COMP 20050: Software Engineering Project 2

2018 / 2019

Assignments

Dr Chris Bleakley

School of Computer Science and Informatics,

University College Dublin,

Belfield, Dublin 4.

Introduction

There are five assignments. Assignments should be submitted using Moodle via the submission links. The assignment deadlines are provided in Moodle. Roughly speaking, one assignment is to be submitted every two weeks of term.

Due to the nature of the course, submissions can only be a maximum of 1 DAY late. There is a 10% deduction for late submissions. Submissions later than 1 day overdue will NOT be accepted. Moodle will block late submission.

In the last lab session of term, we hold of competition in which the bots play each other.

A record of attendance is taken during lab times.

Assignment marks will be available in Moodle.

For all assignments:

- The team lead should submit Java source files (.java), a Java executable file (.jar), a readme file describing how to run the code (.txt) and any required documentation files (.pdf). All files should be submitted together in a single .zip file.
- Make sure that the team names and student numbers are included as comments in the header of all source code and the documentation.
- Submission is via Moodle. Moodle requires a single file submission for the GROUP that should be a .zip of all individual files. Use your team name and assignment number as the file name, e.g. speedy_a1.zip
- In addition, every student should INDIVIDUALLY submit their learning journal on Moodle.

The marking scheme is explained the slides "0. Overview" on Moodle.

PLEASE NOTE DOCUMENTATION MUST BE IN PDF FORMAT. THE TEAM SUBMISSION MUST BE IN ZIP FORMAT. OTHER FORMATS OR CORRUPT FILES WILL NOT BE MARKED.

PLEASE NOTE ALL CODE SUBMITTED MUST BE DEVELOPED FROM SCRATCH BY THE TEAM.

USE OF OPEN SOURCE CODE WILL BE TREATED AS PLAGIARISM.

YOU MUST USE THE GITHUB REPO ALLOCATED TO YOUR TEAM WITHIN THE COMP20050 ORGANISATION. (If you create a repo in GitHub yourself, the repo will be public, not private)

NOTE: Programs are marked against the functionality described in this document. It is ok to include extra functionality to your game in addition to the items listed in the pages below. However, any extra functionality should be in addition to the functionality listed below, not instead of. For example, you might choose to add a feature whereby the player can click icons using the mouse. That's ok but you should also allow the user to type in the commands as specified below. The program should always display game events on the command panel when a button are clicked - the same as if keyboard entry was used.

Sprint 1: Board and Checkers

Enter your team selection on Moodle.

Set up a GitHub account and submit your GitHub ID on Moodle.

If you have never played, play some Backgammon with your team!!!

If you haven't done it already, download and install Java and Eclipse. Read the tutorials for Eclipse.

IntelliJ IDEA can be used instead of Eclipse. Personally, I prefer IntelliJ.

As a team, develop an app release that has the following features (Product Backlog subset):

A UI that consists of 3 panels:

- A board panel that displays the board. You can download a board image or draw your own or paint your own. The board image should include the number of the pips. These numbers can be fixed and can be from either players point of view. Although they do not need to be included until Sprint 4, allow room on the panel for a doubling cube and a match score.
- An information panel that allows display of text message telling the players on what is happening, prompts them with what to do and shows previous user inputs.
- A command panel that allows the user to enter text commands.

The app should:

- Display the initial board with all of the players' checkers on the board in the proper places.
- As a test, move one checker around the board from bar to bear-off one pip at a time. Do this for both players. The display might be a little odd when there are checkers from both players on a single point. That's ok because it can't happen in the real game.
- As a test, echo whatever the user types on the command panel to the information panel. When the user enters "quit", the program should terminate.

No game functionality is needed at this stage.

In the module, we support Swing. You can you an alternate Java GUI library but it will not be supported by in the lectures or lab.

Use GitHub for version control.

BEFORE SUBMITTING ON MOODLE, CHECK THAT YOU HAVE BEEN ALLOCATED TO THE CORRECT GROUP. IF NOT, DO NOT SUBMIT. EMAIL THE LECTURER OR TA TO FIX THIS BEFORE SUBMITTING.

As a group, submit the Java source code and Java executable jar file.

Submit individual Learning Journals for this assignment.

Sprint 2: Moving Checkers

As a team, develop an app based on the previous release that adds the following features (Product Backlog subset). The app should

- Announce the game.
- Get the names of the players and tell each player what colour checkers they have.
- Roll to see who moves first and use these dice as the starting value. Roll again if the dice values are equal.
- Allow the players to take turns to roll and to move around the board. The rolls should be automatic. Moves should be entered as "<starting pip> <ending pip>", e.g. "6 3". The program should return an error if there is no checker at the starting pip, or if the pip numbers are out of range, or if the format of the command is invalid. In the event of an error, the users should be allowed to enter the move again. Only one move is entered at a time. The moves do not have to match the dice roll or avoid opponents blocks yet. The program does not need to check that the move is valid yet.
- When the user enters "next", the current user's turn is over and the other user should then roll and make their moves.
- The pip numbers at the edge of the board should be correct for the player who is moving their checkers. For example, pip 24 should be the furthest from the bear off for the current player. Thus, the pip numbers have to change depending on whose move it is.
- When the user enters "quit", the program should terminate.

