Kaitlin Hoffmann

Office Hours:

SH 243 MR 11:00 - 12:30 PM via appointment https://calendly.com/hoffmank4/15min

Email: hoffmank4@newpaltz.edu

For TA Office Hours and Email – Please see syllabus

NEW! Supplemental Instruction: Sign up at my.newpaltz.edu

METHODS

COMPUTER SCIENCE I

OBJECTIVES

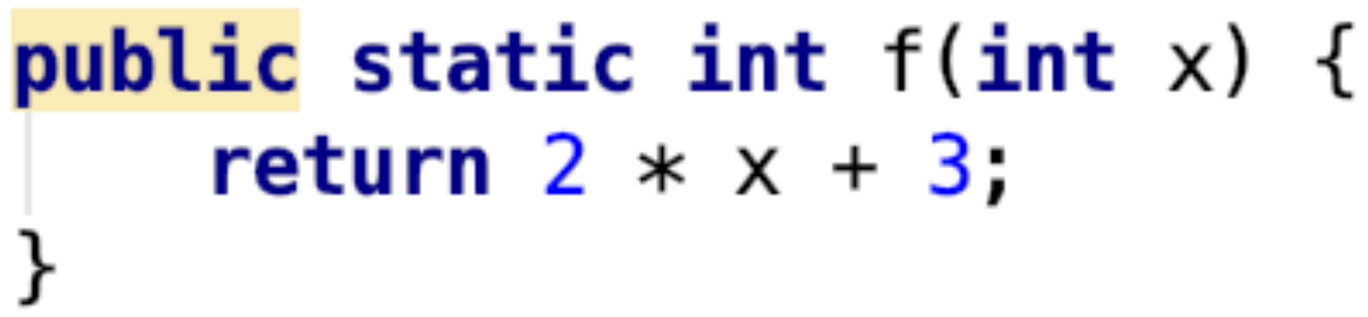
- Methods
- Method Scope
- Method Overloading

METHODS

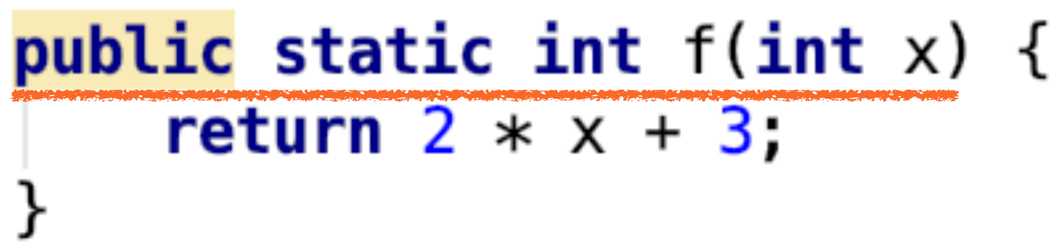
- In programming, there are many times where we may want to reuse the same piece of code throughout a complex program. For now, we would have to copy and paste it.
- However, with methods, we are able to create a piece of code and reuse it over and over by simply calling the method in our program.

- A **method** is like a function mathematics. We can give methods inputs, and they can return to us some outputs.
- **Ex:** $f(x) = 2x + 3$
  - ✦ What is **f(0)**?
  - ✦ What is **f(4)**?
  - ✦ What is **f(-2)**?
- Here f is the **function name**, x is an **input value**, and the answer or **return** is the value of 2x+3.

- To convert this function, **f(x) = 2x + 3**, as a method in Java:



```java
public static int f(int x) {
    return 2 * x + 3;
}
```



```
public static int f(int x) {
    return 2 * x + 3;
}
```

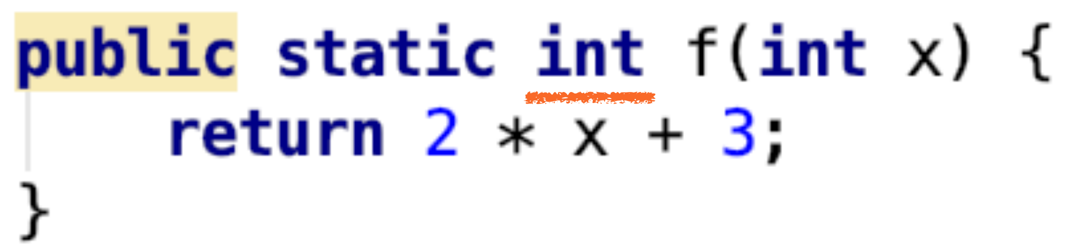
METHOD COMPONENTS

- A **method header** has the format:

  public static returnType methodName(inputs)

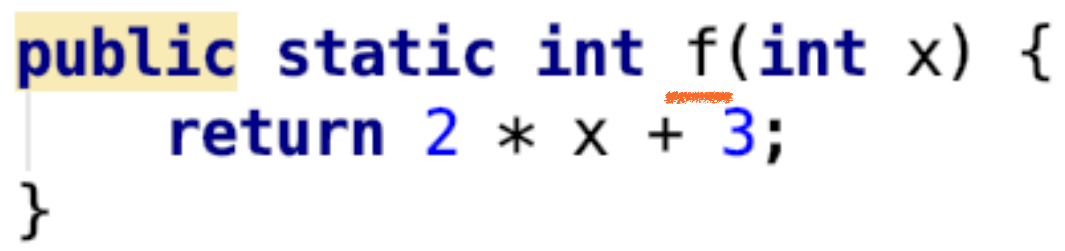
- The method header is the first line of the method.
- Methods can be public or private
- Methods in the same class as the main method have *static* in their method header.
  - *static* means there is only one copy of the method and is available at the class level. This idea will become clear later, and we'll go into more detail then.

RETURN TYPES



```
public static int f(int x) {
    return 2 * x + 3;
}
```

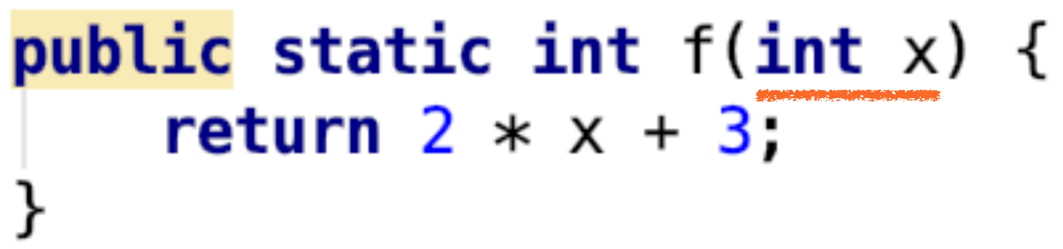
- Every method has exactly **one** return type.
- Return types:
  - int, double, boolean, char, . . . the Java primitive types
  - int[ ], double[ ], . . . Arrays of any type (we'll learn about these soon)
  - Objects (we'll learn about these later)
  - void - void means there is **no** return type.
- In the example above, the return type is an **int**.



```
public static int f(int x) {
    return 2 * x + 3;
}
```

