Programming Lab #3: RollingDice files

This file includes
1. *background information on some over-arching programming ideas*
2. *how to set up your specific integrated development environment (IDE) in order to use the files*
3. *details of the specific coding tasks, in order of completion*

Background

This lab was developed by Maria and Gary Litvin and is used with permission.

This lab focuses on technical concepts we already know – the `if` and `switch` conditionals. But it introduces some new ideas you have not yet experienced. Because of this, you should read through the background information found in the document "RollingDiceLab.pdf" and then return to these instructions. There are some ideas that we will revisit later so don't panic if there's a lot of stuff you don't understand. Work to get an idea of what's going on and register that there is "stuff" you don't get and move on.

Don't go any further until you've read the RollingDiceLab.pdf document!

Objects

A deeper discussion of objects is at the very beginning of the reading. It's important for you do understand the thinking behind the design but don't worry about the discussion of `JFrame` and `Swing`. These are related to GUI development and we will not be responsible for any of that in this lab. It's important to understand that an object can represent something that is abstract – like the rules to a game like Craps.

Unified Markup Language (UML)

On the page numbered 19 there is a picture of classes and how they are related. This is an example of UML. The two main concepts to understand here is that the arrow indicates an is-a relationship (inheritance). For example DisplayPanel is-a JPanel. The diamond end indicates a has-a relationship (service). For example CrapsTable has-a RollingDie. The overlapping diamonds indicates that CrapsTable has more than one RollingDie.

Test Harness

The concept of a test harness is introduced with CrapsTest1.java. This is a small "utility" that allows the programmer to test one component of the program independent of the fully functional application. This could also be referred to as a "sandbox" but sandboxes tend to be more informal.

Setting Up

Note that you will need to unzip the files prior to using them. If you do not already have software to unzip files, you can download the free 7zip application from http://www.7-zip.org/ .

In JCreator:

1. Download and extract the zipped file containing the files and unpack them into a directory named "RollingDie". You may have to use a separate program like 7zip to extract the ZIP file (see link above).

   CrapsGame.java
   CrapsStats.java
   CrapsTest1.java
   RollingDie.java
   RollingDiceInstructions.pdf
   RollingDiceLab.pdf
   CrapsDemo.jar
   ControlPanel.class
   CrapsTable.class
   DisplayTable.class

2. Save these files in a new folder named Craps. Note that you may choose to also have additional folders within Craps as you progress through the tasks.

In DrJava:

1. Download and extract the zipped file containing the files and unpack them into a directory named "RollingDieFiles". You may have to use a separate program like 7zip to extract the ZIP file (see link above).

   CrapsGame.java
   CrapsStats.java
   CrapsTest1.java
   RollingDie.java
   RollingDiceInstructions.pdf
   RollingDiceLab.pdf
   CrapsDemo.jar
   ControlPanel.class
   CrapsTable.class
   DisplayTable.class

2. To create a project file,

   a. Run DrJava.
   b. Use the **New** command in the **Project** menu.

c. Pick a name for the new project file, select a place to save it, and press the **Save** button. I suggest you save it in the RollingDieFiles directory.
d. In the **Project Properties** dialog, edit the project settings as described below.
e. Set the **Project Root** to the directory that corresponds to the default package: RollingDieFiles
f. Press the **OK** button to close the **Project Properties** dialog.
g. Use the **Open Folder...** command in the **File** menu and select the directory you selected as project root in step 3.then press the **Select** button.
h. The source files are now displayed on the left side of the editor.

You are now ready to edit, run, and compile the project with DrJava.

In Eclipse:

1. Download and extract the zipped file containing the files and unpack them into a directory named "RollingDie". You may have to use a separate program like 7zip to extract the ZIP file (see link above).

   CrapsGame.java
   CrapsStats.java
   CrapsTest1.java
   RollingDie.java
   RollingDiceInstructions.pdf
   RollingDiceLab.pdf
   CrapsDemo.jar
   ControlPanel.class
   CrapsTable.class
   DisplayTable.class

2. Make the Java Project "CrapsProject"
3. Copy the following files and put them in your CrapsProject/src folder:
   CrapsGame.java
   CrapsStats.java
   CrapsTest1.java
   RollingDie.java

