Closed Lab #3:
War!

1 Overview

The purpose of this closed lab is three-fold. First, you will be introduced to Java arrays. Second, you will gain experience developing Java classes. Finally, you will gain experience using Java objects to implement an object-oriented application.

2 Requirements

Recall that one of the recommended references for this class is the Java Tutorials website published by Sun Microsystems. The tutorials on this site are organized into small online books, referred to as trails. The Learning the Java Language trail is particularly relevant to this course. To begin this closed lab, you are required to read a segment of this trail focused on Java arrays. The segment is available at the following address: http://java.sun.com/docs/books/tutorial/java/nutsandbolts/arrays.html.

After reading this segment, you are required to implement a Java program that plays 52 hands of the classic card game War. Your program must consist of three classes satisfying the following requirements.

• Card. This class will be used to model a single playing card. The class must define private fields to represent a card’s value (e.g., 2, ... 10, Jack, Queen, King, Ace) and suit (e.g., club, diamond, heart, spade). In addition to any public constructors that you decide to include, the class must define two public methods: The first, toString(), will return a reasonably formatted String object corresponding to the card’s value and suit (e.g., “Ace of Spades”). The second, winner(), will accept a Card object as argument, and return a boolean value indicating whether the first card (i.e., the target of the method call) is greater than the second card (i.e., the argument to the method call). Here greater means that the first card has a higher value, or in the case of a tie in value, has a higher suit alphabetically. (So “3 of Clubs” beats “2 of Clubs”, and “5 of Spades” beats “5 of any-suit”.)

• Deck. This class will be used to model a standard deck of playing cards. The class must define private fields to represent 52 playing cards: 13 different values, 4 suits each. (This is where the array tutorial will come in handy.) In addition to any public constructors that you decide to include, the class must define one public method: The draw() method will return a random Card object from the deck. The card returned by the method will also be removed from the deck. (That is, a card returned by the draw() method must never be returned by any subsequent call to draw().)

• MainDriver. This class will provide the main() entry point to your Java application. It will construct two Decks of Cards before entering the 52-hand loop. In each iteration of the loop, you will draw one Card from each Deck, display the Cards drawn, and declare a winner for that hand. Upon termination of the loop, you will display the total number of hands won by each player and declare a winner for the game.
3 Hint

You may need help selecting a random card. Here is a hint: The java.util package includes a variety of useful classes for constructing Java applications. The class Random is contained within this package; it provides a public default constructor (i.e., a public constructor that accepts no arguments). Once you’ve created an instance of Random, you may invoke nextDouble() on the object to return a random double. Alternatively, you may invoke nextInt() to return a random int.

4 Grading

To receive full credit for this lab, you must demonstrate (within Eclipse) that you have satisfied the above requirements. Your TA may also ask you to explain basic array concepts covered in the online trail segment. When you think you are ready, ask one of the TAs to check your solution. When they have checked your name off of their list, you are free to leave. If you are unable to complete the lab before the end of the period, but have made a reasonable attempt, you will receive 1/2 credit for the day. If the TA feels that you have not made a reasonable attempt, you will not receive any credit for your participation.

5 Collaboration

You may work independently or with one partner. You must not discuss the problem or the solution with classmates outside of your group. Choose wisely. Bad partners are bad news.