METHOD NAMES

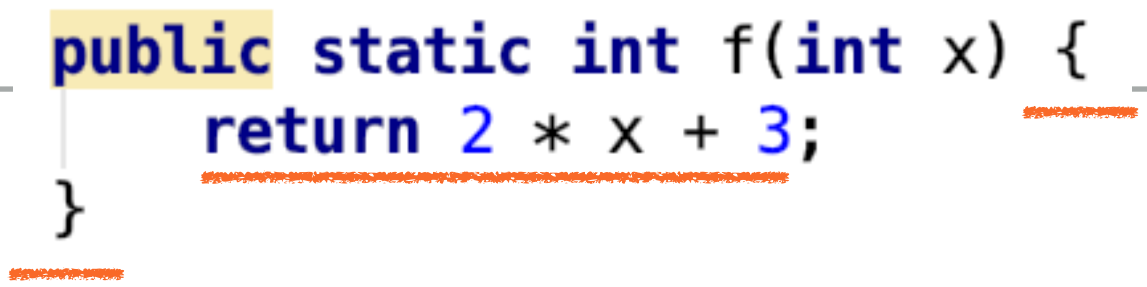
- Method names can be anything just like a variable, however they should tell you something about the purpose of the method.
- Just like with variables, method names should be camel case. **Examples:**
  - getEmployee
  - setSalary
  - printPattern



```
public static int f(int x) {
    return 2 * x + 3;
}
```

PARAMETERS

- Method inputs are called parameters and arguments in computer science.
- A method can have **no parameters** or **as many as needed**.
- Parameters are given as the type followed by a variable.
    - Parameters, like return types, can be primitive, arrays, objects, nothing, a mixture of everything, etc.
- In the example above, the parameter is **int x**



```
public static int f(int x) {
    return 2 * x + 3;
}
```

SYNTAX

- After the method header, there is a set of { } and then any code you want.
- Method bodies can have any code we have learned so far inside them.
- All of our lab problems can be placed into a method. This is good practice!

MAIN METHOD

- Guess what the main header is in all of our programs? A method! It's called the **main method**.

  public static void main(String[ ] args)

  - What is the **name** of this method?
  - What is the **return type** of the main method?
  - What is the **parameter**?

MAIN METHOD

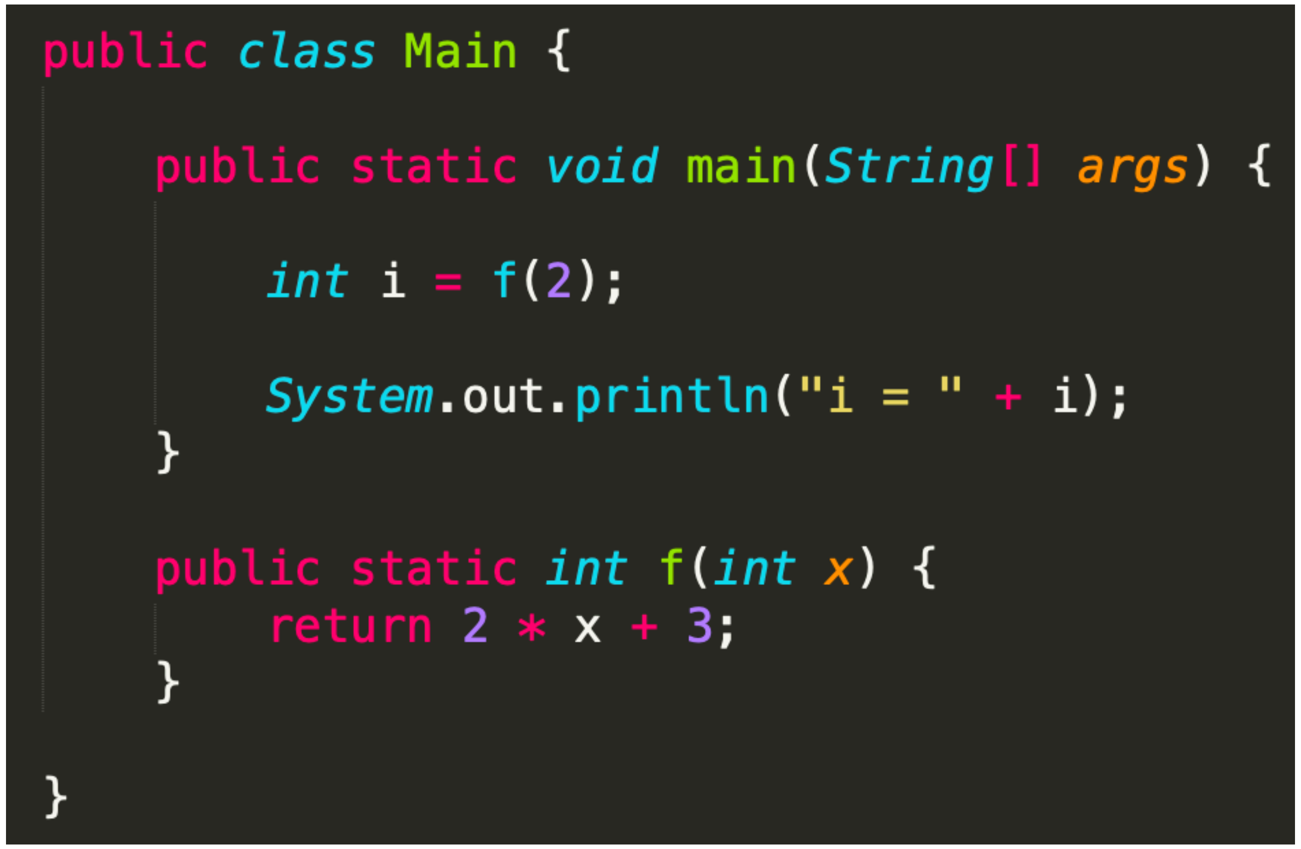
- Guess what the main header is in all of our programs? A method! It's called the **main method**.

  public static void main(String[ ] args)

  - What is the **name** of this method? **main**
  - What is the **return type** of the main method? **void**
  - What is the **parameter**? **string[] args**

HOW DO WE USE A METHOD?

