DM 537 Object-Oriented Programming

Fall 2013 Project (Part 1)

Department of Mathematics and Computer Science
University of Southern Denmark

November 11, 2013

Introduction
The purpose of the project for DM537 is to try in practice the use of programming techniques and knowledge about the programming language Java on small but interesting examples.

The project consists of two parts.

Please make sure to read this entire note before starting your work on this part of the project. Pay close attention to the sections on deadlines, deliverables, and exam rules.

Exam Rules
This first part of the project is a part of the final exam. Both parts of the project have to be passed to pass the course.

Thus, the project must be done individually, and no cooperation is allowed beyond what is explicitly stated in this document.

Deliverables
A short project report (at least 4 pages without appendix) has to be delivered. This report has to contain the following 7 sections:

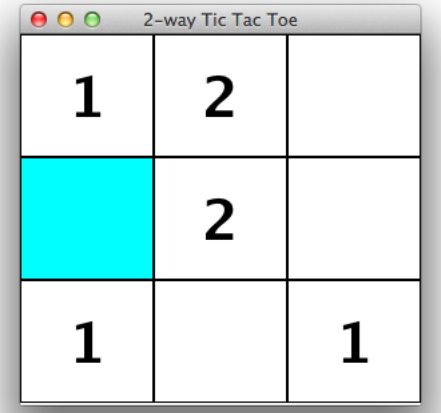
- **front page** (course number, name, section, date of birth)
- **specification** (what the program is supposed to do)
- **design** (how the program is structured)
- **implementation** (how the missing parts were implemented)
- **testing** (what tests you performed)
- **conclusion** (how satisfying the result is)
- **appendix** (complete source code)

The report has to be delivered as a single PDF file electronically using Blackboard's SDU Assignment functionality.

Deadline

Friday, December 6, 23:59

Project "Board Games: Tic Tac Toe & Co"



Tic-Tac-Toe is a simple board game traditionally played by 2 players on a 3×3 grid. The players alternate in placing a mark on one of unmarked fields. A player wins as soon as any of the rows, columns, or diagonals contain 3 of his or her marks in a row. The game is a draw, if no player wins and all 9 fields have been marked.

Here, we consider the extension to n-way Tic-Tac-Toe where n players play on a $(n+1) \times (n+1)$ grid. The winning condition is the same, i.e., the first player to put 3 marks in a row, column, or diagonal wins.

For more information, on the standard variant, consider the Wikipedia entry: `http://en.wikipedia.org/wiki/Tic-tac-toe`

Task 0 – Preparation

On the course home page, you find a directory with a number of Java classes (`CLI.java`, `GUI.java`, `Game.java`, `TTTGame.java`, `TicTacToe.java`, and `UserInterface.java`) as well as two classes (`Coordinate.java` and `TTTBoard.java`) that are incomplete.

Your task is to download these files and put them into a directory for your project. Read all files and look up unknown classes and concepts in the Java API documentation (`http://download.oracle.com/javase/6/docs/api/`) and in the course book. Make sure you understand what is going on here!

Task 1 – Bounding and Shifting Coordinates

Take a closer look at the missing methods in `Coordinate.java`:

- `checkBoundaries` needs to check whether the position represented by this object of class `Coordinate` is on a board of width `xSize` and height `ySize`, where the top-left corner has the position (0,0) and the bottom-right corner has the position (`xSize`-1, `ySize`-1). In the case of a 3×3 grid, the valid positions are (0,0), (1,0), (2,0), (0,1), (1,1), (2,1), (0,2), (1,2), and (2,2). Here, for example, the top-right corner is (2,0).

- The method `shift` needs to construct a new object of class `Coordinate` with coordinates shifted `dx` to the right and `dy` down. For example, calling `pos.shift(-1,1)` with `pos` representing the position (1,1) would result in a new position (0,2).

Task 2 – Implementing The Board
Your next task is to implement the missing methods in `TTTBoard.java`:

- `isFree` needs to return true if, and only if, the given position is free, i.e., if a value of 0 is saved in the appropriate cell of the array representing the board.
- `getPlayer` needs to return the number of the player that made a move at the given position or 0, if the position is free.
- `addMove` needs to first check that the position is actually on the board (Hint: use `checkBoundaries`). If not, an `IllegalArgumentException` should be thrown. If the position is valid, it should be marked by the given player number.
- `checkFull` needs to return true if, and only if, there are no more free positions on the board.
- `checkWinning` needs to return 0 if no player has won yet. Otherwise, it should return the number of the player who has 3 in a row, column, or diagonal. Make sure that your code does not only work for 3×3, but also for quadratic grids of any size!

Task 3 – Testing the Game
There are two user interfaces for the game: a graphical one (`GUI.java`) and a command line-based one (`CLI.java`). Test at least one of these two with your implementations from Tasks 1 and 2 with both the standard 3x3 grid but also with at least one larger board. Document the results of your testing and fix possible errors in `Coordinate.java` and `TTTBoard.java`. It is important to test at least the case of Player 1 winning, the case of Player 2 winning, and the case of the board being full with no player winning. In all cases, you should obtain a message indicating the result from the user interface.

Task 4* – Connect Four

A similar game is Connect Four, where discs are dropped as far down in the selected column as possible. There are two players on a larger board and the goal is to have four of your discs in a row, column, or diagonal. The main difference to Tic Tac Toe with respect to game play is, that when you click on a column, the mark is made in the lowest free position of that column. You find more information about Connect Four on the Wikipedia entry: `http://en.wikipedia.org/wiki/Connect_Four`

Your challenge task is to implement the three new classes `CFGame.java`, `CFBoard.java`, and `ConnectFour.java`, which instead of Tic Tac Toe implement Connect Four. Start with a copy of `TTTGame.java`, `TTTBoard.java`, and `TicTacToe.java`, respectively.

Note that this task is optional and does not have to be solved for this part of the project to be considered as passed.

Task 5* – Go

A much more complicated board game is Go, which is typically played on a 19×19 grid. It is very rich in strategic challenges and can be played for many hours. You find more information about Go on the Wikipedia entry: `http://en.wikipedia.org/wiki/Go_%28game%29`

Your challenge task is to implement Go by three new classes `GoGame.java`, `GoBoard.java`, and `Go.java`.

Note that this task is optional and does not have to be solved for this part of the project to be considered as passed.