CSCI 202: Object-Oriented Programming

Homework 1 - Due Thursday February 8, 2024 at 11:59pm

Resources:

- Java API index: https://docs.oracle.com/en/java/javase/17/docs/api/index-files/index-1.html

Program description:

In this homework you will create a Java program for playing the card game twenty-one. Most of the code is given to you. Your task will be to complete key components of this program by writing the missing Java code. When you are done the program should look similar to the following (see Figure 1):

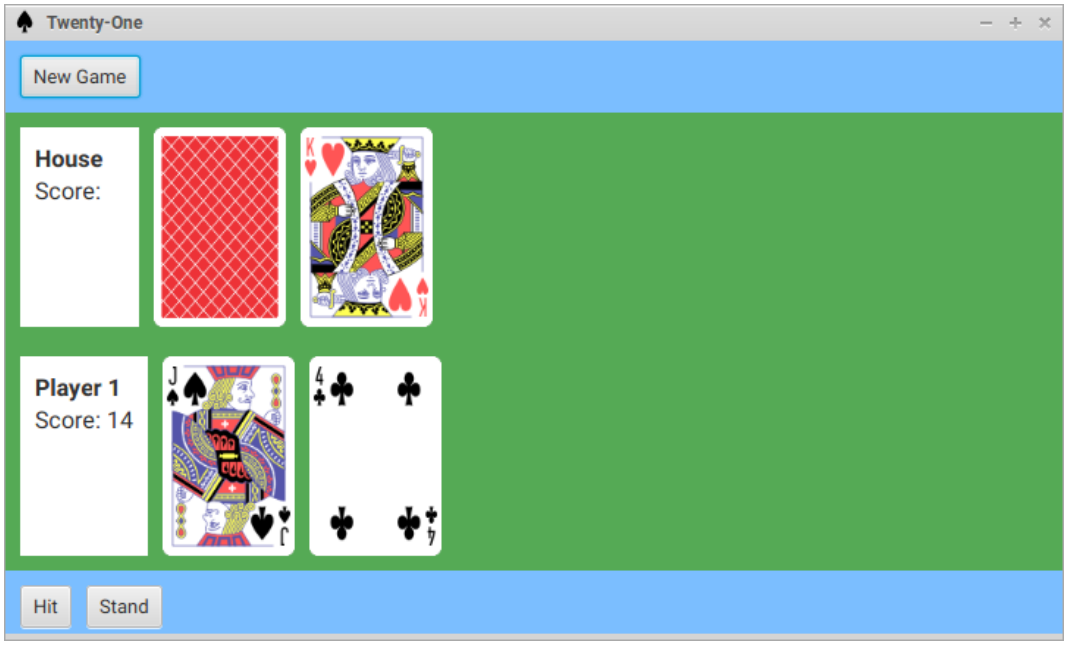

FIGURE 1. Completed program

How to play twenty-one:

Equipment: Standard deck of 52 playing cards.

Goal: Beat the house by getting a closer score to 21 without going over 21.

Card Values: Face Cards (Jack, Queen, King) are worth 10 points. Numeric cards (2 through 10) are worth the value on the card. An Ace is worth either 11 points or 1 point. It is worth 11 points if this does not make the total score go over 21, otherwise it is worth 1 point.

Set up: The deck of 52 cards is mixed. Then two cards are given to you face up from the deck. The house gets one card face up and one card face down from the deck.

Play: Once the game has been set up the game starts. You can either hit or stand. Hit means you take another card from the deck. Stand means you do not take another card. You can hit as many times as you want before standing, as long as your score does not go over 21. If your score goes over 21, you bust and the house wins (at which point the game ends). Once you stand you are done for the rest of the game. Then it is the house's turn. The house must hit (take a card) as long as the score of his face up cards is less than 17. Once his score is 17 or more, he must stand and the game ends.

Winning: When the game ends the house reveals his face down card and adds it to his score. If you went bust (your score is over 21) then the house wins. Otherwise the player whose score is closest to 21 without going over wins. If you have the same score as the house, then the house wins.

