All programming languages come with a large amount of useful pre-written code, often called the language's *standard library*. It's good to use such code because it saves you time, is well debugged and is efficient.

In Java there are two types of built-in prewritten library code: static utility methods (like the Math methods) and methods in built-in Java classes (like String and Random). Let's explore both types.

**Static utility methods**

Search the internet for "java 11 math"[1]. The first result should be the Javadoc for the Math class. In it you will see a large list of methods marked as static. These are all available in other classes by simply calling Math.foo(params) where foo is the name of the method you want.

1) Find the static utility methods in the Integer class. Note that some methods are not marked as static. Ignore those for now and just look over the static ones. Write one or two lines of code that would convert the number 62 into a binary string and print it. Be sure to prefix your Integer static method call with "Integer.".

2) Write one or two lines of code that would convert the binary string "111110" into an int and print it.

3) Write one or two lines of code that would convert the binary string "111110" into a hexadecimal string and print it.

Lots of built in classes in Java have static utility methods. Before you write your own method to do something of general utility, it may be worth investigating to see if a related Java class has already done it for you.

**Methods in built-in Java objects**

The static utility methods have something in common: they don't need to remember anything between uses. For example there's no reason the Math class needs to remember anything between the calls Math.sqrt(10) and Math.sqrt(20). This makes it most efficient to provide these as static methods.

---

[1] I search for "Java 11 Math" when looking for the Math class documentation because I have Java version 11 on my computer. If you know your Java version, you can use that instead of 11.

On the other hand, if a utility method needs to remember things between calls, the data it remembers is stored in an object, and to access the utility functions associated with that stored data you need to request that the object invoke the method you want.

For example, the Random class defines several methods for getting random values of various types. The Random class does its work by keeping track of a long sequence of random values and where in the sequence it currently is. So, to use the Random utility functions, you must create a Random object and have it give you the values.

```
Random rand = new Random();    // create Random object
int a = rand.nextInt(10);      // random value in 0...9
double b = rand.nextDouble(); // random value in [0,1)
```

Tip: rand.nextInt(n) returns one of n possible integers 0...n-1. To change the minimum value, add it to the result: m + rand.nextInt(n) generates one of n different integers, the smallest one being m.

4) Read the Javadoc for Random to see what kinds of random values you can create. Ignore the methods that return Streams.

5) Assuming rand has already been initialized as above, write one or two lines of code that generates a random die roll and prints it out.

6) Assuming rand has already been initialized as above, write one or two lines of code that generates a random pair of rolled dice and prints out their sum.

7) Write a complete program that simulates the rolling of a pair of dice 100,000 times and outputs with what frequency a particular dice sum occurs. Define as a class constant DICE_VALUE the dice sum you are collecting statistics for. For example, if DICE_VALUE is 3, your program should output a number near to 0.056 (ie, 2/36). To make the Random class available in your program include the following line at the top of your program file.

   ```
   import java.util.Random;
   ```