CS 342 **Project#2** **Fall 2019**

Baccarat JavaFX GUI

Due Date: Friday, October 18th, @11:59pm

Description:

In this project you will implement a one player version of the popular casino game Baccarat. This is a somewhat simple game to understand and play which should allow you to focus on learning GUI development in JavaFX and trying your hand at event driven programing.
This project will be developed as a Maven project using the template provided. **You may work in teams of two but do not have to.**

How the game is played:

The user will first bid some amount of money on either The Banker to win or The Player to win or it will be a Draw. These are the only three outcomes of the game.

Next, the dealer will deal two cards each to The Banker and The Player. The Player always gets to go first.

For the cards: 10's and face cards are worth zero points. Ace's are worth one point and all other cards are worth their face value.

If either The Banker's hand or The Player's hand add up to 8 or 9 points, it is a "natural" win and the game is over and bets are paid.

Otherwise, The Player will go first: if hand totals to 5 or less, The Player gets one more card. If the hand totals to 6 or 7 points, no more cards are given.

counting the points: if the total value of the two cards is greater than 9, remove the first number of the total. For example, if I had a 9 and a 6: 9 + 6 = 15 so I would have 5 after dropping the 1.

Next it's The Banker's turn: if the bankers first two cards total 7 or more, no more cards are dealt. If The Banker's cards total 2 or less, The Banker gets one additional card. If The Bankers first two cards total 3, 4, 5, or 6, it depends on if The Player drew another card and if so, the value of that card to determine if The Banker receives another card. See the included PDF for a chart.

The winning hand is the one with a total of 9 or as close to 9 as possible. See the included PDF for betting payouts.

Implementation Details:

You will implement the following:

class BaccaratGame
This class is provided for you but you need to add the following members:

ArrayList<Card> playerHand
ArrayList<Card> bankerHand
BaccaratDealer theDealer
BaccaratGameLogic gameLogic
double currentBet
double totalWinnings

and the method:
public double evaluateWinnings()

This method will determine if the user won or lost their bet and return the amount won or lost based on the value in currentBet.

class BaccaratGameLogic
You need to implement this class with the following methods:

public String whoWon(ArrayList<Card> hand1, ArrayList<Card> hand2)
public int handTotal(ArrayList<Card> hand)
public boolean evaluateBankerDraw(ArrayList<Card> hand, Card playerCard)
public boolean evaluatePlayerDraw(ArrayList<Card> hand)

The method whoWon will evaluate two hands at the end of the game and return a string depending on the winner: "Player", "Banker", "Draw". The method handTotal will take a hand and return how many points that hand is worth. The methods evaluateBankerDraw and evaluatePlayerDraw will return true if either one should be dealt a third card, otherwise return false.

class BaccaratDealer
You need to implement this class with the following members:

```
ArrayList<Card> deck;
```

and the methods:

```
public void generateDeck()
public ArrayList<Card> dealHand();
public Card drawOne()
public void shuffleDeck();
public int deckSize();
```

generateDeck will generate a new standard 52 card deck where each card is an instance of the Card class in the ArrayList<Card> deck. dealHand will deal two cards and return them in an ArrayList<Card>. drawOne will deal a single card and return it. shuffleDeck will create a new deck of 52 cards and "shuffle"; randomize the cards in that ArrayList<Card>. deckSize will just return how many cards are in this.deck at any given time.

class Card
You need to implement this class with the following members:

```
String suite
int value
```

and a two argument constructor:

```
Card(String theSuite, int theValue);
```

Note: You must implement the above just as they are described in the project write up. We will use these data members and methods to test your projects. Failure to do so will result in significant loss of points.

CS 342 Project#2 Fall 2019

The GUI:

You are welcome to use/discover any widget, pane, node, layout or other in JavaFX to implement your GUI. **For this project, you are not allowed to use Scene Builder or FXML layout files**. The following elements are required:

- **There should be an area to display both the Players and Bankers cards with each clearly labeled. You may use images or text to display the cards.**
- **There should be a button to start each round of play.**
- **There should be an area where the user can enter the amount to bid and decide if they are bidding on The Player, The Banker or a Draw.**
- **There should be an area displaying the current winnings for the user**
- **There should be an area that displays the results of each round of play; for example:**

Player Total: 7 Banker Total 5
Player wins
Sorry, you bet Draw! You lost your bet!

Player Total: 3 Banker Total: 8
Banker wins
Congrats, you bet Banker! You win!

You will need to have a menu bar in your program with one tab: **Options**

Under options you will have **Exit** and **Fresh Start**. **Exit** will end the program while **Fresh Start** will reset the current winnings to zero and allow the user to play another game.

Playing the game in your program:

When your program starts, the user should press the play button. The user will then need to bid a dollar amount and decide what they are betting on. Once the choice is made on what to bid on, your program will deal the Player and Banker's hands, pause for a few seconds, end game if there is a "natural" win and post a message and update the current winnings field or allow the Player to either get another card and display it or not, pause again, allow the Banker to select another card and display it or not, pause again, then report the results of the game and update the current winnings field. To play again, the user would press the play button.

Testing Code:

You are required to include JUnit 5 test cases for your program. Add these to the src/test/java directory of your Maven Project. A minimum of two unit tests per required method and one per class constructor.

CS 342 Project#2 Fall 2019

How to Start:

- **Deal a few hands of Baccarat, or play online, to understand the logic and game flow as well as the payouts for a winning bid.**
- **Work on the BaccaratDealer and Card classes first. Try to display Cards in your GUI. You could make them ImageViews or plain text or Buttons or something else. Figure out what you will display them in: Two HBoxs, a Grid, a BorderPane or something else?**
- **Work on swapping out new cards for old. How do you "remove" items from a layout and add new ones?**
- **Design your overall layout on paper first before you implement it. With the exact layouts and widgets you will use and how they are composed.**
- **Test everything as you go, do not wait till the end; it will save you a lot of debugging time!**

Electronic Submission:

If you worked in a group, only one of you needs to submit a project. You must include a PDF file called Collaboration.pdf. In that document, put both of your names and netIds as well as a description of who worked on what in the project. Zip the Maven project BaccaratMaven (and PDF if you worked in a group) and name it with your netid + Project2: for example, I would have a submission called mhalle5Project2.zip, and submit it to the link on Blackboard course website.

Assignment Details:

Late work is accepted. You may submit your code up to 24 hours late for a 15% penalty. Anything later than 24 hours will not be graded and result in a zero.

We will test all projects on the command line using Maven 3.6.1. You may develop in any IDE you chose but make sure your project can be run on the command line using Maven commands. Any project that does not run will result in a zero. If you are unsure about using Maven, come see your TA or Professor.

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml.

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing

your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml.