UNIVERSITY OF VIRGINIA, DEPARTMENT OF COMPUTER SCIENCE

Basic Java 3 - Blackjack

Nada Basit and Mark Floryan

January 13, 2022

1 SUMMARY

For this assignment, you will be writing some code to play the game blackjack. You will write one class, and also a blackjack playing bot that tries to win as many chips as possible. Your summary:

1. Download the starter code and import the project into Eclipse.
2. Implement the DeckStack.java class.
3. Implement the MyBlackjackPlayer.java class.
4. **FILES TO DOWNLOAD:** blackjack.zip
5. **FILES TO SUBMIT:** DeckStack.java, MyBlackjackPlayer.java

1.1 DECKSTACK.JAVA

Your first task is to implement the DeckStack class. Casinos will sometimes use multiple decks to make cheating harder, and to increase the house odds. For our simulation, we want the dealer to be able to deal cards from a shuffled stack of multiple decks. To simulate this, you

will use the Deck.java class from lecture, but write another class that uses it. This class will be called DeckStack.java and it will contain the following fields / methods:

```
//An array of decks of cards that comprise this multi-deck.
private Deck[] decks;

//Constructor: instantiates the number of decks specified and
//adds them to the list of decks
DeckStack(int numDecks);

//Deals the top card from the stack of decks and returns that Card.
//You can implement this several ways, as long as it correctly
//deals a card from one of the decks.
public Card dealTopCard();

//Reshuffles all of the decks.
protected void restoreDecks();

//returns the number of cards left to be dealt in the
//entire stack of cards.
public int cardsLeft();
```

Once you implement this class, the rest of the blackjack simulator has been written for you and should work. You can run the project and see if the simple player we provided plays blackjack (*More detail on the simple player below*).

1.2 Implement a Blackjack Player

Your second task is to implement a basic blackjack player. This involves writing two required (and one optional) methods:

```
/* Returns the number of chips you'd like to bet this hand */
public int getBet();

/* Returns the Move this player would like to do right now */
/* Make move is called until the player returns Move.STAY */
public Move getMove();

/* The dealer will call this method to show you their entire */
/* set of cards once the hand is completely over. You may use */
/* this information if you'd like. */
public void handOver(Card[] dealerHand);

```

```
/* The Move enum looks like this */
public enum Move{
    STAY, HIT, DOUBLE;
}
```

Your job is thus to implement getBet() and getMove() methods such that you maximize your final chip count after 1000 games (the simulator automatically plays 1000 hands). Your player will begin with 1000 chips. In order to reason about your best move, you can access the following variables from within your methods:

```
/* Returns the number of chips the player has */
public int getChips();

/* A list of the player's cards */
/* Access using this.cards */
protected ArrarList<Card> cards;

/* Example of how to see the dealer's face-up card */
this.dealer.getVisibleCard(); //returns a Card object
```

Your goal, as stated earlier, is to maximize your profit after 1000 games. Good luck!

1.3 Grading Guidelines

Your submission will be graded using the following guidelines:

1. The correctness of your *DackStack* class. You will get a majority of the credit for this assignment if this class works as intended.

2. Your blackjack player must successfully play blackjack without crashing the simulator or causing other runtime errors / issues.

3. *MyBlackjackPlayer.java*: Your blackjack player needs to be implemented to apply some reasonable strategy. A grader will glance at your class and make sure you've put in effort to produce a player that is better than random (better than randomly choosing whether to hit or stay, for example).