Program Overview:

Download and unzip Hw1.zip from canvas. Then open up the unzipped project with IntelliJ.

The Java source files are located in package `twentyone`.

Each card in the program is an instance of the `Card` class. Cards have a rank (such as Queen), a suit (such has Hearts), and a value for if the card is face up or not. The rank is an instance of type `Card.Rank` (which can be called `Rank` inside the `Card` class). The suit is an instance of type `Card.Suit` (which can be called `Suit` inside the `Card` class).

The twenty-one game is represented by an object. It is an instance of the `TwentyOneGame` class. Its data fields make up the state of the game. Actions in the game can be made by calling the appropriate method of this object.

The class `TwentyOne` consists of the code for the program's graphic user interface. It also contains the twenty-one game object (an instance of the `TwentyOneGame` class). You will not need to edit any code in this file.

Part 1: Finishing the Card class

The `Card` contains the type `Rank` and `Suit` (specified as enums). DO NOT CHANGE THESE.

In the `Card` class do the following:

1. Add private data fields for the card's rank, suit, and a boolean that represent if the card is face up or not (true = face up, false = face down). The card's rank must be of type `Rank`. The card's suit must be of type `Suit`.
2. Finish writing the constructor. Initialize the data fields to their corresponding parameters.
   1st parameter: the card's rank.
   2nd parameter: the card's suit.
   3rd parameter: a boolean indicating if the card is initially face up or not (true = face up, false = face down).
3. Finish writing the `getRank()` method to return the card's rank.
4. Finish writing the `getSuit()` method to return the card's suit.
5. Finish writing the `isFaceUp()` method. This method returns the value that indicates if the card is face up or not (true = face up, false = face down).
6. Finish writing the `setFaceUp(boolean newValue)` method.
   Passing `true` to this method sets the card face up.
   Passing `false` to this method sets the card face down.

Do not add any more members to the public interface of class `Card`.

Part 2: Shuffling the deck of cards

In the `TwentyOneGame` class the deck of cards are stored in an array called `deck`.

Locate the `shuffleDeck()` method in class `TwentyOneGame`. In here write code to randomly shuffle the array `deck` using the algorithm described below. Do not use any of the Java library utilities for shuffling the deck (except for the object `generator` defined in this class for generating random numbers).

To shuffle `deck`, loop over `deck` backwards starting at the last element and going down to the second element. In each loop iteration swap the current element (i.e., `deck[i]` where `i` is the loop counter) with a random element to the left of or at the current position. For example, if the current position is 5, then we would randomly select an index `j` from 0 to 5 (inclusive) and then swap the card at position 5 with the one at position `j`.

To generate a random integer between 0 (inclusive) and n (exclusive), call `generator.nextInt(n)`

Part 3: Computing the score

Your task in this part is to compute the score of a given player's hand.

Locate the `getScore(ArrayList<Card> hand)` method in class `TwentyOneGame`. Its parameter `hand` is an ArrayList of all the cards in someone's hand. Write the body of this method to compute and return the total score of all the cards in the ArrayList `hand`.

Value of cards:

- A card that is face down does not add to the total score.
- A face card (Jack, Queen, or King) is worth 10 points.
- A numeric card (2 through 10) is worth the card's rank value.
  The rank value of a `Card c` is `c.getRank().getValue()`
- An Ace is worth either 11 points or 1 point.
  It is worth 11 points if it does not make the total score (the sum of the scores of the rest of the cards) go over `TARGET_SCORE`. Otherwise it is worth 1 point.
  Example: If the hand is {A♠, 5♢, 8♡}, then A♠ is worth 1 point since $11 + 5 + 8 = 24 > 21$.
  But if the hand is {A♡, A♣}, then one of the Aces is worth 11 points and the other is worth 1 point.

Do **not** write the magic number 21 directly in your code. Instead, use the class constant `TARGET_SCORE`.

Note: The house's score will not be shown until the end of the game.

When you are done, zip up your project in a file called Hw1.zip and upload the .zip file to Canvas (click on Assignments and go to Homework 1).

Enjoy the game!