Guess a C♠rd Game

Introduction

This assignment is due by 10pm on Friday of Week 7 (8 May, 2020). It 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. 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 your online tutorial session.

Specification

For this assignment you will write a program that will allow a person to play a *Guess a C♠rd Game*. This section specifies the required functionality of the program. The interface must be a BlueJ Terminal Window, otherwise zero marks will be awarded. Only a simple screen presentation of the game is required; however, more marks will be gained for a game that is easy to follow with clear informative messages to the player.

The aim of the Guess a C♠rd Game is for a player to correctly guess a playing card drawn randomly by the computer. The player has a set number of attempts to guess the card. However, they are allocated points at the start of the game and are penalised for each incorrect attempt. The game ends when a correct guess is entered, the points are zero or less, or the player has run out of guesses. The points remaining at the end of the game are counted as the score for the game. The player may play a series of games, with cumulative game results displayed after the final game when the player decides to stop playing.

The deck of cards from which the cards are drawn is a standard fifty-two card deck. It has four suits: Hearts♥, Diamonds♦, Clubs♣ and Spades♠. Each suit has thirteen cards: an Ace, which is regarded as number 1 for this game; cards numbered 2 to 10 inclusive. Each suit also three "face cards": a Jack, number 11 for this game; a Queen, number 12 for this game; and a King, number 13 for this game.

Game play

The *Guess a C♠rd Game* begins with a welcome message followed by a message inviting the player to enter their name. The player's name cannot be blank and can contain only alphabetic characters. Two numbers, the first with a value from 1 to 4 (inclusive) and the second with a value from 1 to 13 (inclusive), are then randomly generated by the program but hidden from the player. The two numbers represent the card drawn from the deck. So, if the random numbers were 2 and 1, this would mean the card drawn was the Ace of Diamonds. The random numbers 4 and 13 would mean the card drawn was the King of Spades … and so on.

The player starts each game with a point score of 40 points. They will first guess the suit of the card drawn from the deck. The player will enter a single character to signify their choice of suit: H for Hearts, D for Diamonds, C for Clubs and S for Spades. The player has three attempts to guess the correct suit. Points are deducted from the player's score **each time** the player guesses incorrectly. The points deducted during this part of the game are shown in Table 1.

Incorrect guess	Points deducted
1st	5
2nd	10
3rd	15

Table 1: Point deductions for incorrect guesses for the card suit. For example, 2 incorrect guesses will be 15 points deducted.

If the player guesses incorrectly on the first two attempts, a message is displayed informing them so and they are invited to enter another guess. After a third incorrect guess the player is informed once again of their incorrect guess, then shown the correct suit by the program and the game moves to the next phase. If the player guesses correctly, on either of their first, second or third attempts, they are shown a message informing them so and the game also moves to the next phase.

After correctly guessing, or failing to guess and been shown the suit of the card drawn from the deck, the player then has four attempts to guess the correct card number. To do this the player will enter an integer value. Points are deducted from the player's score each time the player guesses incorrectly. The points deducted during this part of the game are shown in Table 2.

Incorrect guess	Points deducted
1st	2
2nd	6
3rd	12
4th	20

Table 2: Point deductions for incorrect guesses for the card number. For example, 3 incorrect guesses will be 20 points deducted.

The game continues while the player's points score is greater than 0, until the player guesses correctly, or they have run out of guesses. If the player guesses incorrectly on the first three attempts and they have a points score above 0, the player is shown a message informing them that the correct card number is higher or lower than their last guess. After a fourth incorrect guess the player is shown the correct card number, the game ends, and they lose the game. If the player guesses correctly on any one of their four attempts, they win the game and are shown a message informing them so.

At the end of each game, the player is also shown their points score for the game they have just played. They are also asked if they want to play another game. If they choose not to play another game, they are shown the number of games they have played, the number of games they have won and their highest score for the games they played. The program then terminates.

Please note that your program must deal with invalid values entered by the player with respect to the data values mentioned in the previous sections. During the running of the game, if the player enters an invalid value for either the card suit or the card number, the program should display a warning message and the player should be invited to enter another number without penalty (the invalid value entered will not be counted as a guess).

Program design

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

Students are advised to follow good programming practices and to use loops and appropriate fields where required to ensure good program design.

The data type of each field must be chosen carefully and you must be able to justify the choice of the data type. You may want to include comments in the class to state any assumptions made. Each class should also include appropriate accessor and mutator methods for its fields.

Validation of values for fields and local variables should also be implemented where appropriate. You should not allow an object of a class to be set to an invalid state.

Class designs

Your program should consist of at least four classes: *Player*, *Card*, *Game* and *RandomNumber*. This section gives an outline of these classes with suggested fields and some behaviours.

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.

score – the current game score

guess – the last suit or number guessed

highestScore – the highest game score achieved

numberOfGamesPlayed – the total number of games played

numberOfGamesWon – the total number of games won

The class must also have a non-parameterised (“default”) constructor and parameterised constructor that accepts a value for the name of the player. A Player object should also be able to return its state in the form of a String.

Card class

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

suit – the card suit.

number – the card number

The class must also have a non-parameterised (“default”) constructor and parameterised constructor that accepts values for the suit and number of the card. A Card 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 field (at least):

cardGamePlayer (an object of type Player)

The Game class will have methods to manage the playing of the game. These should include (at least) the following behaviours:

- Display a welcome message on the screen.
- Request the player to enter their name.
- Request the player to enter a suit.
- Request the player to enter a card number
- Compare the suit entered by a player with the hidden suit.
- Compare the number entered by a player with the hidden number.
- Display the result of the attempt at guessing the suit.
- Display the result of the attempt at guessing the number.
- Display the result for the end of a game.
- Display the overall result when the player decides to stop playing.

Note that all code for input from the terminal or output to the screen must be in the Game class. There should be no code for input from the terminal or output to the screen in any other class.

RandomNumber class

An object of the RandomNumber class will generate a random number from 1 to a maximum value specified. The maximum value should be specified via a paramaterised constructor.

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 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

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.

The interviews will be organised during week 7 and will take place online via Zoom or other video facility during Week 8. 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 uploaded to Moodle by 10pm Friday of Week 7 (8 May, 2020).

The submission requirements for Assignment 1 are as follows:

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

- the BlueJ project you created to implement your assignment.
- 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.

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.

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 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 (e.g. 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/policy-bank/academic/education/conduct/plagiarism-policy.html)