CM10228 Coursework 1
Dungeon of Doom - Part 2

February 10, 2017

1 Introduction

Due Date: 17.00 on 3rd March 2017

Overall, your mark in Programming 2 is composed of,

1. 50% coursework,
2. 50% exam.

The coursework component (part 1. above) is made up of three exercises (CW1, CW2 and CW3) each of which will build upon the last. These are as follows

1. CW1: **Dungeon of Doom Part 2:**
   Extending code from CM102227 CW2 –or the supplied code- to allow multiple concurrent, networked agents in one dungeon (Java).
2. CW2: **Dungeon of Doom Part 3:** Adding a GUI (Java).
3. CW3: **Dungeon of Doom Part 4:** Extending game logic (C).

This document provides requirements for the first coursework (CW1). CW1 is worth 1/5th of the coursework component (i.e. 10% of your total mark for the unit). This means that it is worth less than CW2 and CW3 but should take you less time to complete.

2 Game Specifications

2.1 Core Requirements

This coursework emphasises networking, threading and concurrency. It also tests your ability to read and extend previous submitted code or other people's code. To complete the coursework, you will need to extend the Dungeon of Doom game to provide the following additional functionality:

1. Your version of Dungeon of Doom should work on a client/server model.
2. Your code allows you to play Dungeon of Doom against one or more of your bots in the same dungeon. You should also be able to support one or more of your friends and their bots joining you in the dungeon.
3. No location in the dungeon can be occupied by more than one actor (i.e. human players or bots) simultaneously.
4. Implement a basic chat system. Clients can send messages to the server for broadcasting to all other clients using the protocol SHOUT <message> e.g. SHOUT Hello World

You can choose to add this new functionality to either 1. the code you submitted for CW2 in CM10227 or 2. to use our example code available on Moodle.

2.2 Core Requirements - Automated Testing

Important: The client and server code you submit will be tested using a test suit which will check to see whether your server code can accept connections from both your, and our, client code and whether your client code can connect to our server. To allow us to run this test suit your submitted code **MUST** use the three classes below which are included in our given code on Moodle.

1. **DODServer:** - runs your server implementation using the port number supplied.
   Usage: java DODServer <portnumber>
2. **HumanClient** - runs your human player client implementation using the hostname and port number supplied.
   Usage: java HumanClient <hostname> <portnumber>
3. **BotClient** - runs your bot player client implementation using the hostname and port number supplied. If you have implemented multiple bots, choose the one that showcases your programming skill the most!
   Usage: java BotClient <hostname> <portnumber>

Failure to follow the submission specifications, would result a *-10* penalty applied to the final mark.

2.3 Advance Features

2.3.1 Advanced Chat

Implement an advanced chat system. Marks will be awarded if every client uses its own identification system, (e.g. user name and/or messages colouring) such that clients can send private messages to each other

2.3.2 Chat Log

Implement a chat log. Marks will be awarded if you keep a history log of the messages sent by clients saving them in a log file named log.txt.

2.4 One Final Suggestion

Note that we are not asking for any documentation of your requirements-gathering and design for this coursework, but you may find that doing these properly still helps you perform the tasks, above, regardless of whether anyone looks at the output. You may also want to show these requirements and/or design documents to the tutors in lab in the early stages of development, just to get feedback about whether you are on the right track.

3 Submission

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

CW1-<bucs username>.zip
e.g. CW1-abc123.zip

The zip file must contain a project folder, titled *Project*, containing your source code and any resources files needed. You should not include packages or other subfolders. No compiled code nor non-needed files, i.e. version control files, should be included.

4 Marking Scheme

Criteria	Max Score	Description
Core Specifications		
Network	20	Server allows at least one client to be connected.
Concurrency	20	Server allows more than one client to be connected and uses thread safety techniques.
Collision Avoidance	10	Players and bots avoids colliding with each other and with walls.
Commenting, formatting and clarity	25	Code uses clean code practices, object-oriented programming techniques, and is consistently commented.
Chat system	10	Server can receive and broadcast messages from one client to all other clients. Protocol implemented correctly.
Advance Features		
Advance chat system features	10	Clients can send message to named clients using an identification system.
Chat Log	5	Server implements a history log of the messages sent by clients saving them to a log file named log.txt.