1. For static methods, we write our methods outside of the main method but inside the class header (right before the last curly bracket).
2. Then, we **call** our method inside our main method using its name (just like when using a variable).
3. If there is a return type, **assign** the method to its return type. Example:
   - **int num = f(2);**
4. If the return type is void, we just call the method **as is**:
   - **printGreeting("Hello!");**



```java
public class Main {

    public static void main(String[] args) {

        int i = f(2);

        System.out.println("i = " + i);
    }

    public static int f(int x) {
        return 2 * x + 3;
    }

}
```

Output

```
i = 7
```

```
public static int f(int x) {
    return 2 * x + 3;
}
```

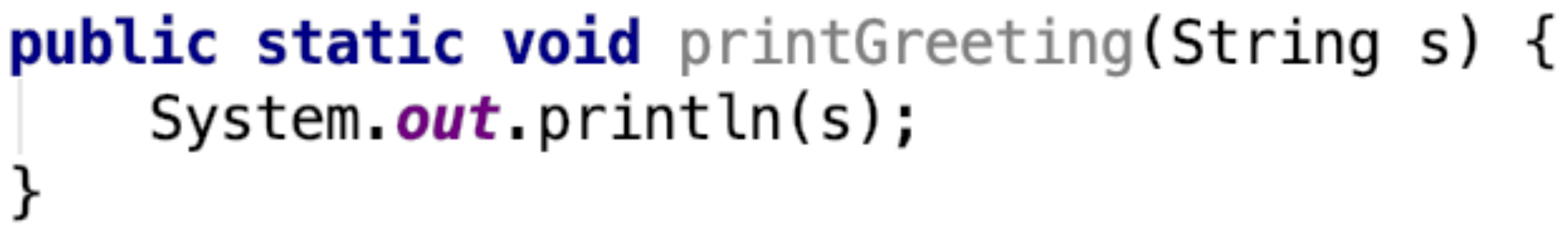
RETURNING

- **return** means "give this value back to who called me".
- The return **must** match the return type indicated on the method after *public*.
  - For example, the return type above is an **int** so we must return an int – which we are! 2 * x + 3 will produce an int.
- If the return type is **void**, we **DO NOT** have a return.
  - **Ex:**

```
public static void printGreeting(String s) {
    System.out.println(s);
}
```

WHEN DO WE USE VOID?

- For now, you should use void when you are simply **printing**. We'll learn about when to use void when we learn about arrays and references.
- However, if you are ever in doubt, just have a return type. With more practice and experience, you'll understand when void is used.



```
public static void printGreeting(String s) {
    System.out.println(s);
}
```

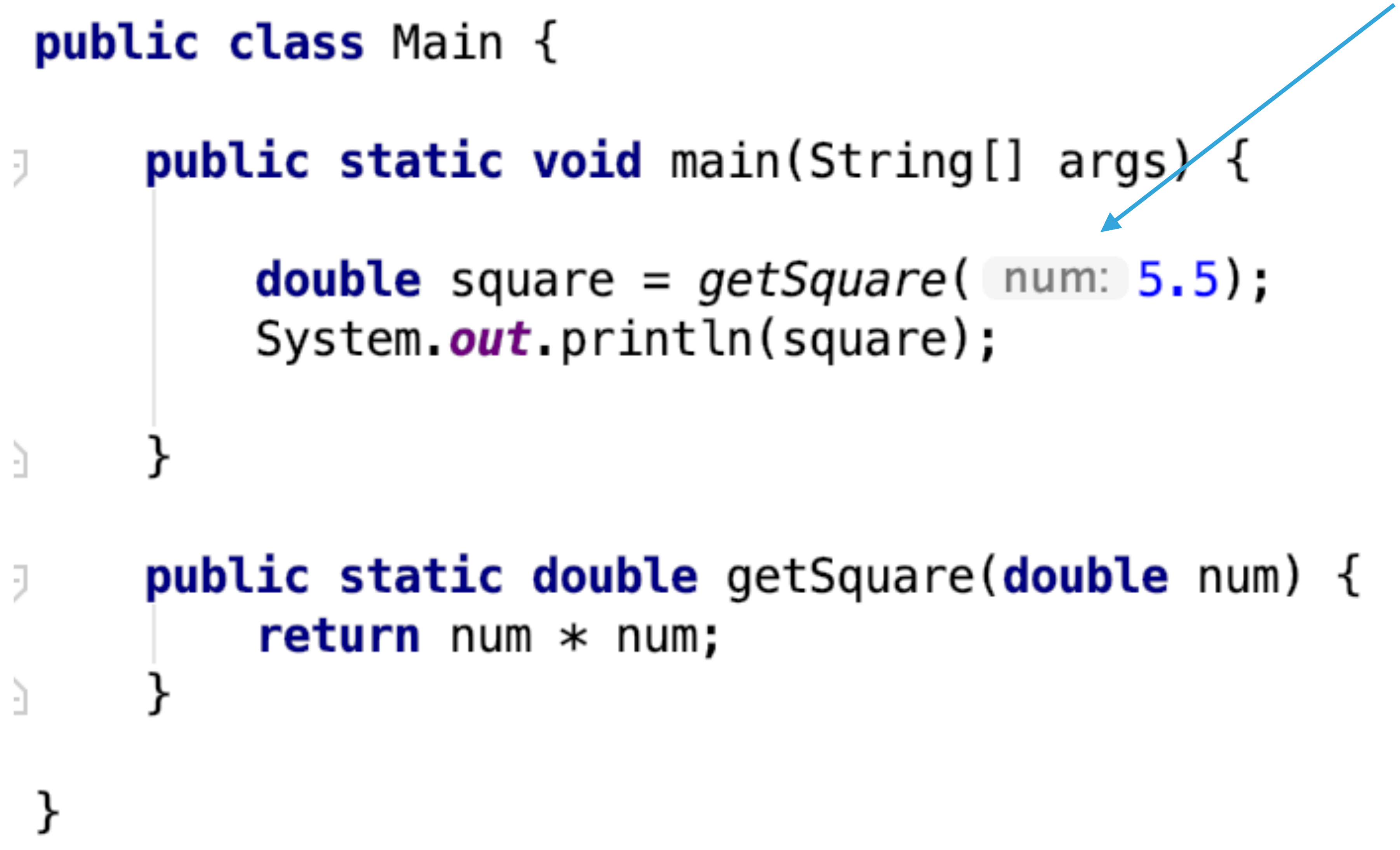
EXAMPLE 1:

- Write a method to compute and return *n* raised to the second power. Let's say *n* may be an integer or double that a user enters.

EXAMPLE 1:

- Write a method to compute and return *n* raised to the second power.

