# *Simple Board Game*

**Due Date: 10pm on Friday in Week 7 (18th Sep 2020)**

## Introduction

This assignment is worth **10%** of the marks for your final assessment in this unit. **Heavy penalties will apply for late submission.** This is an individual assignment and must be entirely your own work. You must attribute the source of any part of your code which you have not written yourself. ***Your program will be checked with a code similarity detector***. Please note the section on plagiarism in this document.

**The assignment must be done using the BlueJ environment.**

The Java source code for this assignment must be implemented in accordance to the ***Java Coding Standards*** for this unit.

Any points needing clarification may be discussed with your tutor in the tutorial session.

## Specification

For this assignment you will write a program that plays a rather simplistic ***Board Game***. This section specifies the required functionality of the program. ***Only a simple text interface (using the BlueJ Terminal Window) is required for this program***; however, more marks will be gained for a game that is easy to follow with clear information/error messages to the player.

The aim of your program is to ***simulate*** a simple *Board Game* commonly played among children, where the players take turn to roll a dice, and move their game tokens on a physical board. The winner is the one who reaches a final position on the board first. Your program will display a menu which allows the user of the program to select various options to simulate the various board game operations. As this is not a graphical program, you will show the players' "positions" by simply displaying numbers on the screen (eg. "***`Player David is on position 25`***", etc). The dice rolls will also be simulated by the program, which will update each player's position accordingly.

For this assignment, the program will only handle *TWO* players. It will keep track of the positions of the players until one, or both, reaches the position of 50. If a player reaches certain special positions (11/22/33/44 or 5/15/25/35) he[**] will either receive a penalty or a bonus (see rules in the ***Program Logic*** section below).

** *for the rest of the document, "he" will be taken to mean "he/she".*

## Program Logic

The *Board Game* begins by displaying a menu with the following options :

```
Welcome to the Simple Board Game
===========================
(1) Start/Restart a Game
(2) Play One Round
(3) Display Players' Positions
(4) Display Game Help
(5) Exit Game
Choose an option:
```

**Option (1)** starts a new Game, and asks the user to enter a name for each "player". A player's name must be alphabetic (including spaces) and must not be blank (empty string or a string containing only spaces). The players' positions start from 0. If this option is chosen again after the players have already been set up, 2 "new" players are set up (ie. with 2 new names, and both their starting positions set to 0). Note that the new players replace the previous players – there are only ever two players at any one time.

**Option (2)** simulates a "***dice roll***" operation. When this option is chosen, the computer generates 2 random numbers between 1-6 (ie. simulating a 6-sided dice), one for each player. It then updates both players' positions accordingly. The rules for each "round" are as follows:

