A Blackjack Game Program: Part 2 Methods

Please refer to Part 1 for the rules of the game and sample output.

Code Skeleton

In this part of the lab, you will break the program into small, easy to implement parts. Each part will be placed in a separate method which can be called as many times as you wish.

Since this is the first time you will have seen a somewhat complex program, we will provide a set of method stubs for you. A method stub is a method which has the correct header and which will compile, but it doesn't do anything. Stubs serve as placeholders; they define the structure of the final program. Using stepwise refinement, you will implement each stub, one at a time, thoroughly testing as you go.

On the class webpage for the lab, you should see links for a Netbeans project called **BlackJackGameStubs**. Download it and unzip it (if you are unfamiliar with how to unzip a file, please consult the instructor). Open up the project in Netbeans and rename it so that the *project* name contains the word **BlackJack** concatenated with your name (e.g. first initial and last name). You may leave the java file names unchanged. To rename a project, right-click on the project name and select "rename". Make sure that you select the check-box that says "Also Rename Project Folder" – this is very important to do!

If you look at the source code, you will see two classes: **BlackJackGame.java** and **GameMain.java**. The latter just contains the main method. The game stubs are in **BlackJackGame.java**. Comments are included describing what each method should do. You should not have to make any changes to **GameMain.java**. The reason for having two Java classes will be discussed in class. Note, this structure is a little different than what your book does in Chapter 5.

Implementing the Methods

One of the methods you will see is called `prompt()`. This will be where the prompt loop that you wrote in Part 1 will go. Copy your code into this method. The user's answer should be stored in the ans variable which is declared and returned in the method. Once you have entered the code, test it by calling it from the play method. For example, in `play()`, enter the code:

```
String playAgain = prompt();
System.out.println("You answered " + playAgain);
```

Run the code to see if it runs properly. Test it on all possible categories of inputs.

Continue to implement the remaining methods one at a time. Read the comments to help you understand what each method does. It is suggested that you implement them in the following order:

```
pickCard
getSuit
getFace
toStringSuit
toStringFace
toStringCard
cardValue
play
```

The order is chosen so that each method relies on the methods listed above it but not the methods below it. Thoroughly test each method before moving on to the next one. This way any errors you encounter must be due to the method you are currently working on since the earlier methods you know to be working correctly. You can test each method by writing temporary code in `play` which calls the method and prints the result. This code can be removed after testing.

Submitting Your Code

Zip together your entire Netbeans project (ask if you aren't sure how; the resulting file should have a .zip extension.) and submit to WISE as a *single* zipped attachment. Remember, it is important that your project name (though not necessarily the java files) be named as described in the beginning of these instructions.