Use GitHub for version control.

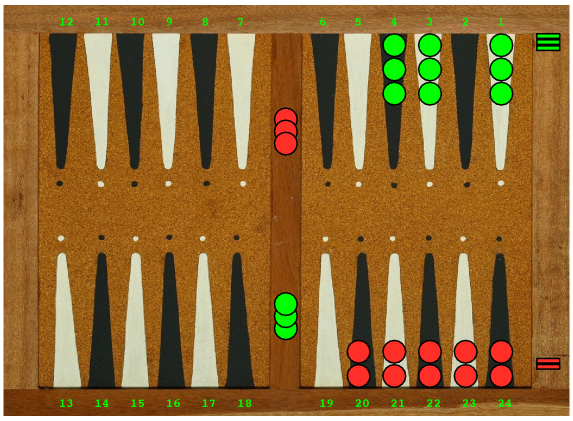
Submit the Java source code and Java executable jar file.

Submit individual Learning Journals for this assignment.

Sprint 3: Game Play

As a team, develop an app based on the previous release that adds the following features (Product Backlog subset):

- After a player rolls the dice, the app should list the legal plays available to the current player as a menu. The list of legal plays should allow hits, bear offs and doubles. Plays should be shown in the following form '24-20 13-11'. An asterisk should be used to indicate a hit, e.g. '6-5* 8-5'. The words 'Off' and 'Bar' should be used to indicate the bear and the bar, e.g. 'Bar-19 19-13 13-7 7-1' and '2-Off 1-Off'. The list should not repeat duplicate moves. Duplicate plays are plays which lead to the same board position. For example, '24-23 23-21' and '24-22 22-21' are duplicate plays. However, '24-23 23-21' and '24-22* 22-21' are not duplicate plays because of the hit.
- If no play is possible, the app should inform the user and automatically continue on to the next roll. If the play is forced (i.e. only one legal play), then the app should inform the user and automatically continue on to the next roll. The game should wait for a second to let the players read the messages.
- The app should allow the current player to select which legal play to make by typing in a letter corresponding to the play, e.g. "A" for the first legal move, "B" for the second, "AA" for the 27th, "AB" for the 28th, and so on. If the user response is invalid, an error message should be displayed and the question should be asked again. Uppercase and lowercase text entry should be allowed. Leading and trailing spaces should be allowed.
- Display all plays on the board.
- Allow the user to enter the command "cheat". This should cause the checkers to move to the following position. The players should be able to play as normally from this position. This command is added for the purposes of testing.



- When the user enters "quit", the program should terminate.
- When the game is over, announce the winning player.

[HINT: ASSIGNMENT 3 IS MORE DIFFICULT THAN IT SOUNDS. PLEASE ALLOW SUFFICIENT TIME.]

Animating the moves is not needed for marking.

Use GitHub for version control.

Submit the Java source code and Java executable jar file.

Submit individual Learning Journals for this assignment.

Sprint 4: Match Play

As a team, develop an app based on the previous release that adds the following features (Product Backlog subset):

- At the beginning of the match, ask the players how many points they want to play to.
- Display the doubling cube, the match score and the match length on the board panel.
- Allow players to enter ‘double’ as a command before they enter a play. If doubling is not legal at that time, display an error message.
- Allow the player receiving the cube to accept or reject the double. If the double is accepted, update the doubling cube showing which player owns the cube. If the double is rejected, end the game and allocate the points to the winner.
- When the game is over, update the match score on the board and report the new match score on the information panel. Ask the players to press any key for starting the next game in the match.
- When the match is over, announce the winner and ask if the players want to play again (yes/no). If the user response is invalid, an error message should be displayed and the question should be asked again. If they choose to play again, the program should ask for the player names and match length again.
- Modify the `cheat’ command so that the board goes to a position where all of the checkers have been bore off except that both players should have two checkers on their own ace points.
- The players should be able to enter `quit’ as a command at any time to quit the program.

Use GitHub for version control.

Submit the Java source code and Java executable jar file.

Submit individual Learning Journals for this assignment.

Sprint 5: Bot

Download Assignment 5 Start from Moodle.

Develop your Bot for use in Competition Week.

- DO NOT change the public API of Bot.
- DO NOT change any of the other classes.
- You can add private methods and/or variables to the Bot class.
- Can the name of the Bot class to match your team name.
- You should enter your team name in the comments at the top of Bot.

Play the bot in one player games to see how good it is and to come up with ideas for improvements.

Note: a Bot that does not play according to the rules of Backgammon or cheats will be disqualified from the competition.

Use GitHub for version control.

Submit the Java source code for the Bot.

Submit a document describing your Bot algorithm.

Submit individual Learning Journals for this assignment.