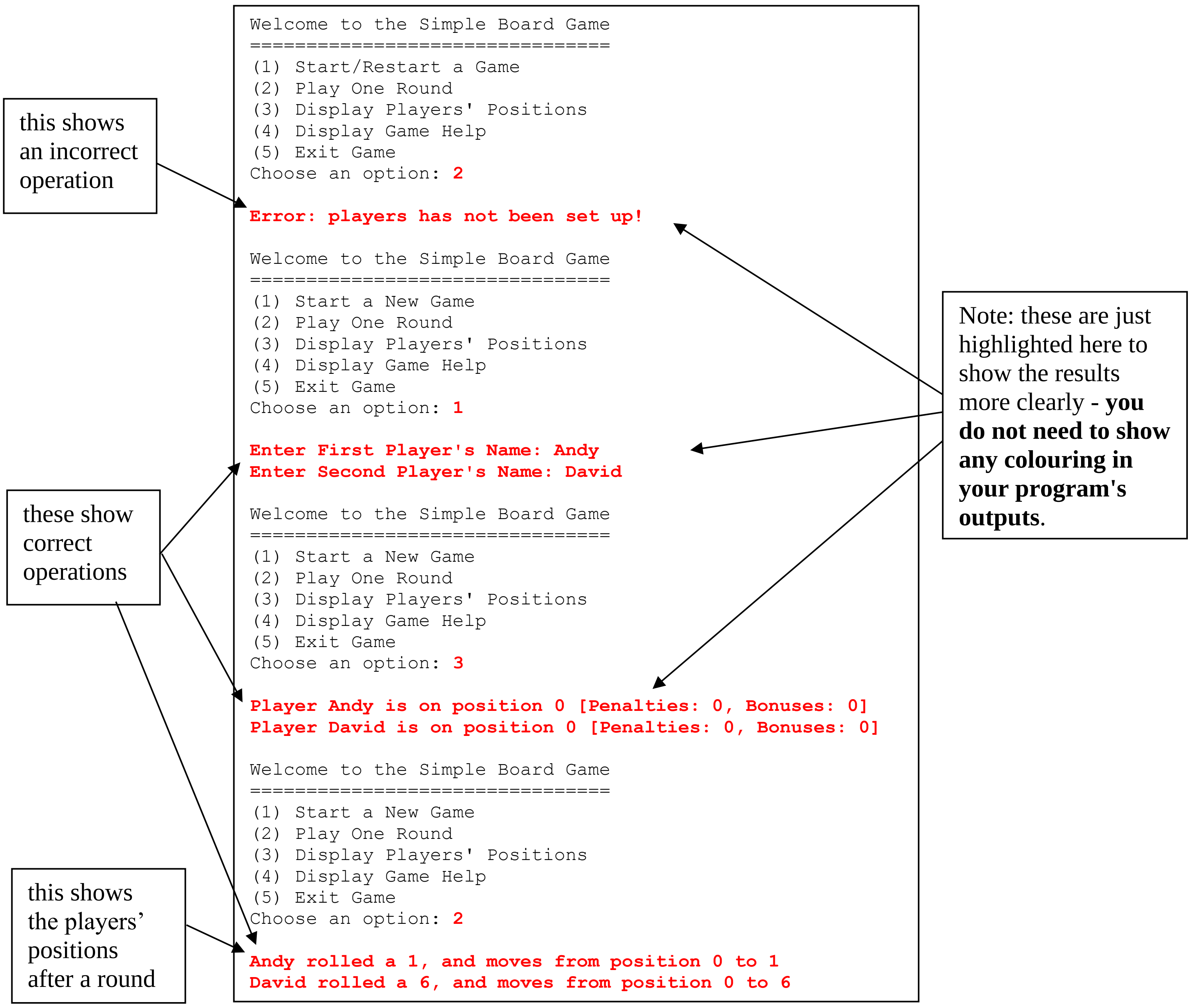
- if both players reach position **50** or more at the same time, the game is a ***Draw***
  - note that both players can reach 50 (or more) at the same time, since for each round, a dice is thrown for each player, ***before*** a winner is decided
- a player is considered a winner if he reaches position 50 or more, and the other player has not reached 50
- if any player reaches one of the special positions **11/22/33/44**, he will be given a penalty, by having 5 subtracted from his current position
- if any player reaches one of the special positions **5/15/25/35**, he will be given a bonus, by having the computer roll the dice a second time for him, and his position moved once more. The bonus move has the special condition that if his new position (after the bonus dice roll) is on 11/22/33/44, no penalty is imposed.
- after each round, the players' positions are printed on the screen (with the winner shown if the game is finished, or when there is a Draw, and if a penalty/bonus was imposed).
- when one player wins (or when there is a Draw), it is regarded as the end of the game. If the user selects **Option (2)** again at this point, the program should simply display the final result and ask him to select **Option (1)** instead (to restart a game).

**Option (3)** displays the current players' positions (both players), and the total number of penalties/bonuses received.

