UF UNIVERSITY of FLORIDA
Herbert Wertheim College of Engineering

COP3502

P1: Blackjack

Overview

In this project students will simulate a simplified version of the game Blackjack. Students will implement random card drawing, calculation, state tracking, and console input systems. The project is designed to be a playful yet practical opportunity to practice data types, console I/O, control structures, and libraries.

Rules of the Road

Forewarning: this specification does NOT follow the rules of casino Blackjack. Don't drop out of your studies and move to Vegas just because you can beat your own randomized AI!

A typical game will play out as follows:

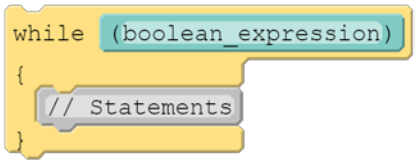
The player will be dealt one card at the start of the game (where a game is one round of play). The player, based on the value of his cards, can either ask for another card (a hit) or choose to hold (stand). The dealer will then begin his turn and try to beat the player's hand. The player is competing against the dealer, who has his own hand. Whomever is closer to 21 at the end of the game (as long as they don't exceed 21) wins the game.

1. *Player's turn*: Player tries to reach or come close to 21 without going over (as 21 is the highest hand).
2. *Dealer's turn*: Dealer tries to beat exceed the player's hand without going over 21.
3. Determine the winner at the end of the game and increment the win count.
4. Repeat!

You will need a while loop in your program. The loop will allow you to play successive games without restarting the program each time; it will also allow you to k eep a win count over multiple games.

Here is the basic syntax of a while loop:

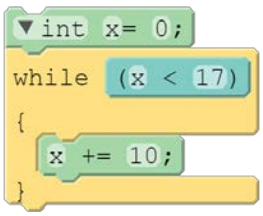
```
while (boolean expression)
{
    // Statements
}
```



If the boolean expression evaluates to true, the loop continues. If/when it ever evaluates to false, then the loop terminates (we skip over the loop block) and continue with the program.

Here is an example of a while loop:

```
int x= 0;
while (x < 17)
{
    x += 10;
}
```



The variable **x** is initialized to zero and the conditional statement is checked. Since `0` is less than `17`, the expression evaluates to `true` and the statement in the while loop block will be executed. The number `10` is added to the value of **x**, so **x** is now `10`. We've reached the end of the loop, so we jump back up to the conditional statement. The expression evaluates to true since `10` is less than `17`, so we execute the statement in the loop again. Now **x** is equal to `20`. Once more the loop block ends and we jump back up to the conditional statement. Since `20` is not less than `17`, the expression evaluates to `false` and the loop block will be skipped, and we jump to the code after the loop.

In this project, you will need to loop until the player chooses the exit option from the menu.

Structure

Now that you have a general overview, here are the nitty gritty details.

When the program starts it should print "**`START GAME #1`**" to the console, and the player should automatically be dealt their first card. With each new game played it should print the corresponding game number in this way.

To determine what card the player is dealt, a random number must be generated between 1 and 13; it is then added to the player's hand. The range of random generation will be from 1-13. The values correspond to these cards:

Value	Card
1	ACE!
2-10	(correspond to their face values)
11	JACK!
12	QUEEN!
13	KING!

Face cards (King, Queen, Jack) are worth a value of 10. If the player is dealt a King (generated as 13), the value 10 should be added to the player's hand value rather than a 13. Aces have a value of 1. Every other card will be worth its face value.

Whenever a card is dealt, print the value of the card and the total value of your hand. (See sample output for format). If the card you were dealt was a face card or an ace then print the type of card. For example, if you were dealt a King you would print "**`Your card is a KING!`**" If the card was not a face card or an ace print the value of the card. For example, "**`Your card is a 2!`**".

After the card is dealt, print the menu to the screen. The menu should look like this:

```
1. Get another card
2. Hold hand
3. Print statistics
4. Exit
```

If the player chooses option 1, the player will be dealt another card. If the player has a hand of 21, they automatically win and "**`BLACKJACK! You win!`**" is printed; then a new game is started. If the player's hand exceeds 21, the dealer automatically wins; print, "**`You exceeded 21! You lose.`**", then start a new game.

If the player chooses to hold their hand (option 2), then the dealer will be dealt his hand. To determine the dealer's hand, generate a random number between 16 and 26 (both inclusive).

If the dealer's hand is above 21, the player automatically wins. If both the dealer and player have the same value hand then no one wins. In this case print "**`It's a tie! No one wins!`**". Otherwise, whoever has the higher

hand value wins the round. If the player wins, print "**`You win!`**". If the dealer wins print "**`Dealer wins!`**" After the winner (or tie) has been determined, a new game is started.

If the player chooses option 3, you will print the statistics of the game. Throughout your program you will need to keep track of the number of games played, the player's number of wins, the dealer's' number of wins and the number of ties. Print out all these values as well as the percentage of player wins to the total number games played. Format your percentage value to one decimal point. See sample output for format.

If option 4 is selected, exit the program.

If any other input is entered, print the following text:

```
Invalid input!
Please enter an integer value between 1 and 4.
```

Then redisplay the menu. You can assume that the input will be numeric for this project (but not future projects.)

Random Numbers

For random numbers, you **must use** the `P1Random` class provided:

```
P1Random rng = new P1Random();
```

You can get a new int value between zero (`0`) and some number (exclusive!) using the `nextInt()` method:

```
int myNumber = rng.nextInt(10);     // Yields a random number in range [0,9]
```

To get a number in a specific range, use these commands:

```
int myNumber = rng.nextInt(13) + 1;  // A random number in range [1,13]
```

```
int myValue = rng.nextInt(11) + 16; // A random number in range [16,26]
```

NOTE: you must only pull random numbers as you need them, and you must use them in that order.

Do not get random values and throw them away! Do not get clever with how you generate random numbers! If you do, your output will diverge, and it could **seriously impact** your grade.

Submissions

NOTE: Your output must match the example output *exactly*. If it does not, ***you will not receive full credit for your submission***!

File: BlackJack.java
Method: Submit on ZyLabs

Do not submit any other files!