth of extra features working correctly.
- Documentation clear, concise and helpful. Sensible criticism of your work, and good ideas for improvement.

Distinction Pass (70%+)

- Specification describes the system coherently. Design description starts with a high level design and showing refinement to appropriate levels. Algorithms and implementation plan are given at an appropriate level of detail and are well explained. Sophistication in testing approach e.g. automation, unit testing.
- A sophisticated program showing good use of Java, but not overly complex. Commenting is appropriate and gives sufficient information, but is not overly verbose. Layout of code is correct and clear. Fully appropriate use of exception handling.
- GUIs are clean, their functionality / usability is apparent
- All or nearly all advanced functionality works well.
- Documentation clear, concise, attractive and informative. Detailed and thoughtful criticism of your work, and well-thought-out ideas for improvement, possibly including descriptions of alternate implementations.

What to hand in:

By the date/time specified above, you should upload a zip file which must be in the following format. The name of the zip file should be in the form

CW2-<your id>.zip

for example : CW2-abc123.zip

This zip file must contain

- Your documentation and analysis document **as a PDF**
- A **project folder** which contains
  - your source code
  - if needed, including our source code
  - README with simple instructions of how to start your game - e.g.

> To run the game compile and run these classes (they contain the main methods)
>
> Server MyTextServer.java – text UI server
>
> Server MyGUIServer.java – GUI server
>
> Client (Human) PlayGame.java – will launch a human client GUI
>
> Client (Simple Bot) SimpleBot.java – results can only be seen if you run MyGUIServer

  - NOTE: the folder should not contain compiled class files or any additional files not needed to run the game

Upload the file to Moodle on or before the deadline specified. Remember, marks are not given just because your code compiles and runs. It should run robustly, and it should be well commented and well documented.

Final Notes:

Whilst satisfying the requirements, above, please note:

- You are encouraged to use the eclipse IDE in programming and debugging your code. However, your code must run from the command line, and should work on LCPU. But, your code should be platform and operating-system independent i.e. should not rely on any external libraries which are only compatible with for example, UNIX or Windows.
- This assignment is expected to take the average student (one who got about 58 in Programming I) about 23 hours. As mentioned in class, this is a double unit, so it is expected you spend at least 12 hours a week of self-study on the course, besides the two hours in lab and the two hours in lecture. We expect you will spend about 7.5 self-study hours a week for the next two weeks on this assignment, in addition to the two hours you have in lab. (Notice

this still leaves you plenty of time for lecture preparation.) Because programming takes different people radically different amounts of time, some students will spend something more or less than 15 hours on this assignment.

- An important note on plagiarism: Plagiarism occurs any time that you use someone else's ideas without acknowledging them. Plagiarism is much, much worse than asking for help – in fact, asking for help isn't bad at all. If you get help on any part of your assignment you should acknowledge this in the comments. This includes asking coursemates, tutors, or other staff questions, and of course any code or ideas you get from books or the Internet. Say exactly where your ideas came from, and exactly how much of the code you hand in is your own.