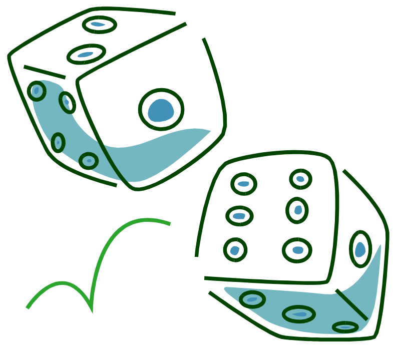
Java Programming – TwoDice

Review the **Die** class that was discussed in Chapter 5.



```
public class Die
{
  private final int MAX = 6;  // maximum face value
  private int faceValue;     // current value showing on the die

  //-----------------------------------------------------------------
  //  Constructor: Sets the initial face value of this die.
  //-----------------------------------------------------------------
  public Die()
  {
    faceValue = 1;
  }

  //-----------------------------------------------------------------
  //  Computes a new face value for this die and returns the result.
             //  This simulates rolling the die
  //-----------------------------------------------------------------
  public int roll()
  {
    faceValue = (int)(Math.random() * MAX) + 1;
    return faceValue;
  }

  //-----------------------------------------------------------------
  //  Face value mutator. The face value is not modified if the
  //  specified value is not valid.
  //-----------------------------------------------------------------
  public void setFaceValue (int value)
  {
    if (value > 0 && value <= MAX)
      faceValue = value;
  }

  //-----------------------------------------------------------------
  //  Face value accessor.
  //-----------------------------------------------------------------
  public int getFaceValue()
  {
    return faceValue;
  }

  //-----------------------------------------------------------------
  //  Returns a string representation of this die.
  //-----------------------------------------------------------------
  public String toString()
  {
    String result = Integer.toString(faceValue);

    return result;
  }
}
```

Once the **Die** (or any) class is created, it can be used by any number of Java programs: (1) drivers and (2) and other classes. Look at the following 2 lines of code:

```
Die myDie = new Die();
myDie.roll() ;
```

The 1st blue statement above creates a new **Die** object called **myDie**. Look at the constructor for **Die** (the red text above). You will see each new **Die** object is created with a value of 1 (as if a 1 is rolled).

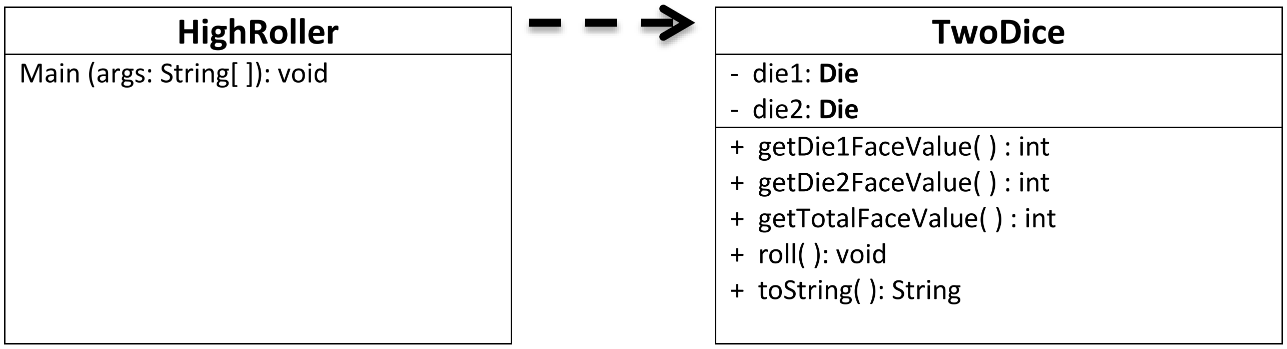
The 2nd blue statement above uses the **Die** object just created (called **myDie**) and rolls the die. Specifically, it calls the **roll ( )** method from the **Die** class which randomly updates the **faceValue** of **myDie** (like rolling the die).

