CM500288: Principles of Programming
Coursework 2: Java Connect4

1 Introduction

This document provides the specification for the second Principles of Programming coursework: **Connect4**.

You can use any environment for the development of your code, but we must be able to compile and run your code using Java 11 without requiring the installation of additional libraries, modules or other programs.

2 Learning Objectives

At the end of this coursework you will be able to

- Plan, organise and implement program code to support reuse and maintainability of a software project.
- Provide a critical review of a software artefact in terms of software quality, design, reuse and robustness, and offer solutions to correct issues encountered.

3 Connect4

For your first coursework, we asked you to investigate and replicate a program that we supplied (SRPN). In this second piece of coursework you will need to analyse, design and improve on a piece of code that you will find on Engage (Connect4).

The challenges in this coursework arise from the fact that the Java code supplied should allow a human to play the game Connect4 against a computer. Connect4 is described at `https://en.wikipedia.org/wiki/Connect_Four`. Please ignore the "Rule variations" listed on Wikipedia. You should only implement the standard version of the game.

The game Connect4 is a game that can be played between a human and a computer player, on a 6x7 grid. The goal is to connect 4 tokens either diagonally, horizontally or vertically.

You have a program to work with. Unfortunately, our code

- Doesn't compile or run.
- Is poorly designed (e.g. is largely made up of a huge main function without comments). With everything we've learned about Object Oriented (OO) programming, we will need to redesign the code to adhere to these principles.
- Needs extension to meet an updated specification (see below).

To resolve these problems, and therefore to gain marks on this coursework, you will need to satisfy Requirements 1-4 below.

4 Requirements

1. Provide a bug and omission list (1-2 pages of single-spaced, 10 point font) explaining why our version of the code doesn't work (in the case of bugs) and/or doesn't provide the functionality needed to play Connect4. Each bug/omission on the list should be described in the following format (we are strict on this format, do not change it and do not submit in another format such as a table):

   - Class : Line(s) : Bug/Omission : Type if Bug (syntax/run time/logic)
   - Solution.

   For example:

   - Board.java : Line 7 : char entered where int expected : runtime error
   - Solution: Guard against non integer inputs.

   It is likely an omission will not have a line number, e.g. does the game actually do what it claims to? Report it as above but without the line number in that case.

   Written work must be submitted as a PDF.

2. Write a report (2-3 pages of single-spaced, 10 point font) describing the ways in which our code could be restructured to better reflect the fundamental OO concepts of modularisation and encapsulation. Include a discussion in that

report of the extent to which you think that inheritance, abstract classes and interfaces might be used to improve our code. You may argue either for or against their inclusion.

In this report you must briefly, in your own words, explain what each of these object oriented principles are. Give an example for each, using the Connect4 game, indicating what you'd need to change to adhere to these principles.

3. Submit a revised version of our code which compiles and runs to provide a working version of the game Connect4. More specifically, provide an altered but uncompiled reworking of our code that allows 1 human player to play a complete game of Connect4 against 1 computer.

   Your code will ideally follow the object oriented principles you described in Requirement 2. You must not rely on external libraries, please complete this assignment with the use of standard libraries only please. If you are using an IDE (not replit) please do not submit the entire directory structure of your project: all .java files should be in the same folder.

   To get started on this assignment, you may wish (but are not obliged) to start by commenting out large parts of our code and altering our printBoard() method. This approach has the advantage of giving you a relatively manageable starting point in the debugging process.

   You may also wish to tackle our placeCounter() method next.

4. Write a further short "report" (approx. 1 page) to describe how the code could be altered to allow a human player to play a game of ConnectN against two computer players (3 players in total). Be specific, what in your code would need to be changed?

   ConnectN is a game identical to Connect4 except that:

   - The winning condition is that N counters of the same colour are placed in a line and
   - One human player competes against two computer players.

   N will be passed to the code as a **command line argument** and $2 < N < 7$. You may adopt an approach that keeps/replaces as much of our code as you find appropriate. You may include a single class diagram and up to half a page of high-level pseudo code in this part of your coursework. Note, we say "report" because it is possible this may be no more than a few paragraphs, depending on how you designed your original Connect4 game.

   Written work must be submitted as a PDF.

5 Submission Instructions

Your submission must be a zip file that can be extracted to a directory with the name: PoP-Connect4. The directory should contain:

- a bug/omission list,
- the two reports,
- a sub-directory (folder) containing the Java files required to run Connect4.

Your .zip file must be submitted to the Engage assignment page by the submission deadline shown – leave adequate time for reading through your reports and checking that your code compiles and runs. Submissions received after this deadline will be capped at 40% if received within 5 working days. Any submissions received after 5 working days will marked at 0%. If you have a valid reason for an extension, you must submit an extension request by following the appropriate process outlined on this page `https://engage.bath.ac.uk/learn/mod/page/view.php?id=33003` – unit leaders cannot grant extensions.

You should leave yourself time to download your file from Engage, extract it, and check that you have attached the correct file, with the content that you want to be marked. **You** are responsible for checking that you are submitting the correct material to the correct assignment.

Please check Engage for the submission deadline.

6 Assessment

6.1 Conditions

The coursework will be conducted individually. Attention is drawn to the University rules on plagiarism. While software reuse (with correct referencing of the source) is permitted, we will only be able to assess your contribution.

6.2 Marks Table

Below is a breakdown of marks, with descriptions on how each criteria is met.

Criteria	Max Score	Description
Bug and Omissions List	max 25	Bug and omissions should be reported in the given format, stating what type of bug it is or if it is an omission. Sensible solutions must be included for each.
Report on OO concepts in relation to restructuring our code	max 25	The report should include a description of each OO concept, with examples of how to implement each in code, referring to the Connect4 code and an explanation of how the code could be restructured to adhere to these concepts.
Revised (working) Connect4 code implementing OO concepts	max 40	Marks will be awarded for functionality, code quality, appropriate use of object oriented programming principles and comments used to document methods. For quality and OO, code should make proper use of coding techniques, such as data encapsulation and generalisation, use of methods and return statements, classes, and robustness. Appropriate variable and method names. Code is well-structured and indented.
Report on extending the code to support ConnectN 3-player	max 10	The report may be brief but should describe both of the additions, what specifically will be changed in your code. The report should acknowledge how good OO design makes extension easier.

In cases where it is not clear from the assignment how it should be marked, you may be called on to explain or demonstrate your program. Such cases include but are not limited to suspected plagiarism.