Creating Methods

Java
Mr. Poole

Again what is a method?

A method is a block of code which runs when it is called.

You can pass data, known as parameters, into a method.

Example: `Math.max(3, 5);`

This method calls the block of code that find the maximum of the two parameters given.

Example of a Method

```
class starter {
    public static void main(String args[]) {
        // Your code goes below here

    }
}
```

Normally how your code should look!

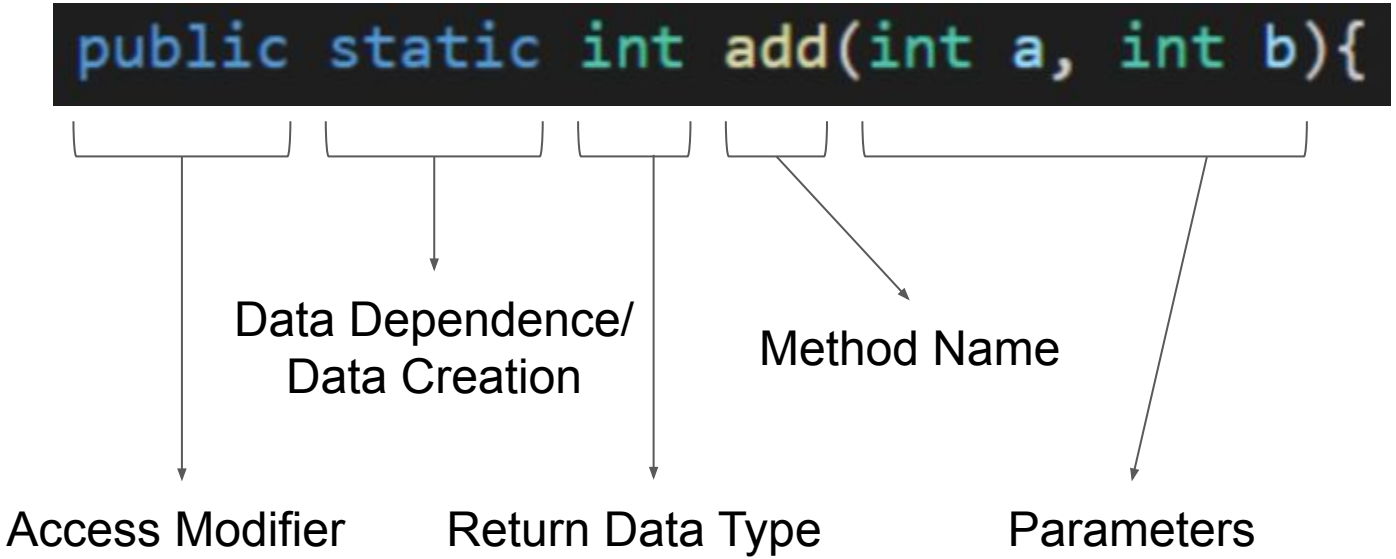
Let’s make an addition method!

Goal: Take in two integers as parameters and add them up

Creating a Method - Addition

```
class starter {

    public static int add(int a, int b){

    }

    public static void main(String args[]) {
        // Your code goes below here

    }
}
```

Breaking down the “add” Method



Access Modifier

```
public static int add(int a, int b){
```

What can access your method!

Types: Private, Protected, Public

Examples of when methods can be used with given modifiers.

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

We will mostly use Public.

Data Dependency/ Data Creation

Types: Static and Non-Static

```
public static int add(int a, int b){
```

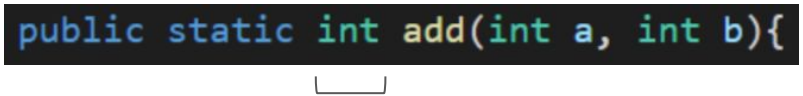
Static method belongs to the class itself and a non-static (aka instance) method belongs to each object that is generated from that class. If your method does something that doesn't depend on the individual characteristics of its class, make it static (it will make the program's footprint smaller). Otherwise, it should be non-static.

```
class Foo {
    int i;

    public Foo(int i) {
       this.i = i;
    }

    public static String method1() {
       return "An example string that doesn't depend on i (an instance variable)";
    }

    public int method2() {
       return this.i + 1; // Depends on i
    }
}
```

We will swap between static and non-static

Return Data Type - **IMPORTANT**!

This is the type of data that is returned to the caller of this method.



Example: When we call the below, we expect an integer to be given back as an answer.

```
int x = add(2,3);
```

Valid Data Type: String, double, int, boolean, **void**

Void - this means nothing will be returned.

How to - Return Data Type

```
public static int add(int a, int b){
```

Example of returning an integer.

```
public static int add(int a, int b){
    int x = a;
    int y = b;
    int sum = x+y;
    return sum;
}
```

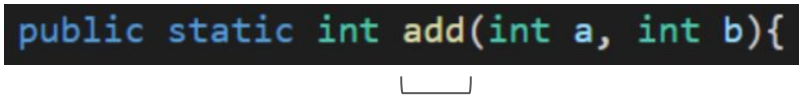
Example of returning void.

```
public static void printAddition(int a, int b){
    int x = a;
    int y = b;
    int sum = x+y;
    System.out.println(x + " + " + y + " = " + sum);
    return;
}
```

WARNING: Return immediately ends your method. It should be the last thing called.

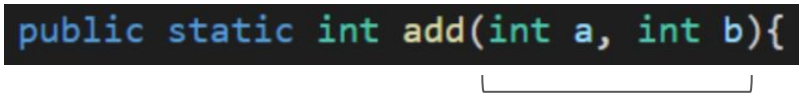
Method Name

- Can be whatever you want!
- Capitalization does matter.
- Usually helps to name it what it does.



Parameters

Parameters are data that is passed into the method for use. This data isn't able to be changed but can be used!



Methods can take as many parameters as you want! They should be used though.

Example: a and b are used in the add method below. They must be defined as ints

```
public static int add(int a, int b){
    int x = a;
    int y = b;
    int sum = x+y;
    return sum;
}
```

Calling Void Methods

```
public static void main(String args[]){
    add(1,2);
}

public static void add (int a, int b){
    System.out.println(a + b);
    return;
}
```

Calling Non-void Methods

```
public static void main(String args[]){
    int a = add(1,2);
}

public static int add (int a, int b){
    return a + b;
}
```

Creating a Method - Try it!

```
public static void main(String args[]){
    int a = add(1,2);
}

public static int add (int a, int b){
    return a + b;
}
```

You can call it using the code above for the example **add**.

Lab - Methods

1. Create a toString method in your starter.java
2. The function itself should print out whatever String is given to the user.
3. This method should have 1 String parameter
4. It should return nothing

Continued

1. Create a second method called toStringCombined.
2. The function itself should print out two given Strings side by side with a space in between.
3. Takes 2 String parameters
4. It should return nothing.