Method Overloading

- Method names **do not have to be unique** as long as the parameters re different for methods with the same name
- *Method overloading*: when more than one method of the same name is included in a class
- The compiler uses the types, order, and number of parameters to determine which method to execute

Example: (includes **two** drawBar() methods)

```
Public class MethodOverloadingExample {

      public static void drawBar(int length) {

            for (int i = 1; i <= length; i++) {
                  System.out.print("*");
            }
            System.out.println();
      }

      public static void drawBar(int length, String mark) {

            for (int i = 1; i <= length; i++) {
                  System.out.print(mark);
            }
            System.out.println();
      }

      public static void main(String[] args) {

            drawBar(10);            //one parameter
            drawBar(5, "0");        //two parameters
      }
}
```

Output:

```
**********
00000
```

ICS4U Module 3: Note & Exercise 2c

Documenting Methods

- Methods should be commented in a specific way
- The reader should understand what task the method is performing and what data, if any will be returned
- Use comments of the form /** */ above the method declaration
- The comment block should contain three parts:
    - Description of what the method does
    - Precondition: "pre:"
        - The assumptions, or initial requirements
        - Should not include data types (compiler does this part)
    - Postcondition: "post:"
        - States what must be true after the method has been executed
        - Should not state **how** the method accomplished its work!

Example:

```
/**
 * Print a bar of asterisks across the screen
 * pre: length > 0
 * post: Bar drawn of length characters, insertion point moved to next line
 */

public static void drawBar(int length) {

            for (int i = 1; i <= length; i++) {
                  System.out.print("*");
            }
            System.out.println();
      }
```

Output:

```
*
*
*
*
*
*
*
*
*
*

o
o
o
o
o
```

Programming Exercise:

Create an DrawLine application similar to the DrawBar example provided. Overload your drawLine() method to have two versions:

`drawLine(int length)` and `drawLine(int length, String mark)`

Output should look similar to the sample provided on the right.

Submit your source code to the Google Doc "ICS4U – Activity Submission Form"