***Ignore** the **num:** This is my text editor. I am not writing this and nor should you!



```
public class Main {

    public static void main(String[] args) {

        double square = getSquare( num: 5.5);
        System.out.println(square);

    }

    public static double getSquare(double num) {
        return num * num;
    }

}
```

Output

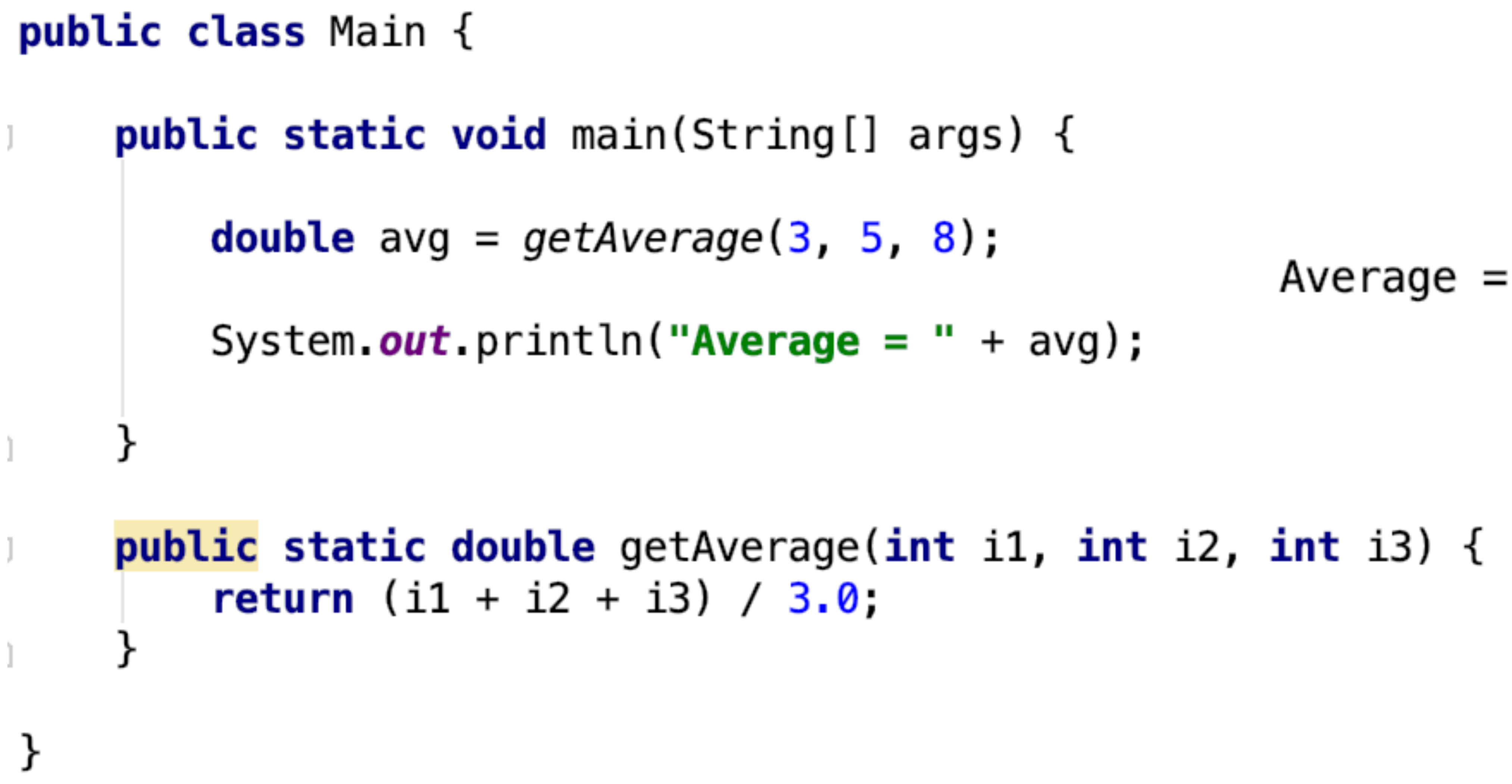
```
30.25
```

EXAMPLE 2:

- Write a method that finds the average of 3 integers.
  1. What should my parameters be?
  2. What should my return type be (what am I returning)?
- If you are ever confused on how to put a program in a method, first solve it like we have been doing, then place it in the method. Overtime, creating methods will become easier and second nature.

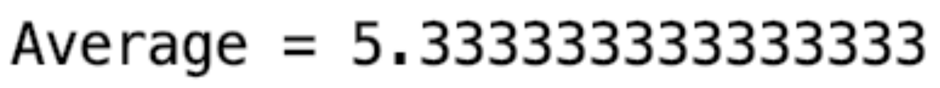
EXAMPLE 2:

- Write a method that finds the average of 3 integers.
  1. What should my parameters be? **3 integers**
  2. What should my return type be? **double**



```java
public class Main {

    public static void main(String[] args) {

        double avg = getAverage(3, 5, 8);

        System.out.println("Average = " + avg);

    }

    public static double getAverage(int i1, int i2, int i3) {
        return (i1 + i2 + i3) / 3.0;
    }

}
```

Output



```
Average = 5.333333333333333
```

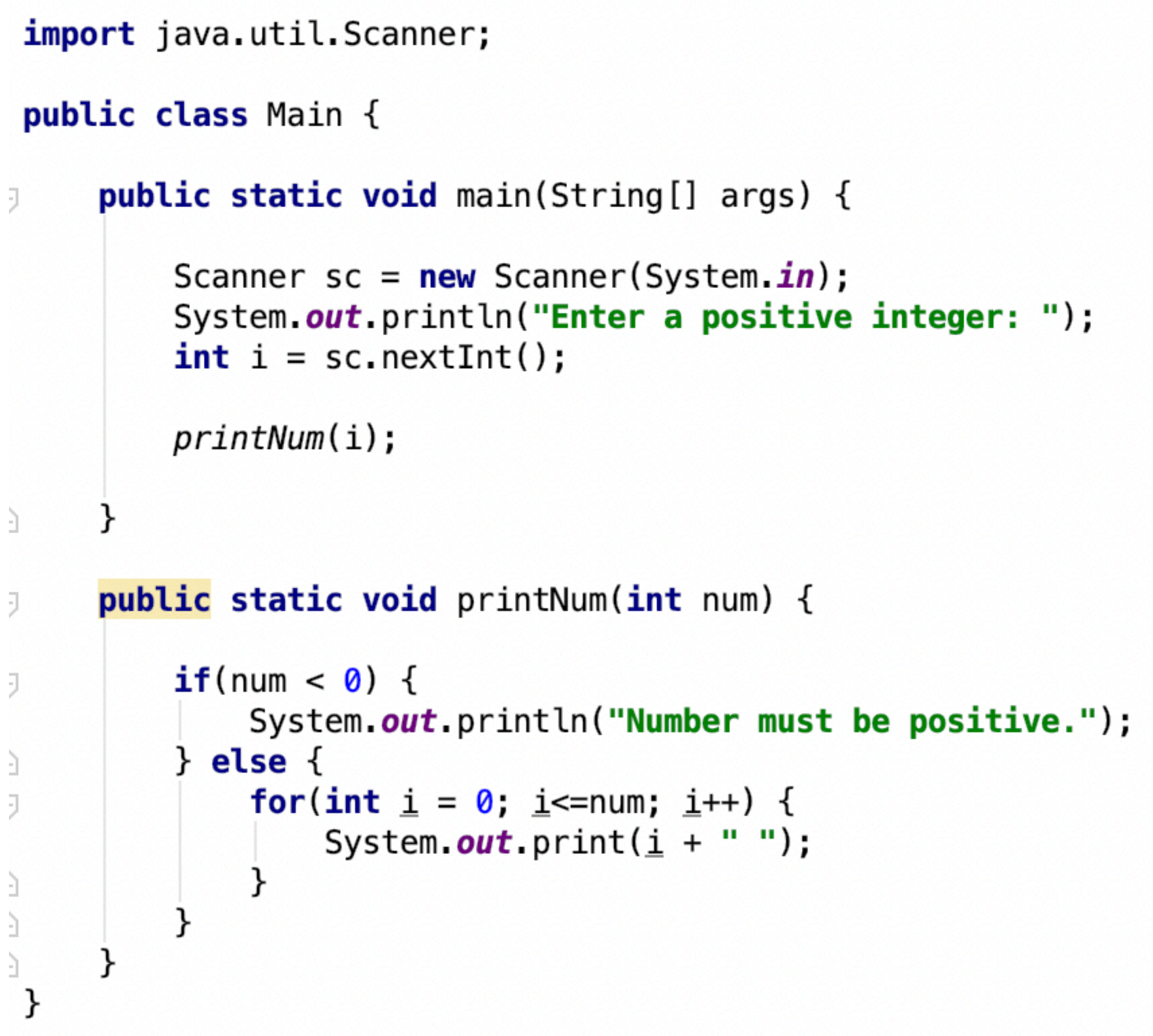
EXAMPLE 3:

- Write a method that prints 0 to a positive integer inclusive.
    1. What should my parameters be?
    2. What should my return type be (what am I returning)?

- Write a method that prints 0 to a positive integer inclusive.

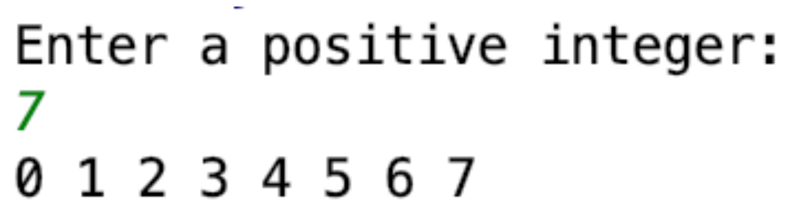
EXAMPLE 3:

1. What should my parameters be? **1 integer**
2. What should my return type be (what am I returning)? **void since I'm printing**



```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a positive integer: ");
        int i = sc.nextInt();

        printNum(i);

    }

    public static void printNum(int num) {

        if(num < 0) {
            System.out.println("Number must be positive.");
        } else {
            for(int i = 0; i<=num; i++) {
                System.out.print(i + " ");
            }
        }
    }
}
```

Output



```
Enter a positive integer:
7
0 1 2 3 4 5 6 7
```

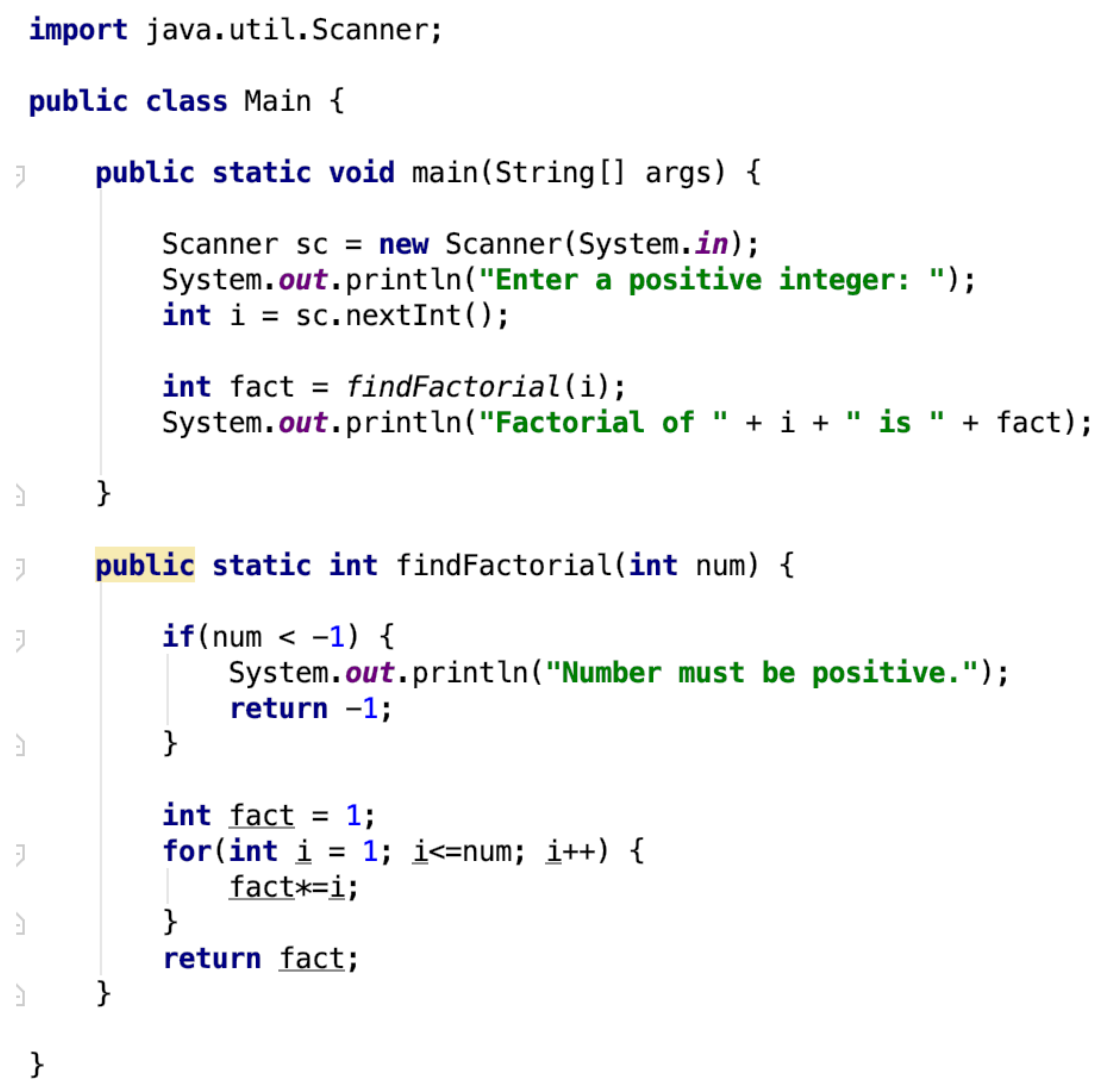
EXAMPLE 4:

- Write a method that finds the factorial of a number.
    1. What should my parameters be?
    2. What should my return type be (what am I returning)?