**Option (4)** displays some brief instructions regarding game play.

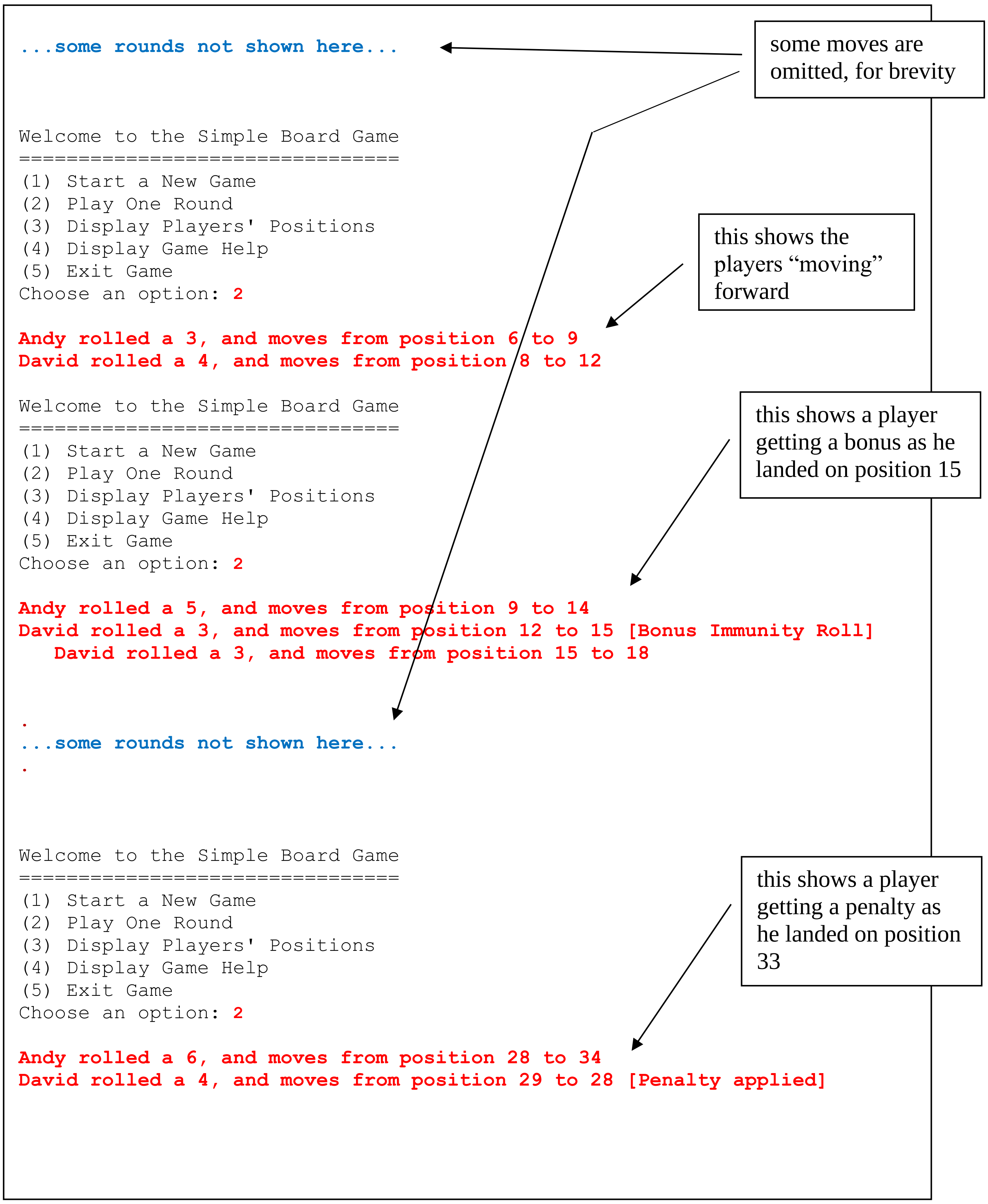
**Option (5)** exits the program.

Sample screenshots of typical inputs/outputs (including invalid inputs) for **Options (1), (2) & (3)**:


this shows an incorrect operation
these show correct operations
this shows the players' positions after a round
Note: these are just highlighted here to show the results more clearly - **you do not need to show any colouring in your program's outputs**.

```
Welcome to the Simple Board Game
================================
 (1) Start/Restart a Game
 (2) Play One Round
 (3) Display Players' Positions
 (4) Display Game Help
 (5) Exit Game
Choose an option: 2

Error: players has not been set up!

Welcome to the Simple Board Game
================================
 (1) Start a New Game
 (2) Play One Round
 (3) Display Players' Positions
 (4) Display Game Help
 (5) Exit Game
Choose an option: 1

Enter First Player's Name: Andy
Enter Second Player's Name: David

Welcome to the Simple Board Game
================================
 (1) Start a New Game
 (2) Play One Round
 (3) Display Players' Positions
 (4) Display Game Help
 (5) Exit Game
Choose an option: 3

Player Andy is on position 0 [Penalties: 0, Bonuses: 0]
Player David is on position 0 [Penalties: 0, Bonuses: 0]

Welcome to the Simple Board Game
================================
 (1) Start a New Game
 (2) Play One Round
 (3) Display Players' Positions
 (4) Display Game Help
 (5) Exit Game
Choose an option: 2

Andy rolled a 1, and moves from position 0 to 1
David rolled a 6, and moves from position 0 to 6
```

