COMP 20050

Software Engineering Project 2

2016 / 2017

Assignments

Chris Bleakley

School of Computer Science,
University College Dublin,
Belfield, Dublin 4.

Introduction

There are five assignments. Assignments should be submitted using Moodle via the 'Assignment 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, submission 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.

The last week of term is Competition Week. During Competition Week, we will arrange for the bots to play each other during the normal lab session.

The assignments do not have to be done during lab times. However, demonstrators are only available during lab times. 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) 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 THE DOUCMATAION 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.

Sprint 1: Board and Tokens

If you have never played, play some Monopoly!!!

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

Note: Sprints 1-4 are focused on 2 to 6-player Classic Monopoly.

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 Monopoly board. You can download a board image or draw your own.
- 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 textual commands.

The app should:

- Display a blank board.
- Display the player's tokens on the board. The tokens could be as simple as colour dots.
- As a test, move the tokens around the board one square at a time, to make sure that the display is working properly.
- As a test, echo whatever the user types on the command panel to the information panel.

Do NOT include the following yet: buying and selling property, cards, money, houses and hotels, getting out of Jail, penalty. Treat all squares as free parking.

NOTE: check Moodle for source files with constant definitions.

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.

Submit the Java source code and Java executable jar file.

Submit individual Learning Journals for this assignment.

Sprint 2: Money and Property

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

- Get the names of the human players. Allow 2 to 6 players.
- Allocate money to players.
- Roll for the players to see who goes first.
- Allow the players to take turns moving the tokens around the board according to the dice rolls, e.g. each player is prompted and types “roll” and their token is moved. Take into account re-rolls on doubles. Do not implement go to Jail on 3 doubles, yet (see a later Sprint).
- Provide an information message regarding the square that the player has landed on.
- During their turn, the player must roll and move their token at least once (“roll”), more if they roll doubles. The program should allocate more money to players if they pass Go.
- When a player lands on an owned property that has not been mortgaged, they must pay rent before the end of the turn (“pay rent”). Just use a fixed rent for properties for now. Do not include rent multipliers based on dice rolls or colour groups or stations owned (see a later Sprint).
- When a player lands on an unowned property they may buy it put it (“buy”). Auctions should not be included (see a later Sprint).
- The user can query their owned property (“property”).
- The user can query their bank account balance (“balance”).
- The player finishes their turn (“done”). The computer should check that no money is owed before allowing the turn to end.
- The game should detect when the game is over and display the winner.
- If a player finished the game early (“quit”) then the amount of assets they have is calculated and the winner is displayed.
- The user can ask for a list of all valid commands when the user asks (“help”).
- The user should receive appropriate error messages if their input is invalid.

Do NOT include the following yet: bankruptcy, cards, houses and hotels, auctions, selling and mortgaging property, jail, and tax.

In summary, at this stage the game should just allow players to move around the board, to buy properties and to pay rent. Since there is no bankruptcy, players balance can go negative.

In our game, assume that the bank has infinite funds, hotels and houses.

Use GitHub for version control.

Submit the Java source code and Java executable jar file.

Submit individual Learning Journals for this assignment.

Sprint 3: Property

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

- If a player cannot pay their debts (i.e. is not permitted to finish their turn) they must declare bankruptcy (“bankrupt”). This process is simplified compared to the classic rules. The properties are returned to the bank and put back on the market. All building and demolished. The debt is not paid. The player’s turn ends and they leave the game. If there is only one player left in the game, then the winner is declared.
- Include multipliers in the calculation of rents, e.g. colour groups, dice rolls and numbers of stations.
- When a player has all properties in a colour group, they can build a house or hotel (“build <property name> <number of units>”, where a units is a house or hotel). When a player has 4 houses, they can replace them with a house. Enforcing even development is not required.
- A player can sell a house or hotel back to the bank at half price (“demolish <property name> <units>”).
- A player can mortgage any property that they own (“mortgage <property name>”). For property name, you can just use the first word of the name except Old Kent Rd is Kent, Pall Mall is Mall, King’s Cross Station is Kings, and The Angel Islington is Angel.
- The user can choose to redeem one of their own mortgaged properties at any time during their turn (“redeem <property name>”).
- The user should receive appropriate error messages if their input is invalid.
- For simplicity, players can only sell and mortgage property during their own turn. This is different from the board game rules.

Do NOT include the following yet: cards, tax and jail.

In our game, the following are NOT included: payment of debts with a player declares bankruptcy, auctions (i.e. when a player declines to buy) and private sales (i.e. player to player sales).

Use GitHub for version control.

Submit the Java source code and Java executable jar file.

Submit individual Learning Journals for this assignment.

Sprint 4: Cards and Tax

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

- Modifying the code to process turn “done” and dice “roll” so that a player cannot roll again, or complete their turn with a negative bank balance. If they have a negative bank balance, they must sell or mortgage assets to make the balance positive. If they have no assets left, they must declare bankruptcy.
- Simplify the code by removing the “pay rent” instruction. The rent amount should be directly deducted from the player’s balance. Notify the user of the rent payment. This may make their balance negative temporarily.
- When a player lands on a Chance or Community Chest square, automatically draw a card and perform the action on the card. If the player must pay a fine, subtract the amount directly from their bank balance. If they receive money, add the amount to their bank balance. Etc. Notify the user about the actions.
- When a player lands on the Goto Jail square, they should move directly to Jail.
- See the next page for a list of cards.
- When a player lands on Tax, deduct the amount from their bank balance. Notify the user.
- When a player rolls three doubles in a row, they should go to Jail.
- The Get Out Of Jail functionality should be implemented. A player in jail must enter “card”, “roll”: or “pay”.
- The user should receive appropriate error messages if their input is invalid.

The app should now be a complete implementation of 2 to 6-player Classic Monopoly.

Use GitHub for version control.

Submit the Java source code and Java executable jar file.

Submit individual Learning Journals for this assignment.

Community Chest Cards

Advance to Go.

Go back to Old Kent Road.

Go to jail. Move directly to jail. Do not pass Go. Do not collect £200.

Pay hospital £100.

Doctor's fee. Pay £50.

Pay your insurance premium £50.

Bank error in your favour. Collect £200.

Annuity matures. Collect £100.

You inherit £100.

From sale of stock you get £50.

Receive interest on 7% preference shares: £25.

Income tax refund. Collect £20.

You have won second prize in a beauty contest. Collect £10.

It is your birthday. Collect £10 from each player.

Get out of jail free. This card may be kept until needed or sold.

Pay a £10 fine or take a Chance.

Chance Cards

Advance to Go.

Go to jail. Move directly to jail. Do not pass Go. Do not collect £200.

Advance to Pall Mall. If you pass Go collect £200.

Take a trip to Marylebone Station and if you pass Go collect £200.

Advance to Trafalgar Square. If you pass Go collect £200.

Advance to Mayfair.

Go back three spaces.

Make general repairs on all of your houses. For each house pay £25. For each hotel pay £100.

You are assessed for street repairs: £40 per house, £115 per hotel.

Pay school fees of £150.

Drunk in charge fine £20.

Speeding fine £15.

Your building loan matures. Receive £150.

You have won a crossword competition. Collect £100.

Bank pays you dividend of £50.

Get out of jail free. This card may be kept until needed or sold.

Sprint 5: Bots

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.
- 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.

Use GitHub for version control.

Submit a document describing your Bot algorithm.

Submit the Java source code and Java executable jar file.

Submit individual Learning Journals for this assignment.