EXAMPLE 4:

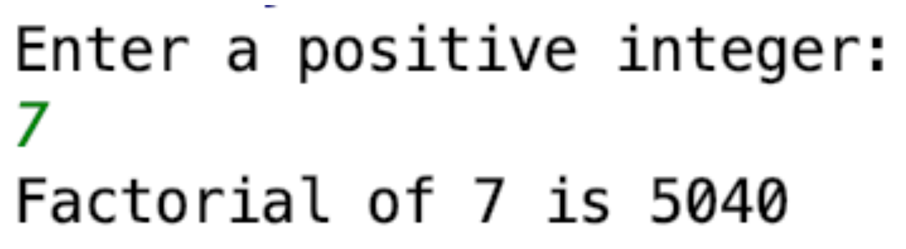
- Write a method that finds the factorial of a number.

1. What should my parameters be? **int for the number I am finding the factorial of**
2. What should my return type be (what am I returning)? **int since I'm returning the factorial**



```java
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a positive integer: ");
        int i = sc.nextInt();

        int fact = findFactorial(i);
        System.out.println("Factorial of " + i + " is " + fact);

    }

    public static int findFactorial(int num) {

        if(num < -1) {
            System.out.println("Number must be positive.");
            return -1;
        }

        int fact = 1;
        for(int i = 1; i<=num; i++) {
            fact*=i;
        }
        return fact;
    }

}
```

Output



```
Enter a positive integer:
7
Factorial of 7 is 5040
```

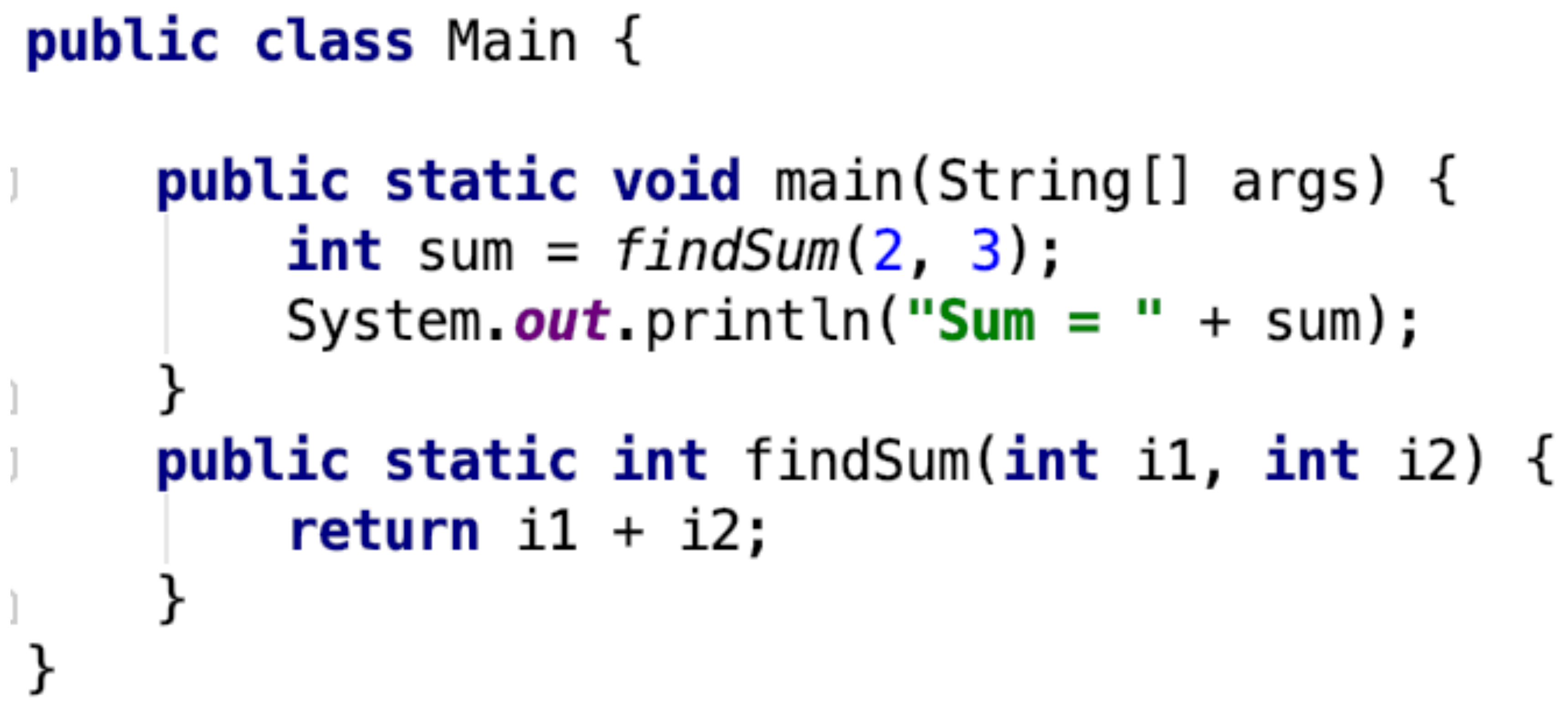
EXERCISE 1:

- Write a method that finds the sum of two integers.

    1. What should my parameters be?
    2. What should my return type be (what am I returning)?

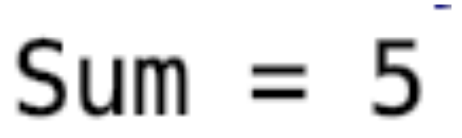
EXERCISE 1:

- Write a method that finds the sum of two integers.
  1. What should my parameters be? **Two integers**
  2. What should my return type be (what am I returning)? **int for the sum**



```
public class Main {

    public static void main(String[] args) {
        int sum = findSum(2, 3);
        System.out.println("Sum = " + sum);
    }
    public static int findSum(int i1, int i2) {
        return i1 + i2;
    }
}
```

