**Programming on Purpose (CSC 120)**
**Winter 2024**

# Lab 6 – Die with Class (in Java)

**2/8/2024**

## 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. Start with the following:

1. There will be no starter code.
2. Create a folder for this lab. Then open the folder in VS Code.
3. Create the file `DiceGame.java`. This is where you will define the `DiceGame` class, which will contain the `public static void main(String[] args)` method.
4. Create the file `Die.java`. This is where you will define the `Die` class.

## 1 Step 1: Write the Die class

In `Die.java`, write the `Die` class. A `Die` object should have two properties/attributes: the number of sides the dice has, and the current value showing on the dice. These properties should be *instance variables* in your class. Make sure you declare the instance variables private.

Add the following methods. Write Javadocs *before you write the code.*

- The constructor takes one parameter that indicates the number of sides that the particular die should have. Be sure that *each* of the instance variables is given an appropriate value.
- The *roll* method rolls the die (causes the current value to be a new random number from 1 to the number of sides).

  Note: you can use the `Random` class (see the documentation 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. And then you have to create a `Random` object so that you can call methods on this object. (Also see the `Boneyard` implementation from class for an example that uses `java.util.Random`.)
- A *getter* method should return the value currently showing on the die.

  Note the conventional name for a getter method for some instance variable named `instVar` is `getInstVar`.

## 2 Step 2: Write `DiceGame`

In the `DiceGame` 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. Show the user the result of the dice rolls.
3. The game is won if one of the dice shows a value that's exactly twice the value of the other. Otherwise, roll again.

Your `main` function 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 (documentation 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();
```

After each turn, the program should check if we've won. If we've won, print a message saying so and end the program.

*While-loops* in Java work the same way as while-loops in Python, but there are a few syntactic differences:

- The condition needs to be in between parentheses.
- The block of code representing the loop body is between curly braces. So:

  ```
  while (condition) {
      ...
  }
  ```

- The logical operators *not*, *and*, and *or* are written as follows in Java: `!`, `&&`, and `||`.

## 3 Step 3: Add a default

In Python, we could give default values to parameters (to constructors or any other methods). Java doesn't have optional parameters, but it does have something Python doesn't. In Python, you cannot have two methods in the same class with the same name. But in Java, you can! It's called *method overloading*, which means two methods can have the same name as long as the parameter lists differ. That gives us a way of accomplishing the same default behavior that the Python version had. Write a second constructor, also called `Die`, but this time make it take zero parameters (that is, make it a default constructor). The constructor should still initialize the two instance variables, but this time, the number of sides should be set to 6. That way, if someone writes:

```
Die d = new Die();
```

they'll get a d6. But if the user does supply an argument, it will run the first constructor you made and initialize the number of sides to the given argument.

Modify the `main` method so it uses these constructors as appropriate.

## 4 Revise your code

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.

Make sure that all modules and functions are documented (even those that you didn't write).

Don't forget to cite who you received help from and include the honor code affirmation in the docstring of each module that you wrote or modified:

```
I affirm that I have carried out my academic endeavors
with full academic honesty. [Your Name]
```

## 5 How to submit

Submit the project by uploading `Die.java` and `DiceGame.java` to Gradescope.

Make sure to add your partner to the submission so that you both have access to it through Gradescope.