You could probably think of many games which use a set of dice which, when rolled, are always rolled together. For programs which use 2 dice, you could create a new class which has 2 instance data items which are **Die** objects.

To create the new class, you will define 2 **Die** objects and take advantage of the methods already defined in the **Die** class. By doing this, our job is easier and we are not duplicating code written in **Die**. **In other words, we can use the Die class to create a new class of two dice.**

TwoDice Class

For your homework this week, you will create a new class (similar to how you created the **Horse** class in Week 7). Look at the UML Class diagram below for a new class used to represent a pair of die.


HighRoller
Main (args: String[ ]): void
TwoDice
- die1: Die
- die2: Die
+ getDie1FaceValue( ) : int
+ getDie2FaceValue( ) : int
+ getTotalFaceValue( ) : int
+ roll( ): void
+ toString( ): String

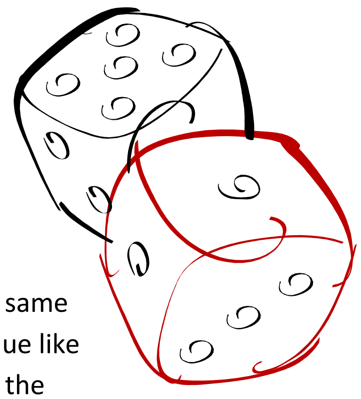
(+ represents the public modifier and – represents the private modifier)

Look at the UML diagram carefully. You are to create the methods above plus a constructor for TwoDice.

TwoDice Class

Create a new class called **TwoDice**

- **TwoDice** will have two instance data items: die1 and die2. **Each is an object from the Die class.** Notice die1 and die2 are not int values—they are Die objects. In a very loose sense, you are using the **Die** class as a data type. The constructor for TwoDice will instantiate 2 Die objects. **Remember: the face value of a Die object when instantiated is 1.**
- **getDie1FaceValue( )** : Use the getFaceValue( ) method from the Die class to retrieve and return the value of die1. This method is available in case you need to know the value of only die1. If you have an object named die1 then **die1.getFaceValue()** will retrieve it's face value.
- **getDie2FaceValue( )** : Use the getFaceValue( ) method from the Die class to retrieve and return the value of die2. This method is available in case you need to know the value of only die2 and work the same as **getDie1FaceValue().**
- **getTotalFaceValue( )** : Use the appropriate method(s) from the **Die** class to retrieve the values of die1 and die2, add those values together, and then return the total.
- **roll( ):** Use the appropriate method(s) from the Die class to roll die1 and die2. If you have a Die object named die1 then **die1.roll()** will roll that one dice. You can do the same for die2. This simulates rolling both dice. NOTE: This method does not return a value like the Die class. It simply updates the face value for die1 and die2. HighRoller can use the accessors for die1 and die2 to retrieve the new values after "rolling".
- **toString( ):** Use the toString( ) method from the Die class and modify as needed to display the two dice in a pleasing manner



HighRoller Class (Driver)

You can find a copy of the **Die** class in the textbook files you downloaded at the beginning of the semester.

Create a new class called **HighRoller**. This is a driver program which simulates a game which uses the **TwoDice** class.

NOTE: **TwoDice** uses the **Die** class. **HighRoller** uses **TwoDice**. However, **HighRoller** does NOT NEED to use the **Die** class or any of its methods directly. It will use methods from TwoDice

Guidelines for the **HighRoller** game:

- The user competes against the computer (You will track two totals).
- The user rolls first. Total the points on the die. Display the total.
- The computer rolls. Total the points. Display the total.
- Determine if the user won, the computer won, or there was a tie.
- Allow the user to play another round. After each round played, display the number of times the user won, the number of times the computer won, and the number of ties.

This is a simple game. The purpose is to get more practice creating and using classes.

To test your application:

- Compile **Die**
- Compile **TwoDice**
- Compile and run **HighRoller**

Purpose: Understand how classes can be used together.

Zip Die.java, TwoDice.java, and HighRoller.java and submit for grading. Make sure you submit all 3 java files.