CM10277: Principles of Programming I
Large Coursework 2: Java Connect4

Dr. Rachid Hourizi & Dr. Michael Wright
Dept. of Computer Science
University of Bath

1 Introduction

The coursework of this unit consists of three parts: the lab sheets and two larger courseworks.

This document provides the specification for the second larger coursework: **Connect4**.

Questions regarding the coursework can always be posted on the Moodle Forum and programming1@lists.bath.ac.uk mailing list.

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 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 piece of code that we supplied (SRPN). In this second piece of coursework you will need to analyse, design and extend a second program that you will find on Moodle (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

Unfortunately, our code

- doesn’t compile or run,
- is poorly designed (e.g. is largely made up of a huge main function without comments) and
- needs extension to meet an updated specification (see below).

To resolve these problems and, by extension, gain marks on this coursework, you will need to satisfy Requirements 1-5 below.

4 Requirements

1. Provide a bug and ommission 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 doesnt provide the functionality needed to play Connect4. Each bug/ommission on the list should be described in the following format:

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

   e.g.

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

2. Write a report (1-2 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.

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 a human marker can use to play a complete game of Connect4 against the computer. 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 report (1-2 pages of single-spaced, 10 point font) describing a program that allows a human player to play 3-handed ConnectN against **two** computer players i.e. a game identical to Connect4 other than that

    - the winning condition is that N counters of the same colour are placed "in a row" 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.

5. Submit an updated and extended version of our code which compiles and runs to provide a working version of the game 3-handed ConnectN (as described in 4., above). More specifically, provide an ammended but uncompiled re-working of our code that a human marker can use to play a complete game of ConnectN against the computer.

5 Assessment

50% of the marks on CM50258 can be gained via coursework. The other 50% can be gained in the exam.

The work described in this document can be used to gain up to 20% of the course total (i.e. max 20/50 of the coursework mark). The remainder of the coursework marks can be gained via lab sheets (10% max) and Large Coursework 1 (20% max).

You may not need to satisfy all parts of all 5 requirements, above, to pass the coursework portion of CM50258. Strategically, you may chose to concentrate on some rather than all of the requirements above. If you do so, we would suggest that you start by satisfying the 'easier' requirements (1, 2 and 3). If you do so, however, be aware that this does not guarantee you a pass mark in the coursework portion of this course. If you have questions about this, or anything else in this document, please email Rachid or Michael.

The coursework will be conducted individually. Attention is drawn to the University rules on plagiarism. Whilst reference to exisitng code (with appropriate citation of that sourse) is permitted, we will only be able to assess your contribution.

You may be called on to explain or demonstrate your program. Cases leading to such a an explanation and/or demonstration include but are not limited to suspected plagiarism.

6 Mark Scheme

Marks for this coursework (Connect4) are available in the following areas:

1. Bug/Omission list: Max 20 marks.
2. Report on restructuring our code to reflect OO concepts: Max 20 marks.
3. Revised (i.e. working) Connect4 code: Max 20 marks.
4. Report on extending our code to support ConnectN: Max 20 marks.
5. Revised and extended ConnectN code: Max 20 marks i.e.
    - 10 marks for functionality and
    - 10 marks for improved reflection of OO concepts.

Total: Max 100 marks.

For the code that you submit, marks will be awarded for functionality, code quality, appropriate use of object oriented programming principles and commenting.

7 Submission Instructions

The deadline for this part of coursework is **5pm 16th December 2016**. Before the deadline upload your solution via Moodle.

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

- a bug/omission list,
- two reports,
- two sub-directories containing the java files required to run Connect4 and 3-handed ConnectN respectively.

Before you upload your solution, make sure that the zip file contains all necessary files and sub-directories. Please download the file from Moodle and double check that you have everything that you want to be marked. You are responsible for checking that you are submitting the correct material to the correct assignment.

8 Feedback

Individual detailed feedback will be provided via Moodle within three weeks of the submission deadline. Note that marks will be moderated, this means that they become only final after the three weeks have expired. Clarification of the feedback can be obtained from the tutors during the labs.