Output



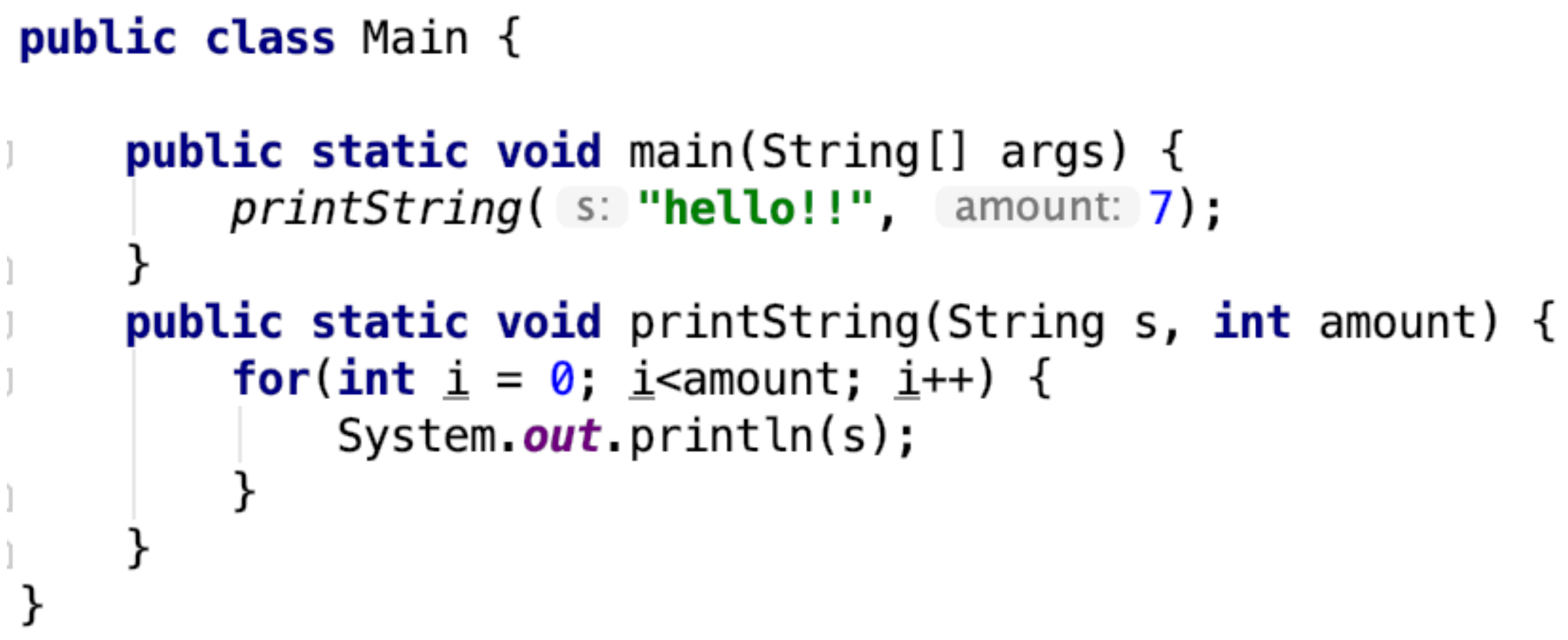
```
Sum = 5
```

EXERCISE 2:

- Write a method that prints a String x amount of times.
  1. What should my parameters be?
  2. What should my return type be (what am I returning)?

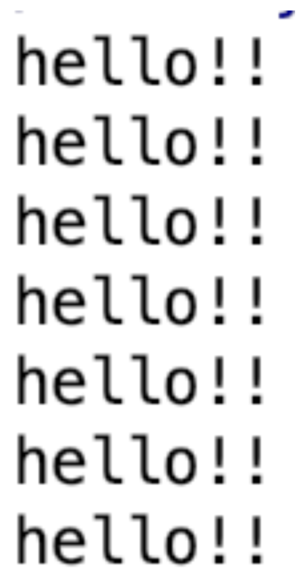
EXERCISE 2:

- Write a method that prints a String 15 times.
  1. What should my parameters be? **1 String and 1 int**
  2. What should my return type be (what am I returning)? **void since we are printing**



```java
public class Main {

    public static void main(String[] args) {
        printString( s: "hello!!",  amount: 7);
    }
    public static void printString(String s, int amount) {
        for(int i = 0; i<amount; i++) {
            System.out.println(s);
        }
    }
}
```

Output

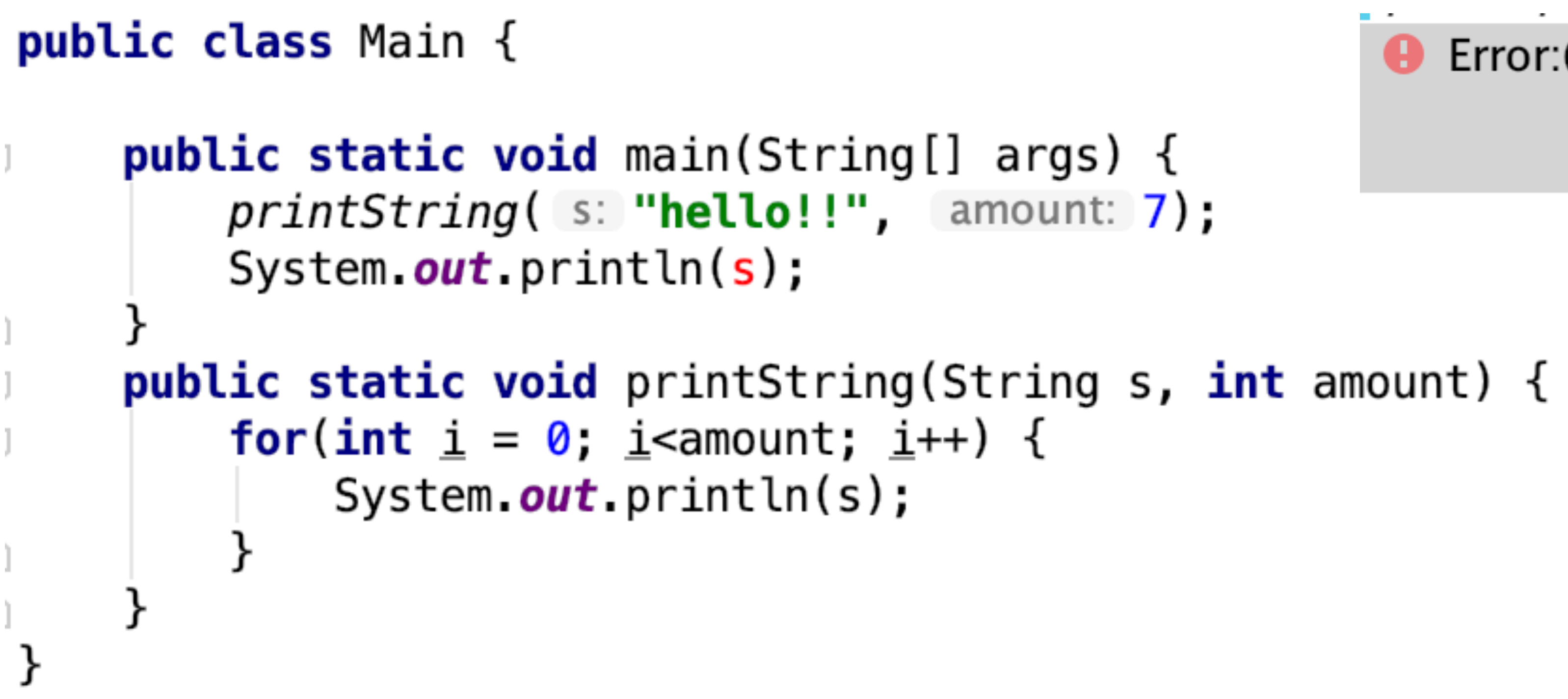


```
hello!!
hello!!
hello!!
hello!!
hello!!
hello!!
hello!!
```

