COMP41670: Software Engineering

2023 / 2024

Group Project

Dr Avishek Nag

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

avishek.nag@ucd.ie

Introduction

There are three Sprints. These Sprints combine to incrementally build into a Java implementation of a board game.

The following steps should be done before starting to code:

- The project will be done in groups of two. Self-select your group at My Class – My Groups on Brightspace. See Brightspace for the deadline.
- Setup Git on your computer.
- Setup your GitHub account.
- Setup a GitHub repo for the project. **THIS REPO MUST BE PRIVATE.**
- Enter your project details via the Group Project Details Quiz on Brightspace. Both group members must do this. See Brightspace for the deadline.
- Give all team members and the Teaching Assistant (user name jordan2doyle1) and the Module Coordinator (user name ChrisBleakley) access to the repo.
- GitHub must be used for source code control for the duration of the project. (See the Grading Scheme below “GitHub not used correctly, up to 2 grade deductions”). Part of the final submission is via GitHub.

Sprint 1: Display & Rolls

Play some Backgammon!!!

The reference rules are at: https://www.bkgm.com/rules.html

As a group, implement and verify a Java program with the following features. Use the console for input and output.

A feature whereby the Backgammon board is displayed with the Checkers in their initial positions.

A feature that prompts the players to enter their names. Their names should be used in later prompts.

A feature that allows the players to take turns rolling the dice. The players should be prompted to enter a command. Entering the “roll” command should cause the result of a two dice roll to displayed on the board panel and reported in the log panel. The dice should be generated randomly.

A feature whereby a “quit” command causes termination of the program.

Submit as per the instructions in the Submission section.

The following feature checklist will be used for grading:

Feature	Marks Available	Comments
Display board	4	
Enter names	2	
Roll command	2	
Quit	2	
TOTAL	12	

Sprint 2: Game

As a group, implement and verify a Java program with the following features.

A feature whereby when the game starts, the program rolls one die for each player to determine which player goes first. The result is used for the first move.

A feature whereby the current player is indicated on the display.

A feature that displays the pip number of every point on the board. The pip numbers should change depending on which player's turn it is.

A feature which lists all legal moves after the dice roll and allows the user to enter a letter code for the desired move. The board should be updated to reflect the move selected. All hits and bear off should be applied. All rules of backgammon should be considered.

A "pip" command which reports the pip count for both players.

A "hint" command which lists all allowed commands. The list should exclude the commands for testing.

A feature whereby the syntax of the commands entered are checked. An appropriate error message is generated if the command is invalid.

A feature that detects when the game is over and announces the winner of a game.

Submit as per the instructions in the Submission section.

The following feature checklist will be used for grading:

Feature	Mark Available	Comments
Roll to start	2	
Current player display	2	
Pip display	2	
All legal moves selection	14	
Pip command	2	
Hint command	2	
Syntax check	2	
Game over	2	
TOTAL	28	

More details on the move selection feature

There are two ways that moves can be selected. We will accept **either option** as being correct.

Option 1: Two step entry

```
13--+---+---+---+---18 BAR  19--+---+---+---+---24  OFF
|   |   |   |   |  |        |   |   |   |   O   O

|   |   |   |   |  |        X   |   |   |   |   X
12--+---+---+---+---07 BAR  06--+---+---+---+---01  OFF
X to play 2-1. Select from:
A) Play 6-4
B) Play 6-5
C) Play 1-off
>> A
13--+---+---+---+---18 BAR  19--+---+---+---+---24  OFF
|   |   |   |   |  |        |   |   |   |   O   O

|   |   |   |   |  |        |   |   X   |   |   X
12--+---+---+---+---07 BAR  06--+---+---+---+---01  OFF
X to play 1. Select from:
A) Play 4-3
B) Play 1-off
>> B
13--+---+---+---+---18 BAR  19--+---+---+---+---24  OFF
|   |   |   |   |  |        |   |   |   |   O   O

|   |   |   |   |  |        |   |   X   |   |   |
12--+---+---+---+---07 BAR  06--+---+---+---+---01  OFF
```

If you implement this option, be careful to deal with the following rules: “A player must use both numbers of a roll if this is legally possible (or all four numbers of a double). When only one number can be played, the player must play that number. Or if either number can be played but not both, the player must play the larger one.” For example, the following is incorrect. The play “C) 5-3” is illegal and should not be listed because, if made, the second die (5) could not be played.

```
13--+---+---+---+---18 BAR  19--+---+---+---+---24  OFF
|   |   |   |   |  |        |   |   |   |   O   O

|   |   |   |   |  |        X   X   |   |   |   O
12--+---+---+---+---07 BAR  06--+---+---+---+---01  OFF
X to play 2-5. Select from:
A) 6-4
B) 5-off
```

Option 2: One step entry

```
13--+---+---+---+---18 BAR  19--+---+---+---+---24  OFF
|   |   |   |   |  |        |   |   |   |   O   O
|   |   |   |   |  |        X   |   |   |   |   X
12--+---+---+---+---07 BAR  06--+---+---+---+---01  OFF
X to play 2-1. Select from:
A) Play 6-4 1-off
B) Play 6-5 5-3
>> A
13--+---+---+---+---18 BAR  19--+---+---+---+---24  OFF
|   |   |   |   |  |        |   |   |   |   O   O
|   |   |   |   |  |        |   |   X   |   |   |
12--+---+---+---+---07 BAR  06--+---+---+---+---01  OFF
```

```
13--+---+---+---+---18 BAR  19--+---+---+---+---24  OFF
|   |   |   |   |  |        |   |   |   |   O   O
|   |   |   |   |  |        X   X   |   |   |   O
12--+---+---+---+---07 BAR  06--+---+---+---+---01  OFF
X to play 2-5. Forced move:
6-4 5-off
```

Sprint 3: Match

As a group, implement and verify the following.

A “dice <int> <int>” command which cause the subsequent dice roll to equal the given numbers.

A “test <filename>” command which performs the commands in the given text file.

A feature that at the start of new match, allows players to enter their names and select the length of the match.

A feature that displays the match score and the match length on the board.

A feature the displays the double with the position showing ownership.

A command “double” that offers a double to the other player. The other player must respond with either “accept” in which case the game continues for twice the stake OR “refuse” in which case the game ends with the doubling player gaining the current stake. Checks must be performed to ensure that the inputs are legal according to the rules of the same.

A feature that announces whether a game ends in a Single, a Gammon or a Backgammon and updates the match score according to the end position and the score.

A feature that detects the end of a match and announces the winner.

A feature that allows the players to start a new match.

The functionality of the “hint” command should be updated.

Submit as per the instructions in the Final Submission section.

The following feature checklist will be used for grading:

Feature	Marks Available	Comments
Dice command	2	
Test command	6	
Names & match length	2	
Display match score	2	
Display cube	2	
Double command	2	
Gave over	2	
Match over	2	
New match	1	
Hint command	1	
TOTAL	22	

Assessment

The following assessments will take place.

Assessment	Due Date
Sprint 1 Design Review	Week 8
Sprint 2 Design Review	Week 10
Final Submission	Week 12

Sprint 1 and 2 are not submitted to Brightspace. They are assessed by Design Review. The Final Submission is submitted to Brightspace.

The Design Reviews will be conducted during the lab sessions. A Demonstrator will meet with your team to review your code. The group will demo the code, show the source code and answer some questions. Progress will be assessed against the features listed, testing, code quality and version control.

The Design Review is graded. Your team must attend the lab session on the given date to participate in the Design Review. If your team does not attend the lab session, the Design Review grading will be excluded from your assessment (i.e. the other grading components become 100% of your grade, rather than 95%).

Final Submission

For the FINAL CODE, submit a zip file named with your group name containing following items:

- A report in PDF format (1 page) including:
    - Group name, group member names and GitHub IDs.
    - A link to your GitHub release (see Brightspace – Learning Materials - About the Tools for more information on creating the GitHub release). Test that the link works after it is added to the report.
    - State the relative amount of work done by each group member for EACH SPRINT (e.g. 50-50). If the workload is not 50-50, please explain why.
    - Your self-assessment checklist that lists for the FINAL CODE whether the features from ALL three Sprints are fully working, partially working, or not working (see example below). Explain partially working features in more detail in the comments.
- A video (max. 5 minutes) in .mp4 format of a screen recording with voice over. The video should:
    - Show the working features
    - Explain how you tested the code
    - Explain the structure of your code
- A directory containing the source code.
- A directory containing an executable JAR file for the program (see Brightspace – Learning Materials - About the Tools for more information on creating the JAR).

Notes

Be careful not to delete the release or repo on GitHub until the end of the semester plus 12 months.

Make sure that the team name, student names and GitHub IDs are included in comments at the top of all source code files.

Use of open-source code will be treated as plagiarism.

Grading Scheme

The marks for the Group Project are allocated as follows:

Assessment	Marks
Sprint 1 Design Review	5%
Sprint 2 Design Review	5%
Final Submission: Functionality	70%
Final Submission: Testing	10%
Final Submission: Code Quality	10%
Group Project TOTAL	100%

Marking is done according to the scoresheets above and below. The numerical value is converted to a grade using the Alternative Linear Conversion Grade Scale 40% Pass.

https://www.ucd.ie/students/exams/gradingandremediation/understandinggrades/

Functionality

Functionality is marked as the percentage of features working correctly. Each feature is scored by its number of points, which indicates the difficulty of implementing the feature.

For example, a project with the following feature checklist would likely be graded as:

Feature	Marks Available	Marks	Comments
Display panels	2	2	Working
Display board	2	2	Working
Enter names	2	2	Working
Roll dice	2	1	Partially working – only 1 dice works of 2
Quit	2	0	Not working
TOTAL	10	7	7/10 = 70% = B-

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 above. However, any extra functionality should be in addition to the functionality listed above, 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.

Test Quality

Testing is marked as follows.

Quality Check	Marks Available	Comments
Do all classes have a JUnit test class?	2	
Do all methods have at least one JUnit test?	4	
Do the JUnit tests provide 80% test coverage.	4	

Do the test scripts provide a complete systems test?	4	
Has the testing approach been explained in the videos?	2	
TOTAL	16	

Code Quality

Code quality is marked according to the quality checklist below. Percentage points are converted to grades as for functionality.

Quality Check	Marks Available	Comments
Is the code readable?	2	
Is the project well-structured into classes?	2	
Does each method perform a single clearly defined function?	2	
Are appropriate data structures used?	2	
Are design patterns used where appropriate?	2	
Is scope minimised?	2	
Is data single source?	2	
Is inheritance used wisely?	2	
Are naming conventions used consistently?	2	
Are constants well used?	2	
Are names clear and meaningful?	2	
Are comments useful?	2	
Is Javadoc used correctly for at least 1 class?	2	
Are class diagrams for the project shown in the video?	2	
Are sequence diagrams for the project shown in the video?	2	
TOTAL	34	

Deductions

If necessary, the following deductions will be applied:

- Based on review of the comments in the report on the relative work done and on review of the group member contributions on GitHub, the grades for individual students in a group may be adjusted. In general, adjustments are only made in the case of large disparities in work done.
- GitHub not used correctly, up to 2 grade deductions
- No report OR no video, 1 grade deduction (e.g. A to B)
- Incomplete report OR incomplete video, 1-3 grade point deduction (e.g. A to A- or B+ or B)
- No GitHub executable jar release or not working, 1 grade point deduction
- No or incomplete source code in Brightspace but available in GitHub, 1 grade deduction

- No or incomplete GitHub source code release but available in Brightspace, 1 grade deduction
- No or incomplete source code submission in Brightspace and in GitHub, graded as NM (no mark)
- Up to 5 working days late, 1 grade point deduction.
- Up to 10 working days late, 2 grade points deduction.