Project 2 : Blackjack

Released: February 28th

Wireframes due: March 7th at 5pm on Gradescope

Final submission: Thursday, March 14th at 5pm on Blackboard

You may work individually or in pairs for this project

1 Introduction

This project will involve creating an app for playing a game of blackjack against a computer. It will use JavaFX for the front end and will have a Java back end that you will implement as described below. When creating your project, make sure to use the correct `pom.xml` file so that the `mvn clean compile exec:java` command will bring up your app. You will also need to write cases that should run and pass when running the `mvn clean compile test` command. **Do not run your code as an application configuration!**

2 Rules of the game

Your app will model a game of blackjack with two players, a banker and a user, who issues input to the game through a GUI. At the start of the game, two cards are dealt to each player. Both of the user cards should be visible to the user but the first dealer card should be hidden from the user. When given two cards, their values are added. Face cards are worth 10 and Aces are worth either 1 or 11, depending on what is most helpful to the player. Once seeing the two cards, the user has an option to stay on their value or to ”hit” and receive another card. They may continue to ”hit” until the user decides to stay or their numerical value becomes 22 or greater. This is called a ”bust” and the user loses the hand regardless of the dealer’s action. Once the user has decided to stay, it becomes the dealer’s turn. The dealer must ”hit” on all hands 16 and lower and must stay on all hands 17 and higher. At the end, whoever’s hand is higher wins. If both players have the same value hand, it is a draw and the hand is over without any money changing hands.

Finally, if the player hits 21 on two cards by having an Ace and a card with value 10, then that player has ”blackjack” and wins 150% of their winnings unless the dealer also has also hit a ”blackjack”, in which case the game is a draw.

3 Requirements

Your app needs to have the following functionality.

- The ability to choose the starting amount of money
- The ability to set and change the bet amount for each round
- The user must see how much money is left from their initial start value. **the user should not be allowed to reach negative values**, and if the player reaches zero dollars, it should return the user to the start.
- The player should be able to hit any number of times until either "busting", i.e. having a total value greater than 21, or staying.
- There should be some graphical element to the cards and app

3.1 Wireframes

The February 27th lecture discusses wireframes and good design principles. You should sketch out what each of the graphical components for your app are for each scene. Simple boxes are sufficient for GUI components in the app. You may draw your wireframes digitally or by hand. You should have one wireframe sketch for each scene and you should indicate how a user transitions from one scene to another. Each scene should also have a one sentence description. **The wireframes are due to gradescope on March 7th at 5pm**.

3.2 App GUI

GUI design for this project is entirely up to you, but it should conform to good design principles. There is an expectation to use some graphical elements for the app and color choice should be logical and visible. The GUI should be intuitive for use by the TAs when grading and any instructions for how to use the app should appear in the app itself.

3.3 Back end Code

You should implement the following classes to run the logic of the game on the back end.

a) `class Card`

- `String suit`
- `int value`
- Card(String theSuit, int theValue);

b) class BlackjackGameLogic

- `public String whoWon(ArrayList <Card> playerHand1, ArrayList<Card> dealerHand)` : given two hands this should return either `player` or `dealer` or `push` depending on who wins.
- `public int handTotal(ArrayList<Card> hand)`: this should return the total value of all cards in the hand.
- `public boolean evaluateBankerDraw(ArrayList<Card> hand)` : this method should return true if the dealer should draw another card, i.e. if the value is 16 or less.

c) `class BlackjackDealer`

- `public void generateDeck()` : this should generate 52 cards, one for each of 13 faces and 4 suits.
- `public ArrayList<Card> dealHand()`: this will return an Arraylist of two cards and leave the remainder of the deck able to be drawn later.
- `public Card drawOne()` : this will return the next card on top of the deck
- `public void shuffleDeck();` : this will return all 52 cards to the deck and shuffle their order
- `public int deckSize();` : this will return the number of cards left in the deck. After a call to `shuffleDeck()` this should be 52.

d) class BlackjackGame

- `ArrayList<Card> playerHand`
- `ArrayList<Card> bankerHand`
- `BlackjackDealer theDealer`
- `BlackjackGameLogic gameLogic`
- `double currentBet` : the amount currently bet from the user
- `double totalWinnings` the total amount of value that the user has.
- `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.

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.

3.4 Test Code

You should also include a test suite for the above back end classes. Be sure to check each method to be sure it behaves as expected. A small amount of the points will be for detecting broken back end classes.

4 Submitting your work

For the wireframes, create a pdf of your wireframe and submit the file to gradescope by March 7th at 5pm. If you are working in a team, submit as a group on gradescope and submit the team form. Check the blackboard / piazza post on this project for the link. Once you have completed your interface, the backend code and the test suite, you are ready to submit the zip of your project. Make sure that the submission runs with the maven command and be sure to perform a `mvn clean` before submitting. For late days, since the project has come out later than expected, late days can be used for either the wireframes or the final submission, e.g. if you use two late days on the wireframes, you can use two days on the final submission and only use two days cumulatively. Remember that you and your partner should have late days remaining if you plan to submit late.