Lab 5: Oggetti Casuali

Objectives

- Practice making and using classes and objects in Java

Preliminary Setup

For this lab, you will be creating an object-oriented (OO) program to play a simple dice game in Java. Start with the following:

1. Find and launch IntelliJ
2. Create an IntelliJ project.
3. In the created src folder, right click to add:
    a. a **Main** class with a **public static void main** method
    b. a Dice class[1]

    Note that to create a class in IntelliJ, right click on the **src** folder in the project, and select **New > Java Class**.

Step I: Write the Dice class

Write the **Dice** class. We are going to be modeling dice with different numbers of sides. Therefore, a Dice object should have two instance variables: the number of sides the dice has and the current value showing on the dice. Make sure the instance variables are declared private.

Add the following methods:

- A constructor that takes an `int` parameter that indicates the number of sides that the particular dice should have. Be sure that the constructor gives each of the instance variables an appropriate value.

1 I've checked, and "Dice" is an acceptable singular form. If this bothers you because you consider "Dice" as **plural** only, then you should instead create a "Die" class. Classes should be named with singular nouns.

    - **NOTE**: In general, a constructor should be sure to assign values to each of the attributes/instance variables, even if a given instance variable is not passed as a parameter to the constructor. In this case, the current value of the dice is not passed as a parameter, but should be assigned by the constructor. What's a good value to assign this attribute?
- A **`roll()`** method that rolls the dice (causes the current value to be a new random number from 1 to the number of sides).
    - Note: you can use the **Random** class [docs here]. In particular, the **nextInt** method might be of use. You'll need to import the **Random** class with **import java.util.Random;** because it is not in your own package.
    - Note: **roll** is a setter method -- it changes the instance variables of the given Dice object. It doesn't need to return anything -- it just changes the instance variable(s).
- A **`getValue()`** method that returns the value currently showing on the dice.
    - Such a method is called a getter method.

Step II: Write main

In the **Main** class, write the **main()** method to play a simple game according to the following rules:

1. The game uses a D6 and a D12 (6-sided and 12-sided dice).
2. On each turn, we roll both dice.
3. The game is won if one of the dice shows a value that's exactly twice the value of the other.
    a. Note: we win if the D12 shows twice the D6 OR if the D6 shows twice the D12.

Your **main** method should create variables for the two dice: each should be an object of type **Dice**.

Your program should ask the user to press the return key before each turn. This can be accomplished with a **Scanner** object [docs here]. **Scanner** can be imported with **import java.util.Scanner;**. For example, you can create a **Scanner** like this:

```
Scanner input = new Scanner(System.in);
```

Once you have the scanner initialized like this, you can ask the user for input like this:

```
System.out.println("Press enter");
String line = input.nextLine();
```

When the user presses return, the program should roll the dice and print out their new values. After each turn, the program should check if we've won. If we've won, print a message saying so and end the program.

Step III: Add a default [Extra Credit]

Now, go back and modify the **Dice** class so that the default number of sides is 6, so that if we ask to create a Dice object without specifying the number of sides, we will get a D6. **We still want to be able to create other sizes** of Dice, we just want the default size to be 6.

Modify **Main** so it uses this default when we want a D6, and specifies a non-default size when we want a D12.

How to turn in this lab

Before turning in any program in this class, remember this mantra:

Just because it works doesn't mean it's good.

Part of your grade will also come from things like how understandable and readable your code is. You will also be graded on the neatness, presentation, and style of your program code.

Turn in the lab via gradescope by 4pm the day after lab.

Ask for help if you're having problems!