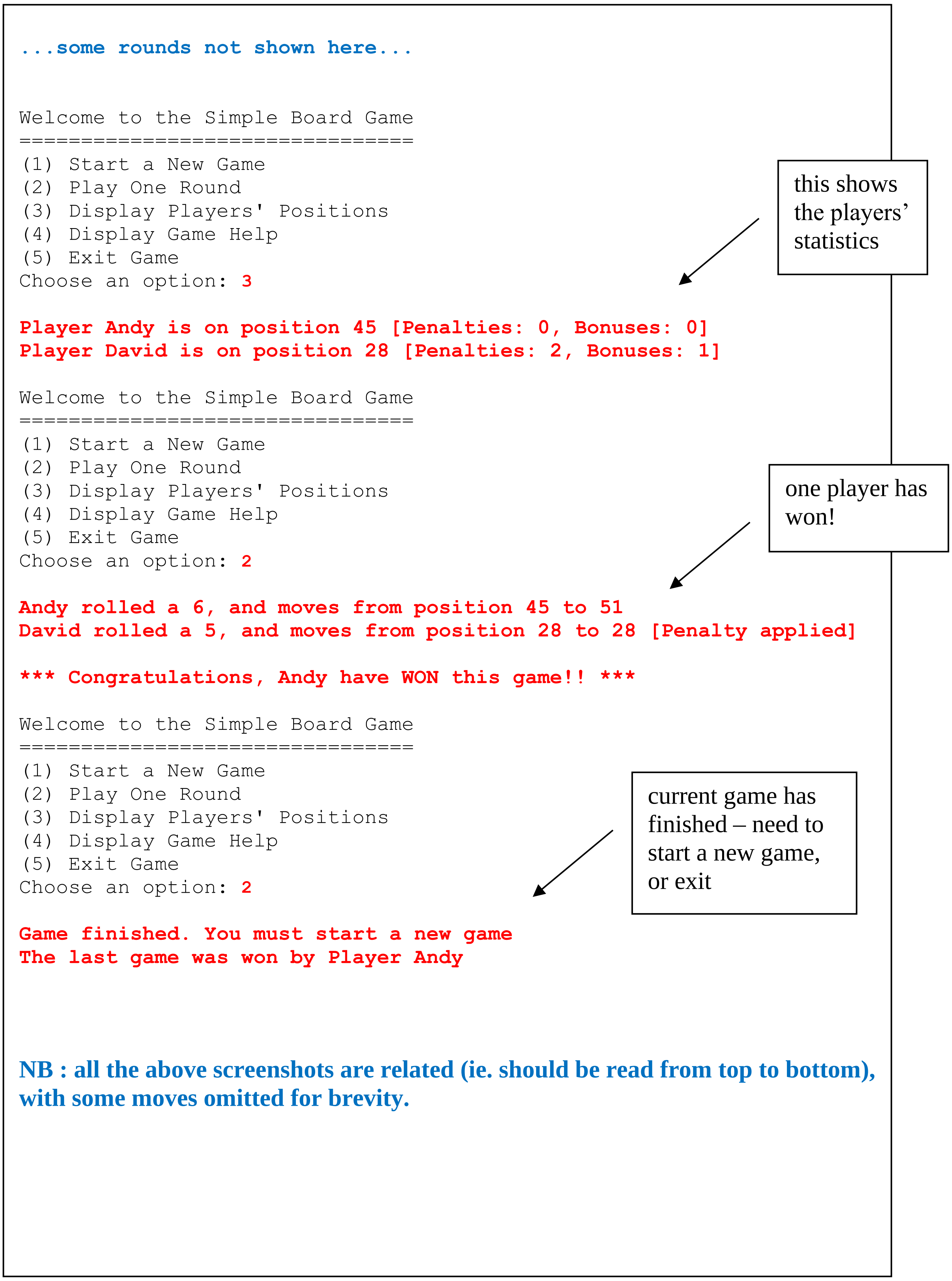
Sample screenshots continued…



```
...some rounds not shown here...


Welcome to the Simple Board Game
================================
(1) Start a New Game
(2) Play One Round
(3) Display Players' Positions
(4) Display Game Help
(5) Exit Game
Choose an option: 2

Andy rolled a 3, and moves from position 6 to 9
David rolled a 4, and moves from position 8 to 12

Welcome to the Simple Board Game
================================
(1) Start a New Game
(2) Play One Round
(3) Display Players' Positions
(4) Display Game Help
(5) Exit Game
Choose an option: 2

Andy rolled a 5, and moves from position 9 to 14
David rolled a 3, and moves from position 12 to 15 [Bonus Immunity Roll]
   David rolled a 3, and moves from position 15 to 18


.
...some rounds not shown here...
.


Welcome to the Simple Board Game
================================
(1) Start a New Game
(2) Play One Round
(3) Display Players' Positions
(4) Display Game Help
(5) Exit Game
Choose an option: 2

Andy rolled a 6, and moves from position 28 to 34
David rolled a 4, and moves from position 29 to 28 [Penalty applied]
```

some moves are omitted, for brevity
this shows the players “moving” forward
this shows a player getting a bonus as he landed on position 15
this shows a player getting a penalty as he landed on position 33

Sample screenshots continued…



```
...some rounds not shown here...


Welcome to the Simple Board Game
=================================
(1) Start a New Game
(2) Play One Round
(3) Display Players' Positions
(4) Display Game Help
(5) Exit Game
Choose an option: 3

Player Andy is on position 45 [Penalties: 0, Bonuses: 0]
Player David is on position 28 [Penalties: 2, Bonuses: 1]

Welcome to the Simple Board Game
=================================
(1) Start a New Game
(2) Play One Round
(3) Display Players' Positions
(4) Display Game Help
(5) Exit Game
Choose an option: 2

Andy rolled a 6, and moves from position 45 to 51
David rolled a 5, and moves from position 28 to 28 [Penalty applied]

*** Congratulations, Andy have WON this game!! ***

Welcome to the Simple Board Game
=================================
(1) Start a New Game
(2) Play One Round
(3) Display Players' Positions
(4) Display Game Help
(5) Exit Game
Choose an option: 2

Game finished. You must start a new game
The last game was won by Player Andy
```

this shows the players' statistics
one player has won!
current game has finished – need to start a new game, or exit

**NB : all the above screenshots are related (ie. should be read from top to bottom), with some moves omitted for brevity.**

**Additional Notes :**

***The menu must be displayed repeatedly after each operation*** (as shown in the screenshots), until the user chooses **Option (5)**. Inputs other than 1-5 (including non-numeric characters) should be rejected, and an error message printed.

If the user chooses **Options (2)** or **(3)**, before a player has been set up, an appropriate error message should be printed, with the operation aborted and menu re-displayed.

Your program must deal with invalid values entered by the user in a sensible manner.

For all the options, the inputs/outputs can be formatted in many different ways. Discuss with your tutor regarding how you should implement these in your program.

Assume that the user inputs are always of the correct data types (ie. when an integer is required, only an integer is entered, etc), except for the main menu options.

The sample screenshots above are meant to be examples. Your user interface need not be exactly as shown in those examples. However, you should discuss with your tutor about what to include in your user interface. Keep it simple.

## Program Design

Your program must demonstrate your understanding of the object-oriented concepts and general programming constructs presented in the unit.

The data type of each field in the classes must be chosen carefully in accordance to the requirements of the program described in the preceding sections, and you must be able to justify the choice of the data type of the fields. You may want to include comments in the class to explain any assumption made.

Your code should conform to the ***FIT9131 Java Coding Standards***; any violations will lead to marks being deducted.

