FIT9131 Assignment A, S1 2022 - Numble Game

FIT9131 Assignment A: Numble Game

Introduction

This assignment is due by 4:30pm Thursday of Week 7 (14th April, 2022). It 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 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 workspace environment on the Ed platform.

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 applied learning classes.

Completion of this assignment contributes towards the following FIT9131 learning outcomes:

1. Design and construct Java programs according to standard object-oriented principles

2. Apply and demonstrate debugging processes to Java applications

3. Develop strategies for efficient and effective program testing

4. Document code according to specific programming standards

Specification

For this assignment, you will write a program that will allow a person to play a number guessing game against the computer. 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 *Numble Game* is for a person and the computer to compete against other to correctly guess a hidden number.

A game consists of four rounds. For each round, a number between 1 and 100 (inclusive) is randomly generated and the players (person and computer) take turns to guess the number. The round ends when the correct guess is given or each player has had three guesses.

If a player guesses the number correctly then they are awarded points according to how many attempts were taken to guess the number. If the round ends without either player guessing correctly then the points are awarded to the player whose last score was closest to the hidden number.

At the end of the four rounds, the player with the highest cumulative score wins the game.

Gameplay

The *Numble Game* begins with a message inviting the human player to enter their name. The name can contain any characters but must be no more than 8 characters in length. The other player will be the computer. A number with a value from 1 to 100 (inclusive) is randomly generated but hidden from the players. The player who will have the first turn at guessing the number is then randomly chosen by the computer. The round then progresses with the players taking turns until the correct number is guessed. Note that your program will generate a number guess for the computer player.

The following are the game rules:

- If the number is not guessed correctly then a message is displayed indicating whether the entered number was higher or lower than the hidden number and then the other player takes a turn. Note this means that after each turn the range of possible numbers is reduced.
- If the player enters a number between 1 and 100 but not within the possible range, then the player is given a warning message but is not given a chance to re-enter the number.
- If the player enters a number less than 1 or greater than 100, then a warning message is displayed and the player is invited to enter another number (with no penalty).
- If the player enters non-numeric characters, then a warning message is displayed and the player is invited to enter another number (with no penalty).
- If a player correctly guesses the number then the round ends, points are awarded to this player according to how many attempts have been made. Note the total number of attempts includes attempts by both players. The other player scores zero for the game.

Number of attempts	Score
1	18
2	12
3	8
4	5
5	3
6	2

- If the human player enters 999, this indicates that they have decided to abandon the round. The computer will randomly decide to abandon a round approximately once in every 20 guesses. If a player decides to abandon the round then the other player is awarded the points for the number of attempts that have been made, i.e. if the computer abandons the game after four

attempts then the human player is awarded 5 points for that round. (*Hint*: to determine if the computer decides to abandon a round then choose a number between 1 and 20 as the 'abandon game indicator'. For each round generate a random number between 1 and 20 inclusive. The game is then abandoned if the random number is equal to the abandon number indicator).

- If the round ends and no player has guessed the number then the player whose last guess was closest to the hidden number is awarded a score of 1 point. If the guesses were equidistant from the hidden number then no score is awarded to either player.

Program design

Your program should consist of at least three classes: Player, Game and NumberGenerator. The following two sections give details of these classes.

Player class

The Player class will specify the attributes and behaviours of a player. An object of the Player class will have the following fields (at least):

Name – the name of the player.

Guess – the last number guessed for the current round

Score – the cumulative game score

The data type of each field must be chosen carefully 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 state any assumptions made. The class must also have a default constructor and a non-default constructor that accepts a value for the name of the player.

The Player class should also have *appropriate* accessor and mutator methods for its fields. Validation of values for fields should also be implemented. You should not allow an object of class Player to be set to an invalid state. There should be no input from the terminal or output to the screen. A Player object should also be able to return its state in the form of a String.

Game class

The Game class will manage the playing of a game. It will have the following fields (at least):

Player1 (an object of type Player)

Player2 (an object of type Player)

Note that one of these players will be the computer.

The Game class will have methods to manage the playing of the game. These should include the **main()** method to start the program and methods for the following behaviours:

- Display a welcome message on the screen.
- Request the player to enter their name.
- Request the player to enter a number.
- Compare the number entered by a player with the hidden number.
- Display the result of the attempt at guessing the number.
- Display the result for the end of a round, including the score for each player and the value of the hidden number.
- Display the game result.

NumberGenerator class

An object of the NumberGenerator class will generate a random number from 1 to a maximum value specified.

Assessment

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

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

You must use the workspace environment in the Ed platform to code all parts of your program. You must not copy paste huge chunks of code from somewhere else.

You must ensure that your program code meets the coding standard requirements outlined in the unit.

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

You must submit your work by the submission deadline on the due date (a late penalty of 10% per day, inclusive of weekends, of the possible marks will apply). There will be no extensions - so start working on it early.

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/design, modify your code, and discuss your design decisions and alternatives. Marks will not be awarded for any section of code/design/functionality that you cannot explain satisfactorily (the marker 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 on the actual code itself.

- For **on-campus students**, interview times will be arranged in the applied classes in Week 7 and will take place on campus.
- For **online students**, the interviews will be organised in the applied classes in Week 7 and will take place online via zoom.

It is your responsibility to make yourself available for an interview time. **Any student who does not attend an interview will receive a mark of 0 for the assignment.**

Submission Requirements

The assignment must be submitted by **4:30 pm Thursday of Week 7 (14th April 2022)**.

The submission requirements for Assignment 1 are as follows:

- The main class in your program **MUST** be called **Game.java** and it should contain the **main()** method to start the program.
- Submit your work via the Ed platform.
- Re-submissions are allowed before the submission deadline. Please ensure that you do not click on submit button after the due date. The last submission will be used for grading purposes and a submission after the deadline will incur a late penalty.
- A signed Assignment Cover Sheet. [Note: You are required to download the Assignment Coversheet, sign the document and upload the pdf file in the Ed platform(you may drag and drop to the Toggle Pane)]

Marks will be deducted for any of these requirements that are not complied with.

Warning: there will be no extensions to the due date. Any late submission will incur a 10% 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 (e.g. interruptions to your home internet connection).

Plagiarism

Plagiarism and collusion are viewed as serious offences. All submitted code will be subjected to a similarity checker, and any submissions determined to be similar to a submission from a current or past student will be investigated further. The outcome of the decision pertaining to plagiarism and/or collusion will be determined by the faculty administration. To ensure compliance with this requirement, be sure to do all your coding in the Ed workspace environment and do not copy and paste any code into the workspace environment.

In cases where plagiarism or collusion has been confirmed, students have been severely penalised, ranging 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.

Student Academic Integrity Procedure (https://www.monash.edu/__data/assets/pdf_file/0004/2300935/Student-Academic-Integrity-Procedure.pdf)