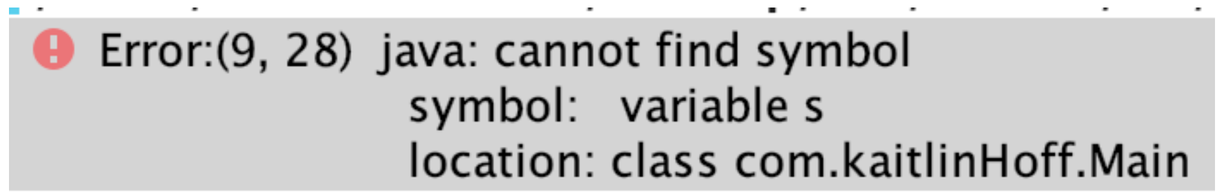
SCOPE OF VARIABLES IN METHODS

- Just like with for loops, variables inside our parameters and the body of our methods **only** exist inside our method. They **DO NOT** exist outside of the method.
- **Ex** - s doesn't exist outside of the method:

Output



```
public class Main {

    public static void main(String[] args) {
        printString( s: "hello!!",  amount: 7);
        System.out.println(s);
    }
    public static void printString(String s, int amount) {
        for(int i = 0; i<amount; i++) {
            System.out.println(s);
        }
    }
}
```



```
Error:(9, 28)  java: cannot find symbol
               symbol:   variable s
               location: class com.kaitlinHoff.Main
```

REUSING A METHOD NAME

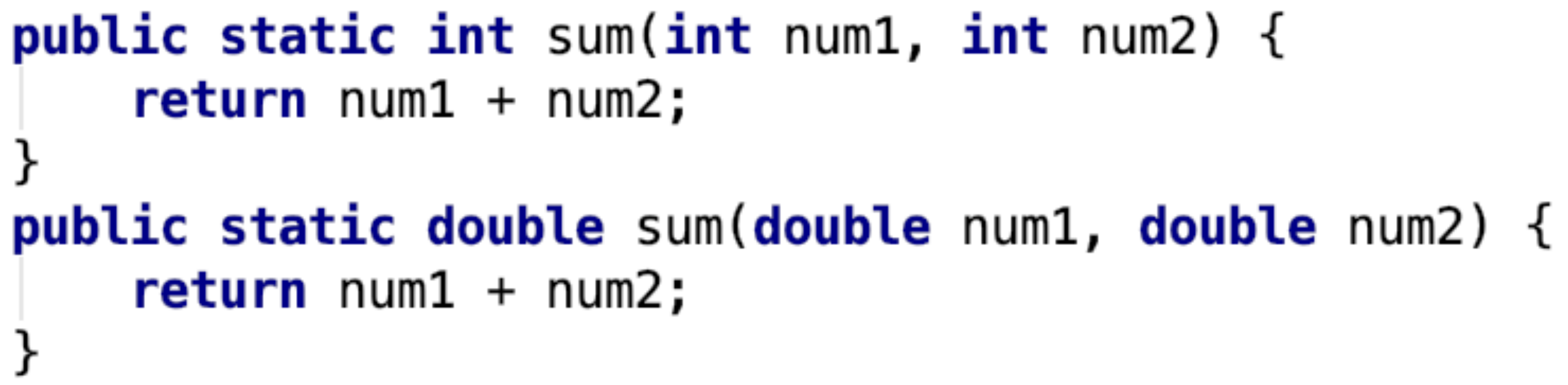
- Do you think you can have two methods with the same name?

REUSING A METHOD NAME

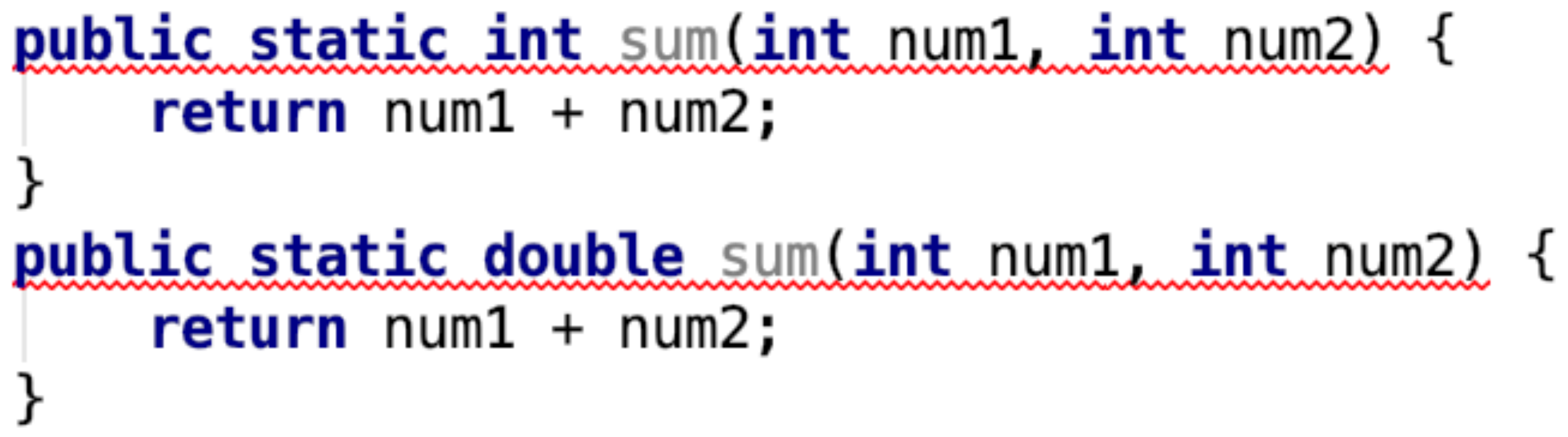
- Do you think you can have two methods with the same name? **It depends!**
- If you have two or more methods all with the same name in the same class, as long as they take a **different number** of parameters or take **different types** of parameters, you can!
- This is called **method overloading**.

REUSING A METHOD NAME

- The first one is **method overloading** and is okay! The second is not allowed since the parameters are identical.



```
public static int sum(int num1, int num2) {
    return num1 + num2;
}
public static double sum(double num1, double num2) {
    return num1 + num2;
}
```



```
public static int sum(int num1, int num2) {
    return num1 + num2;
}
public static double sum(int num1, int num2) {
    return num1 + num2;
}
```