Project 7: Slot Machine

Due: Monday, Mar. 2 at 11:59 PM

Description: In this project, we will implement a simulation of a slot machine! The user starts with 10 tokens, and the slot machine costs 1 token to play. After each play, the user can choose to quit or continue.

On each play, the slot machine generates three random integers between 2 and 7. The user wins tokens if the numbers match one of the following patterns: (1) all three numbers are the same, (2) two of the numbers are the same, or (3) the numbers form a consecutively increasing sequence. Otherwise, no tokens are won. If the user runs out of tokens, the program ends.

We are providing you with a partial implementation of the program, which is stored in the file SlotMachine.java. This file includes the SlotMachine class, most of the main method, and three empty methods. Your goal is to write the body of each empty method so that it works as described below. Then call the appropriate methods in the main method to create a functioning slot machine simulation!

Objectives: Your program will be graded according to the rubric below. Please review each objective before submitting your work so you don't lose points.

1. (25 points) Write the body of the method int randomInt(int lowerBound, int upperBound).
   - Use Math.random() to calculate a random integer between lowerBound and upperBound (inclusive).
   - Do not hard code the range of random integers. Use the parameters instead.
2. (25 points) Write the body of the method int[] randomIntArray(int lowerBound, int upperBound, int size).
   - Construct an array inside the method with size elements.
   - Use the method randomInt() to assign a random integer between lowerBound and upperBound to each element.
   - Return a reference to the array.
   - Do not hard code the range of integers or the array size. Use the parameters instead.
3. (25 points) Write the body of the method int calculateWinnings(int[] wheelNums).
   - Assume the input array has three elements and check it for each winning condition. Return the correct number of tokens for each case.
   - Print the notification that corresponds to the number of tokens returned.
4. (15 points) Complete the main method.
   - Generate an array of 3 random integers in the range from 2 to 7 (inclusive).
   - Print the array to the user in the format shown in the sample run.
   - Calculate the number of tokens won and assign it to the variable tokensWon.

5. (10 points) Use meaningful variable names, consistent indentation, and whitespace (blank lines and spaces) to make your code readable. Add comments where appropriate to explain the overall steps of your program.

Sample Output: Below is an example run of the program. The user input is in bold. The output of your program should match this if the slot machine generates the same random numbers.

```
Java Slot Machine
-----------------
Tokens: 10
Press enter to play (spend 1 token) or type 'quit' to stop.

Spin: [3, 6, 7]
Sorry, you lost :(
Tokens: 9
Press enter to play (spend 1 token) or type 'quit' to stop.

Spin: [2, 4, 2]
Pair! You win 1 token.
Tokens: 9
Press enter to play (spend 1 token) or type 'quit' to stop.

Spin: [3, 2, 4]
Sequence! You win 2 tokens.
Tokens: 10
Press enter to play (spend 1 token) or type 'quit' to stop.

Spin: [4, 4, 4]
Triple 4s! You win 12 tokens.
Tokens: 21
Press enter to play (spend 1 token) or type 'quit' to stop.
quit
You quit with 21 token(s).
Your net profit is 11 token(s).
```

Notice the following things about this example:

1. Each spin consists of three random integers between 2 and 7.
2. The slot machine costs 1 token to play, so the net change in tokens from one play to the next is the number of tokens won minus 1.
3. The code we provide prints all of this output other than the lines reporting each spin and the winning/losing notifications. You will write the code to print this information.

Import SlotMachine.java into Eclipse: Refer to the Project 6 instructions for information on how to import an existing .java file into a new Java project. Before you modify the code, build and run the program to make sure it is imported correctly.

Three Empty Methods: The file SlotMachine.java contains three empty methods. To complete the slot machine simulator, you need to write the bodies of these methods, which are described below.

1) `int randomInt(int lowerBound, int upperBound)`

This method should return a random integer that is greater than or equal to lowerBound and less than or equal to upperBound. For instance, the method call randomInt(7, 10) should return 7, 8, 9, or 10 with equal probability. To write this method, use the random method from the Math class, which returns a random double that is greater than or equal to 0 and less than 1. If you need help, see the following example calculations.