~ ~

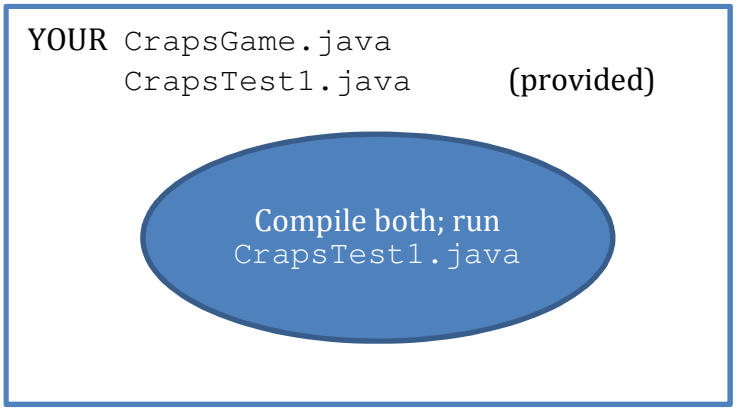
Note that you can double click on CrapsDemo.jar and play a few games of Craps. It is highly recommended that you do this until you completely understand the rules.

Coding Part 1: CrapsGame.java and CrapsTest1.java

`CrapsGame.java` contains a partially implemented method named `processRoll()`. The job of this method is to carry out the rules of the game of `Craps`. Recall that the rules of the game are in the other document, RollingDiceLab.pdf. Notice that this class already has a private data member named `point`. Also note that the variable `result` is important to the code. Read the comments in the class to make sure you understand how everything is supposed to work before you begin implementing the method. As you complete the logic in `processRoll()`, use the already created file `CrapsTest1.java` to make sure the values of `point` and `result` are correct for every possible dice roll combination in the game. We call CrapsTest1.java a test harness because it enables us to make sure the results of `processRoll()` are correct.

Test CrapsGame.java

To test your logic implemented in the `processRoll()` method, copy these two files and put them in their own folder (named CrapsGameTest?) Do not put any other files in this folder.



Coding Part 2: Die.java and DieTester.java

Do NOT move forward before you have completed each step:

1. **Step 1:** - Write the class named Die FROM SCRATCH so that the other files can use it as intended. The Die class simulates ONE 6-faced number cube.

   A Die object knows:

   The value it shows after it is rolled (an integer private data member)

   A Die object has two behaviors:

   `public void roll()` (the method that will generate a random number between 1 and 6, inclusive; this value should be assigned to

the private data member)
`public int getNumDots()` (this accessor method returns the value of the private data member)

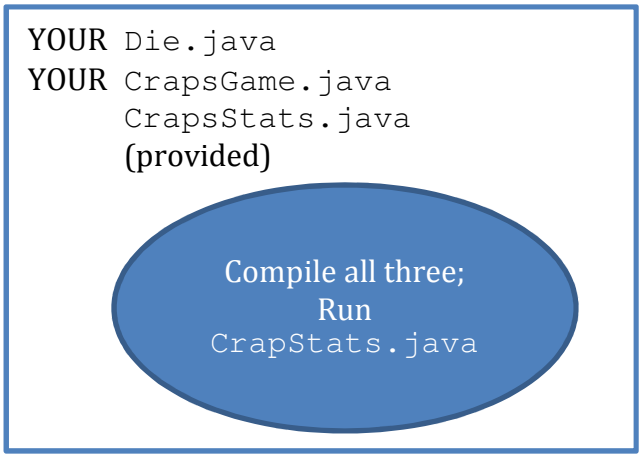
Note that the method signatures must be exactly as written above because the files that will use them call them in that manner.

2. **Step 2** – Write a tester for your Die class, also FROM SCRATCH. Make sure the methods work as you intend. You willnot turn this tester in, but it is a vital step in knowing your code is working correctly.

3. **Step 3** - Test your Die clas with a different pre-made test harness. CrapsStats allows you to test your implementation of the rules quickly and easily without needing to worry about a GUI. Follow the directions below. Compare your simulation result with the theoretical probability of winning in *Craps,* which is 244/495, or about 0.493. *If you run 10,000 trial games, the number of wins should be somewhere between 4830 and 5030.*

Test Die.java

a. To test the `Die.java` class, you were first supposed to create the `Die` class from the scratch, create `DieTester.java` from scratch using the method identifiers written on page 25 of the instructions. Once you have completed this, you are ready to test it with `CrapStats.java`.
b. Test `Die.java` with `CrapStats.java`: Copy the files listed below and put them in another new folder (named DieTest?) Do not put any other files in this folder.



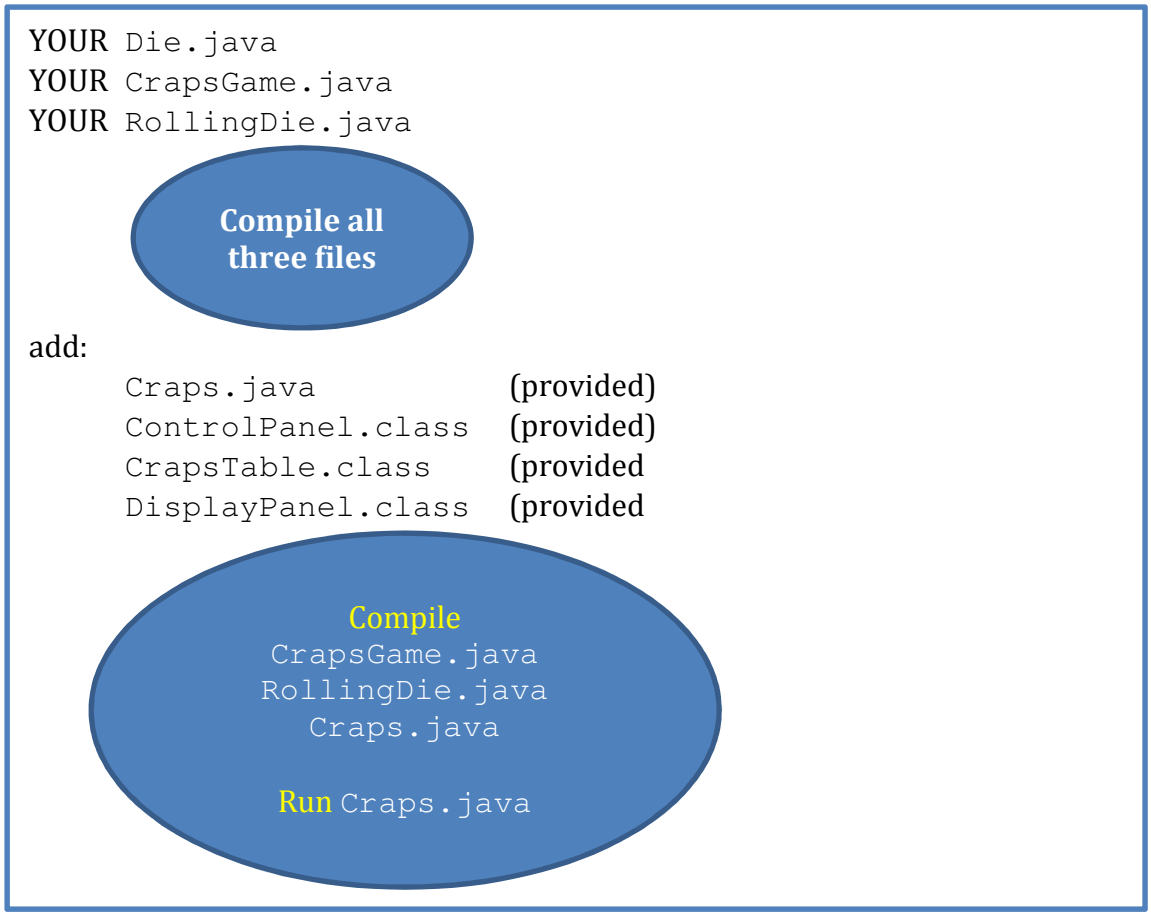
Coding Part 3: complete the drawDots() method in the RollingDie.java class

Read the information again in the RollingDiceLab.pdf document concerning the RollingDie class and how it draws the dots on the Die. Again, there may be some things you don't understand. Don't get bogged down with them! Read through the text until you get to section 7.13 (you don't have to read that part!).

1. **Carefully** review the code in the `drawDots` method of RollingDie.
    a. What do the `x1`, `x2` & `x3` variables represent in the context of a die? What about the `y1`, `y2` & `y3` variables?
    b. The case to draw a single dot representing '1' is shown. Use this as the framework to draw the dots for 2, 3, 4, 5, & 6.
    c. Feel free to be creative with your dot color and even the placement of dots on your Die object.

Test `RollingDie.java`

To test your graphics implemented in the `drawDots()` method, copy the files listed below and put them in another new folder (named DieDotsTest?) Do not put any other files in this folder.



What to turn in

When you are done with this lab upload the source code for the following files:

CrapsGame.java
Die.java
RollingDie.java

to the Programming Lab #3: RollingDice assignment in Canvas. Be sure to review both this document and the assignment rubric for a clear understanding of how the assignment will be graded.