Semester 2 2021 Assignment 1

Number Guessing Game

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 with the Java Coding Standards for this unit.

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

Specification

For this assignment you will write a program that plays a rather simplistic ***Number Guessing Game***. This section specifies the required functionality of the program. ***Only a text interface 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 for a player to guess a number between 1-50, which will then be compared to a number (also between 1-50) randomly generated by the computer. For each "round" of the game, the player will have 3 chances to guess the number. Correct guesses will win some money, while incorrect guesses will attract some penalties.

When the player guesses a number, his win/loss for that round is to be remembered by the program. At any point, the program can display how many wins/losses had occurred so far, how much credit is left, how much has been won/lost so far, and the percentage of wins.

For this assignment, the program will only handle *ONE* player at any one time (for each round). However, the player can be changed while the program is executing. Every time a player is "changed", all his data will be reset for the new round.

See the ***Program Logic*** section below for more details of the program's features.

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

Program Logic

The *Number Guessing Game* begins with a welcome message followed by a menu with the following options :

```
Welcome to the Number Guessing Game
==============================
(1)      Set Up New Player
(2)      Add Credit
(3)      Play One Round
(4)      Display Player Statistics
(5)      Display Game Help (6) Exit Game
Choose an option:
```

Option (1) asks the user to enter a name for the "player". The player's name must not be blank. If this option is chosen again after a player has already been set up, a "new" player is set up (ie. with a new name, $0 credit, 0 wins, 0 losses, $0 winnings). Note that each "new" player replaces the previous "old" player – there is only ever one player at any one time.

Option (2) asks the user to input a number between 1-20. This represent an amount to be added to the player's credit; the credit will then be used to play the game. More credit can be added at any time during the game.

Option (3) asks the user to guess a number between 1-50. The computer then randomly generates a "lucky number", also between 1-50, and then checks if the outcome matches the number guessed. The player has 3 chances to guess the correct number. If he guessed it correctly, he wins some money, otherwise he is asked to guess again, up to a maximum of 3 times for ***each round***. The rules are:

1st guess correct	wins $15 (added to credit balance)
2nd guess correct	wins $10 (added to credit balance)
3rd guess correct	wins $5 (added to credit balance)
3rd guess is wrong, but, within ±5 of the lucky number	wins a random consolation prize of $1-5 (picked by computer) (added to credit balance)
All 3 guesses wrong	loses $5 (deducted from credit balance)

After each guess, the program should also inform the user if his next guess should be higher or lower (to help him make a correct guess) and how many guesses are left.

The amount won/lost is added/deducted to the player's credit balance (the minimum credit balance is $0). The number of wins and losses, plus the amount won/lost, are recorded by the program after each round (up to 3 guesses in a round). This data is to be used for calculating the statistics for **Option (4)** later.

At the end of each round, the program ***must*** also display the user's final guess, and the actual lucky number generated by the computer.

If the user enters a number which is less than 1, or more than 50, it should be rejected, and the *outcome counted as one wrong guess*.

Option (4) displays some statistics about the current player's wins/losses, his overall winning percentage, his current credit balance, and his total winnings/losses.

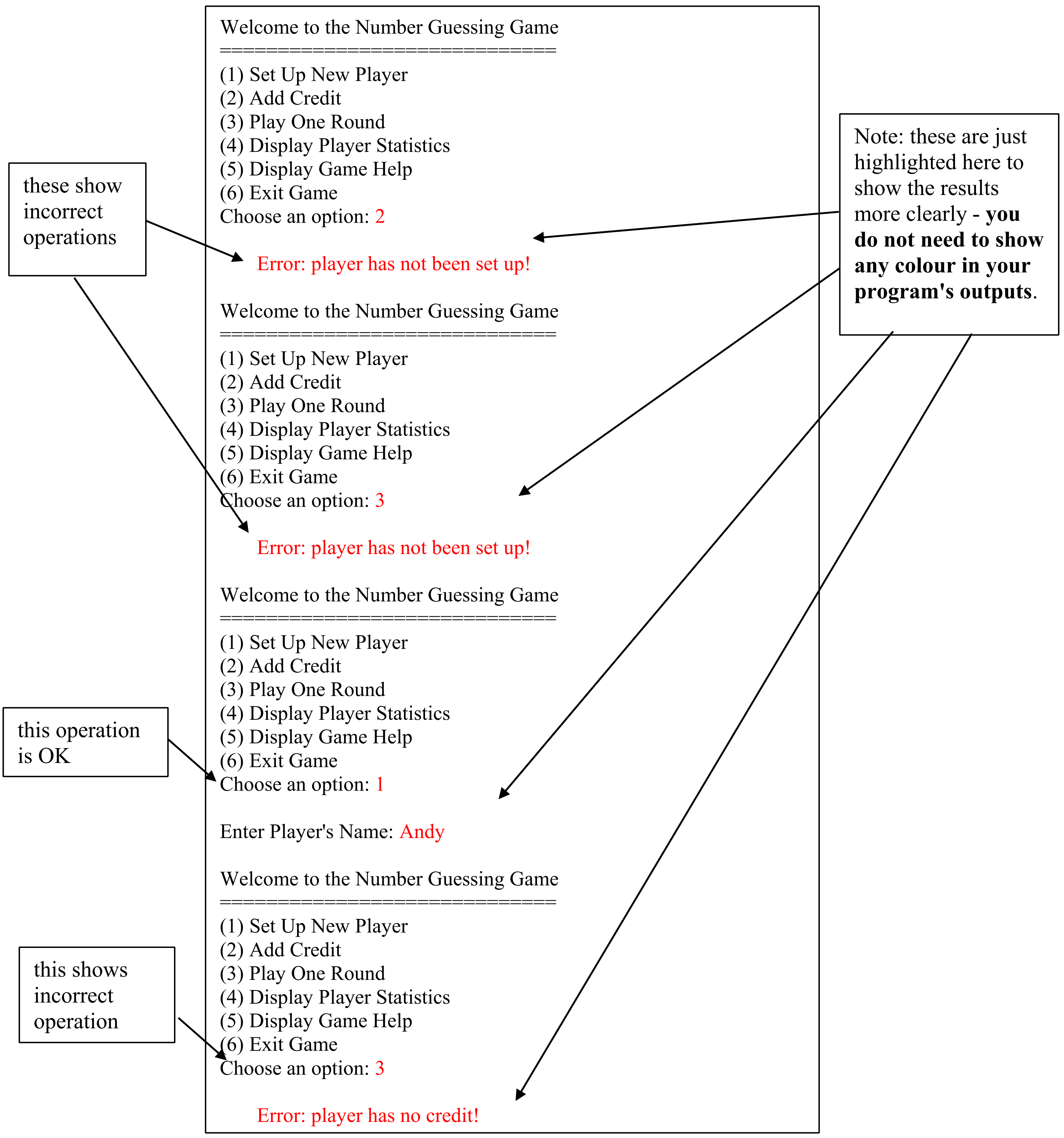
Note: the percentage can be displayed in many ways (with decimal places, rounded-up integers, etc). Discuss with your tutor regarding how you should implement this in your program.

Option (5) displays some brief instructions regarding game play.

Option (6) exits the program.

15 August 2021

Sample screenshots of typical inputs/outputs (including invalid inputs):



```
Welcome to the Number Guessing Game
================================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 2

    Error: player has not been set up!

Welcome to the Number Guessing Game
================================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 3

    Error: player has not been set up!

Welcome to the Number Guessing Game
================================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 1

Enter Player's Name: Andy

Welcome to the Number Guessing Game
================================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 3

    Error: player has no credit!
```

these show incorrect operations

this operation is OK

this shows incorrect operation

Note: these are just highlighted here to show the results more clearly - **you do not need to show any colour in your program's outputs**.

Sample screenshots continued…

```
Welcome to the Number Guessing Game
===============================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 2

Enter a credit amount between 1-20: 25
    Error: only enter a number between 1-20!

Welcome to the Number Guessing Game
===============================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 4

Player Andy has 0 win(s) and 0 loss(es)  ==>  Winning Percentage = 0%.
Total Credit: $0
Total Amount Won: $0
Total Amount Lost: $0


Welcome to the Number Guessing Game
===============================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 2

Enter a credit amount between 1-20: 20


Welcome to the Number Guessing Game
===============================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 3

Enter a number between 1-50 (1 of 3 guesses): 25
Sorry, you need to go HIGHER (2 of 3 guesses): 35
Sorry, you need to go HIGHER (3 of 3 guesses): 38

Lucky Number was: 46, your final guess was: 38
Sorry, better luck next time!
```