Example 1: Suppose we want to generate a random integer that is greater than or equal to 0 and less than or equal to 3. This can be accomplished with the following expression:

```
(int) (Math.random() * (3 + 1))
```

Take a moment and think about why this works. When the random number returned by random() is multiplied by 4, it produces a double that is greater than or equal to 0 and less than 4. Dropping the fractional part with the cast operator produces an integer that is greater than or equal to 0 and less than or equal to 3. (The double returned by random() is always less than 1, which is why it must be multiplied by 4, rather than 3.)

Example 2: Now suppose we want to generate a random integer between 2 and 5. The expression above randomly produces one of four numbers: 0, 1, 2, or 3. We still want to produce one of four numbers, but now we want 2, 3, 4, or 5. This can be accomplished by simply adding 2 to the last expression:

```
2 + (int) (Math.random() * (3 + 1))
```

To write the body of randomInt(), you need to write an expression like those in the last two examples. However, you must use the parameters lowerBound and upperBound so that the method will output numbers in the range specified by the arguments. Try to work out the expression on paper before writing the code. After you write the code, test the method by choosing some arguments and calling the method ten or twenty times in a loop.

2) `int[] randomIntArray(int lowerBound, int upperBound, int size)`

This method should return an array of random integers, where each integer is greater than or equal to lowerBound and less than or equal to upperBound. The number of integers in the array is specified by the parameter size.

To write the body of this method, your code will need to do three things: (1) Construct an integer array with the number of elements specified by size. (2) Assign a random integer between lowerBound and upperBound to each element. (Use a loop and the method randomInt() described above to do this.) (3) Return a reference to the array.

3) `int calculateWinnings(int[] wheelNums)`

This method should return the number of tokens output by the slot machine for a given array of random integers. It should also print a message to notify the user how much they won. The slot machine outputs tokens according to the following rules:

1. A pair of matching numbers wins 1 token. In this case, the method prints the message "Pair! You win 1 token."
2. A consecutive sequence of numbers (e.g., 3, 4, 5) wins 2 tokens. In this case, the method prints the message "Sequence! You win 2 tokens." Note that the numbers do not need to be in increasing order (e.g., 5, 3, 4 also wins 2 tokens).
3. Three matching numbers other than 7 wins tokens equal to the sum of the numbers (e.g., three 5s wins 15 tokens). The method prints the message "Triple xs! You win 3x tokens.", where x is replaced with the number and 3x is replaced with the product.
4. Three 7s wins 50 tokens. The method prints the message "Triple 7s! You win 50 tokens."
5. Every other combination of numbers wins no tokens. The method prints the message "Sorry, you lost :(".

When you write this method, assume that the parameter wheelNums is a reference to an array of only three numbers. Before checking for each case, sort the array. This will simplify each conditional statement. (For instance, sorting makes checking for a sequence much easier.) See the Arrays class documentation for a method that will do this for you.

Complete the Main Method: After you have written (and hopefully tested) the methods described in the last section, add a few lines of code to the main method to complete the program. When you generate the array of random numbers, set the lower bound at 2, the upper bound at 7, and the size of the array to 3. To print the numbers, consider using a method from the Arrays class instead of a loop. Make the output match the sample run.

Upload Code to Zybooks: After you complete each project, you need to upload your source code to Zybooks so it can be graded. If you created the folder structure described earlier, your code is in the folder "Intro to Programming\Projects\Project 7\src". The code is contained in the file Project_7.java. Note that .java files are simply text files that contain Java code. They can be opened with any text editor (e.g., Notepad or TextEdit).

Upload your .java file to Zybooks on the Project 7 assignment page. This requires a few steps. Click the "Submit Assignment" button in the top-right corner. This will reveal a button near the bottom of the page with the text "Choose File." Click this button and browse to the location of your .java file. Finally, click the "Submit for grading" button below the "Choose on hard drive" button.

If Zybooks won't accept the file, make sure you're submitting a .java file, not a .class file. Make sure to see if the output you obtained and the output which I have mentioned match to obtain complete grade.

Submission Link: