Programming in Java

Assignment 3

Card Game

Marks available: 100

Date issued: 01/11/19

Deadline: 11/11/19, 17:00

This assignment is worth 12.5% of your final module mark.

The functionality component of the following exercises is worth 80% of the marks for each exercise. This means, the extent to which your methods provide the required behaviour as determined by the JUnit tests.

The remaining 20% for each exercise will be awarded for the 'quality' of your code. For this exercise that means only your layout and naming. You should refer to the lecture notes on this topic (lecture 3a) and also the style guide document on Canvas called 'style.pdf'.

Submission instructions

Submit your work by the deadline above as a zip archive. Use only 'zip' format and no other. Submit only .java source files in your archive. Do NOT submit .class files or any type of file. Please do not submit entire Eclipse (or other IDE) projects. In your zip you must submit all Java files necessary for the assignment, including any that were provided for you.

To submit, gather all of the Java files into a folder, zip that folder, and then submit the zip.

In your programs, do not output anything to the console unless explicitly asked to. If you add `System.out.println()` calls in your code (e.g. for debugging), remove them before you submit.

Each Java file must have a package declaration at the topic of it (these declarations must be on the very first line of the file). The package declaration for all of the exercises below is:

com.bham.pij.assignments.pontoon;

Do not forget to add the semicolon at the end of this line. All characters in a package name should be lower case.

In each of these exercises, you are not restricted to only creating the *required* methods. If you want to, you can create others. However, you must create **at least** the required methods.

Do not put any code that gets user input in the required methods below. If you do, your work can't be tested. This means that you should not use `Scanner` in any required method or in any method called by a required method, etc.

Please follow these instructions. If you do not you could get a lower mark than you would have done for following them or a mark of zero. Please ask if you are unsure.

Introduction

For these exercises you will write some classes that can be used in a card game. The documentation for the three required classes is available on Canvas. You must read that in order to see what your classes need to do. Open the docs/index.html file to view the documentation.

You must create the classes but the main thing you need to do is implement the methods. You can decide for yourself any validation that you wish your methods to do, but make sure that any such validation does not mean that the method might not function as required.

The underlying implementation of these classes is up to you. You must, however, provide the public interface, as shown in the documentation.

The three exercises below require you to write three seperate classes that can be used together to create a card game. The extension activity at the end, which does not carry any marks, involves you using your classes to actually make a version of the card game 'Twenty-One'.

For those who are unfamiliar with playing cards, a playing card has a *suit* and a *value*. The suit is one of the following: HEARTS, CLUBS, DIAMONDS, SPADES. The value is a numerical value in the range 2-10 inclusive, or if the card is a 'picture card' (JACK, QUEEN or KING) it has the value 10, or if it is an ACE then it has the value 1 and/or 11 depending on the card game. Have a look at some playing cards online if you have never seen them before.

Exercise 1: the `Card` class [50 marks]

The `Card` class represents a single playing card.

Read the supplied HTML documentation for the `Card` class to see what you need to implement. Look at each method to see the required signature. Provide a method for each that implements the required functionality.

Exercise 2: the `Deck` class [30 marks]

The `Deck` class represents a full deck of cards, which is 52 cards in total. There is no need to represent 'jokers'. Once a card is dealt from a deck, it can not be dealt again until the deck is reset.

Read the supplied HTML documentation for the `Deck` class to see what you need to implement. Look at each method to see the required signature. Provide a method for each that implements the required functionality.

Exercise 3: the `Hand` class [20 marks]

The `Hand` class represents a single hand of cards in a game. Each player in a game will usually have one hand of cards.

Read the supplied HTML documentation for the `Hand` class to see what you need to implement. Look at each method to see the required signature. Provide a method for each that implements the required functionality.

Extension Activity [0 marks]

Use your classes to create a game of 'Twenty-One'. This game can operate entirely through the standard input and output channels (no graphics required). You will need to write a main class to run the game, and any other classes that you deem necessary. You can find rules for Twenty-One online. There are numerous variations of the game, including Pontoon and Blackjack.

Think about how you would automate a player for this game. What would be a good set of rules for the player to use?