another incorrect operation

this shows a player's statistics

this shows a LOSS (ie. no match after 3 guesses), with a $5 penalty

Sample screenshots continued…

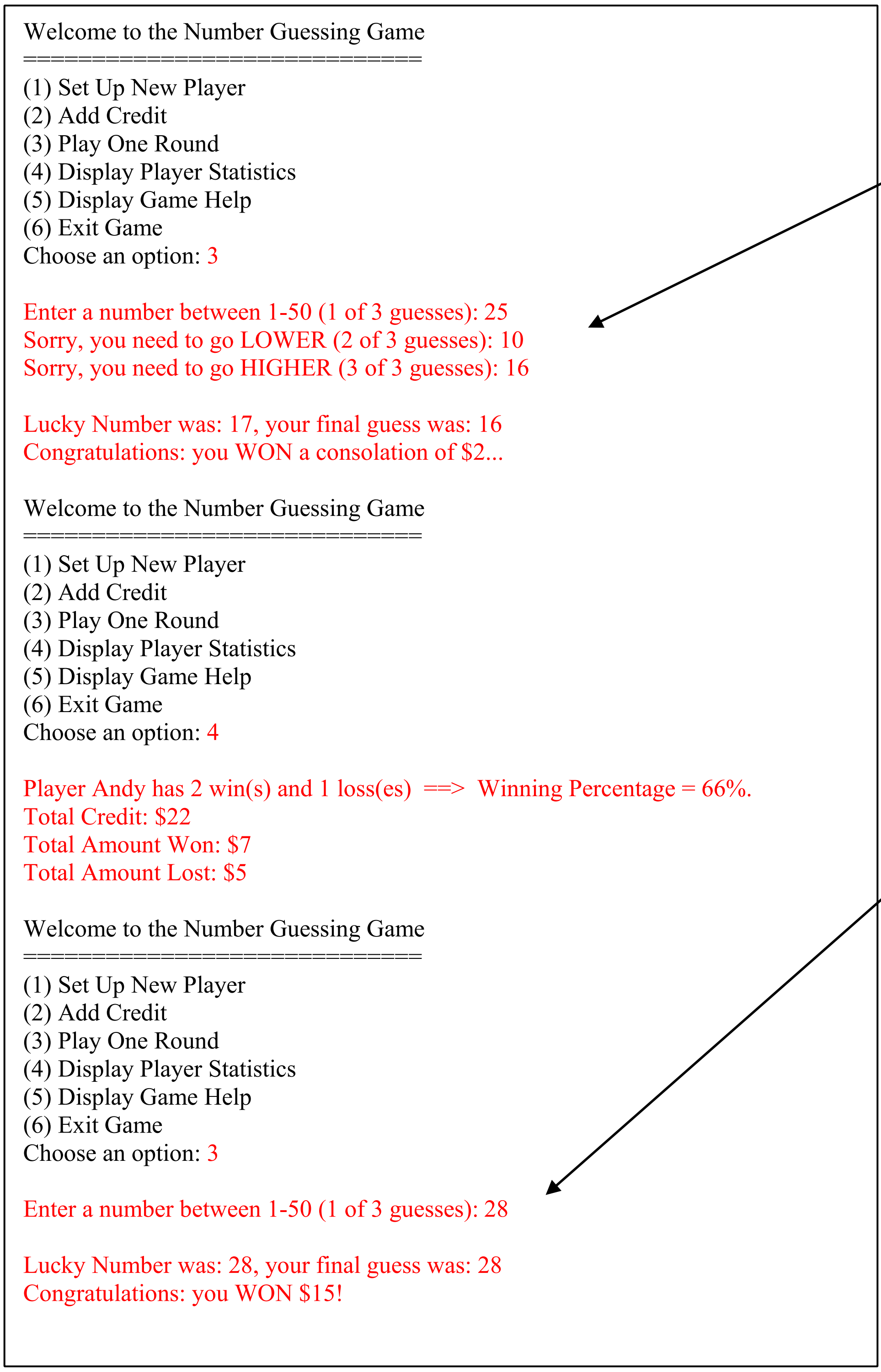
```
Welcome to the Number Guessing Game
==============================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 4

Player Andy has 0 win(s) and 1 loss(es)  ==>  Winning Percentage = 0%.
Total Credit: $15
Total Amount Won: $0
Total Amount Lost: $5

Welcome to the Number Guessing Game
==============================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 3

Enter a number between 1-50 (1 of 3 guesses): 30
Sorry, you need to go HIGHER (2 of 3 guesses): 40
Sorry, you need to go HIGHER (3 of 3 guesses): 47

Lucky Number was: 47, your final guess was: 47
Congratulations: you WON $5!

Welcome to the Number Guessing Game
==============================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 4

Player Andy has 1 win(s) and 1 loss(es)  ==>  Winning Percentage = 50%.
Total Credit: $20
Total Amount Won: $5
Total Amount Lost: $5
```

