Lucky Guess Game

Due Date : 12noon on Friday in Week 7 (12th Sept 2014)

Introduction

This assignment is worth 15% 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. 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 according to the ***Java Coding Standards*** for this unit.

Any points needing clarification may be discussed with your tutor in the tutorial class. Your tutor is your "client" - so you need to implement your program to suit his requirements.

Specification

For this assignment you will write a program that plays a rather simplistic ***Lucky Guess 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 1-100, which will then be compared to a number (also between 1-100) randomly generated by the computer. For each "round" of the game, the player will have 3 chances to guess the number.

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 is the total winning, and the percentage of wins.

For this assignment, the program will only handle *ONE* player at any one time. However, the player can be changed while the program is executing.

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

Program Logic

The *Lucky Guess Game* begins with a welcome message followed by a menu with the following options :

```
Welcome to the Guessing Game
============================
(1) Change (or Set Up) Player
(2) Play One Round
(3) Display Player Wins Statistics
(4) Display Game Help
(5) 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 wins, 0 losses, $0 winnings). Note that the "new" player replaces the "old" player – there is only ever one player at any one time.

Option (2) asks the user to guess a number between 1-100. The computer then randomly generates a "lucky number", also between 1-100, 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 $10, otherwise he is asked to guess again, up to a maximum of 3 times for each round. The program should also tell him if his next guess should be higher or lower. If none of the 3 guesses is an exact match of the lucky number, he wins nothing. However, if his ***final guess*** is within ±5 of the lucky number, he wins a consolation prize of $5. The number of wins and losses, plus the amount won, 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 (3)** later.

At the end of each round, the program should 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 zero, or more than 100, it should be rejected, and the *outcome regarded as one wrong guess*.

Examples of some typical outputs (including some invalid inputs) :

```
...

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

Enter a number between 1-100 (up to 3 guesses) : 10
Sorry, you need to go HIGHER : 14

Luck Number was : 14, your final guess was : 14
Congratulations : you WON $10...

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

Enter a number between 1-100 (up to 3 guesses) : 50
Sorry, you need to go LOWER : -10
Sorry, only numbers between 1-100 are valid. Try again : 2

Luck Number was : 5, your final guess was : 2
Congratulations : you WON $5...

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

Enter a number between 1-100 (up to 3 guesses) : 99
Sorry, you need to go LOWER : 3
Sorry, you need to go HIGHER : 77

Luck Number was : 96, your final guess was : 77
Sorry, better luck next time!

...
```

this shows a WIN (ie. exact match) after 3 guesses

this shows a consolation WIN (ie. ±5 match), and also an invalid input.

this shows a LOSS (ie. no match) after 3 guesses

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**.

Option (3) displays some statistics about the current player's wins/losses, his overall winning percentage, and total $winnings, such as :

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

Player Andy has 2 win(s) and 1 loss(es)  ==>  Winning Percentage = 66%.
Total Wiinnings : $15
```

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

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

Option (5) exits the program.

Additional Notes :

The menu should be displayed repeatedly, until the user chooses Option (5). Inputs other than 1-5 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.

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

For both Options (3) & (4), the outputs can be formatted in many different ways. Discuss with your tutor regarding how you should implement these in your program.

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.

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/behaviours 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 1 **`Player`** object and 1 **`LuckyNumberGenerator`** object.

- a **`Player`** object will remember his own name, number of wins & losses, and total winning.

- a **`LuckyNumberGenerator`** object will be used to generate a number between 1-100.

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 FIT5134 Java coding standards - **10%**; and
- 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%).

Marks will be deducted for untidy/incomplete submissions, and non-conformances to the *FIT5131 Java Coding Standards*.

Interview

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 (the marker may also delete excessive in-code comments before you are asked to explain that code).

Interview times will be arranged in the tutorial labs in *Week 7*. It is your responsibility to attend the lab and arrange an interview time with your tutor. ***Any student who does not attend an interview will receive a mark of 0 for the assignment.***

The actual interviews will take place during the normal lab times in *Week 8*.

Submission Requirements

The assignment must be uploaded to ***Moodle*** before **12noon on Friday in Week 7 (12th Sept 2014)**. The link to upload the assignment will be made available in the ***Assignments*** section of the unit's Moodle site nearer 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 you 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 nearer the submission deadline. You simply complete the editable sections of the document, save it, and include it in your .zip file for submission.

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.

Plagiarism (http://www.policy.monash.edu/policybank/academic/education/conduct/plagiarism-policy.html)

Discipline: Student Policy (http://www.policy.monash.edu/policy-bank/academic/education/conduct/student-discipline-policy.html