Basic validation of values for fields and local variables should also be implemented. You should not allow an object of a class to be set to an invalid state. You should include appropriate constructor(s) for each class, and accessor/mutator methods for all the attributes.

You class design should include at least these classes:

**`Game`**

**`Player`**

**`Dice`**

Discuss with your tutor what attributes/behaviour are suitable for these classes, and how they interact with each other.

You may make the following assumptions:

- a **`Game`** object will be responsible for displaying the menus, accepting user responses, and performing the requested operations. It will make use of 2 **`Player`** objects and 1 **`Dice`** object.
- a **`Player`** object will remember his own name, what board position he is currently on, and how many times he has received a penalty/bonus.
- a **`Dice`** object will be used to generate a ***random*** number between **1-6**. This is used by the program to simulate a dice roll operation.

If your design varies significantly from the above, you must first seek approval from your tutor; otherwise it may not be accepted.

You can include any other appropriate classes in your design. However, it is strongly suggested that you discuss these with your tutor first. You should always start with a simple design first.

## Assessment

Assessment for this assignment will be done via an ***interview*** with your tutor. The marks will be allocated as follows:

- Object-oriented ***design*** quality. This will be assessed on appropriate implementation of classes, fields, constructors, methods and validation of an object's state **30%**
- Adherence to ***FIT9131 Java Coding Standards*** **10%**
- Program ***functionality*** in accordance the requirements **60%**

***You must submit your work by the submission deadline on the due date*** (a late penalty of 20% per day, inclusive of weekends, of the possible marks will apply - up to a maximum of 100%). ***There will be no extensions*** - so start working on it early.

Marks will be deducted for untidy/incomplete submissions, and non-conformances to the *FIT9131 Java Coding Standards*.

## Interview

This is where most of your marks will come from.

You will be asked to demonstrate your program at an "***interview***" following the submission date. At the interview, you will be asked to explain your code, your design, discuss design decisions and alternatives, and modify your code as required. ***Marks will not be awarded for any section of code or functionality that you cannot explain satisfactorily*** (your tutor may also delete excessive in-code comments before you are asked to explain that code).

In other words, ***you will be assessed on your understanding of the code***, and not solely on the actual code itself.

Interview times will be arranged in the tutorial sessions in *Week 7*. It is your responsibility to attend the tutorial and arrange an interview time with your tutor.

The actual interview will take place during the week **after** week 7, via Zoom or other video facilities. ***You must have audio and video capabilities on your computer that can be used for the interview.***

***Any student who does not attend an interview will receive a mark of 0 for the assignment.***

## Submission Requirements

The assignment must be uploaded to ***Moodle*** on or before the due date (as listed at the start of this document). The link to upload the assignment will be made available, along with a check-list of things to submit, in the ***Assessments*** section of the unit's Moodle site before the submission deadline. The submission requirements are as follows:

A **.zip** file uploaded to Moodle containing the following components:

- the **BlueJ** project you created to implement your assignment. The .zip file should be named with your Student ID Number. For example, if your id is **12345678**, then the file should be named **12345678_A1.zip**. Do not name your zip file with any other name.
    - it is your responsibility to check that your ZIP file contains all the correct files, and is not corrupted, before you submit it. If your tutor cannot open your zip file, or if it does not contain the correct files, you will not be assessed.
- a completed **Assignment Cover Sheet**. This will be available for download from the unit's Moodle site before the submission deadline. You simply complete the editable sections of the document, save it, and include it in your .zip file for submission.
    - alternatively, the assignment may allow you to accept an online submission statement in place of the physical cover sheet

Marks will be deducted for failure to comply with any of these requirements.

Warning: **there will be no extensions to the due date**. Any late submission will incur the 20% per day penalty. It is strongly suggested that you submit the assignment well before the deadline, in case there are some unexpected complications on the day (eg. interruptions to your home internet connection).

## Plagiarism

Cheating and plagiarism are viewed as serious offences. In cases where cheating has been confirmed, students have been severely penalised, from losing all marks for an assignment, to facing disciplinary action at the Faculty level. Monash has several policies in relation to these offences and it is your responsibility to acquaint yourself with these.

Academic integrity, plagiarism and collusion :
(https://www.monash.edu/students/academic/policies/academic-integrity)