new statistics after losing

this shows an actual $5 WIN

new statistics after winning

Sample screenshots continued…



```
Welcome to the Number Guessing Game
==============================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 3

Enter a number between 1-50 (1 of 3 guesses): 25
Sorry, you need to go LOWER (2 of 3 guesses): 10
Sorry, you need to go HIGHER (3 of 3 guesses): 16

Lucky Number was: 17, your final guess was: 16
Congratulations: you WON a consolation of $2...

Welcome to the Number Guessing Game
==============================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 4

Player Andy has 2 win(s) and 1 loss(es)  ==>  Winning Percentage = 66%.
Total Credit: $22
Total Amount Won: $7
Total Amount Lost: $5

Welcome to the Number Guessing Game
==============================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 3

Enter a number between 1-50 (1 of 3 guesses): 28

Lucky Number was: 28, your final guess was: 28
Congratulations: you WON $15!
```

this shows a consolation WIN (ie. ±5 match). Note that the amount won is random.

this shows a $15 WIN (ie won on 1st guess)

Sample screenshots continued…

```
Welcome to the Number Guessing Game
================================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 3

Enter a number between 1-50 (1 of 3 guesses): 11
Sorry, you need to go HIGHER (2 of 3 guesses): 41

Lucky Number was: 41, your final guess was: 41
Congratulations: you WON $10!
```

this shows a $10 WIN (ie won on 2nd guess)

```
Welcome to the Number Guessing Game
================================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 3

Enter a number between 1-50 (1 of 3 guesses): 11
Sorry, you need to go HIGHER (2 of 3 guesses): 22
Sorry, you need to go HIGHER (3 of 3 guesses): 35

Lucky Number was: 35, your final guess was: 35
Congratulations: you WON $5!
```

this shows a $5 WIN (ie won on 3rd guess)

```
Welcome to the Number Guessing Game
================================
(1) Set Up New Player
(2) Add Credit
(3) Play One Round
(4) Display Player Statistics
(5) Display Game Help
(6) Exit Game
Choose an option: 4

Player Andy has 5 win(s) and 1 loss(es)  ==>  Winning Percentage = 83%.
Total Credit: $52
Total Amount Won: $37
Total Amount Lost: $5
```

new statistics after winning

…….. (the rest of the game not shown) ……..

NB : all the above screenshots are related (ie. should be read from top to bottom)

Additional Notes :

The main menu must be displayed repeatedly after each operation (as shown in the screenshots), until the user chooses **Option (6)**. Inputs other than 1-6 (including non-numeric characters) should be rejected, and an error message printed.

When **Option (6)** is chosen, the program should clear the player's data before exiting.

If the user chooses **Options (3)**, before a player has been set up ***and*** at least $5 credit is available, an appropriate error message should be printed, and the operation aborted and the menu re-displayed.

If the user chooses **Options (2) or (4)**, before a player has been set up, an appropriate error message should be printed, and the operation aborted and the menu re-displayed.

If the user chooses **Option (1)** when a game is in progress (i.e. the current player has started playing), the program should clear the player's data before starting a new round.

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). However, a good program should at least not crash when the wrong type of data is entered.

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 interface.

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 must be chosen carefully with reference 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 ***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`

`LuckyNumberGenerator`

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 all the menus, accepting user responses, and performing the requested operations. It will make use of one (1) **`Player`** object and one (1) **`LuckyNumberGenerator`** object.
- a **`Player`** object will remember his own name, credit balance, number of wins & losses, and total winnings/losses.
- a **`LuckyNumberGenerator`** object will be used to generate a number between 1N, where N is a user-supplied integer.

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

You can add any other appropriate classes in your design. However, if you do, you must discuss these with your tutor first. You should always start with a simple design, and then improve it if needed.

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 object's state **30%**
- Adherence to ***Java Coding Standards*** **10%**
- Program ***functionality*** in accordance the requirements **60%**

You must submit your work by the submission deadline (a late penalty of 10% 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 *Java Coding Standards*.

<u>Interview</u> (Important: 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, ***<u>you will be assessed on your understanding of the code</u>***, and not solely on the actual code itself.

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

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.***

<u>Any student who does not attend an interview will receive a mark of 0 for the assignment.</u>

[redacted]

Do not name your zip file with any other name

[redacted]

there will be no extensions to the due date

[redacted]

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.

Your program will be checked with a code similarity detector.