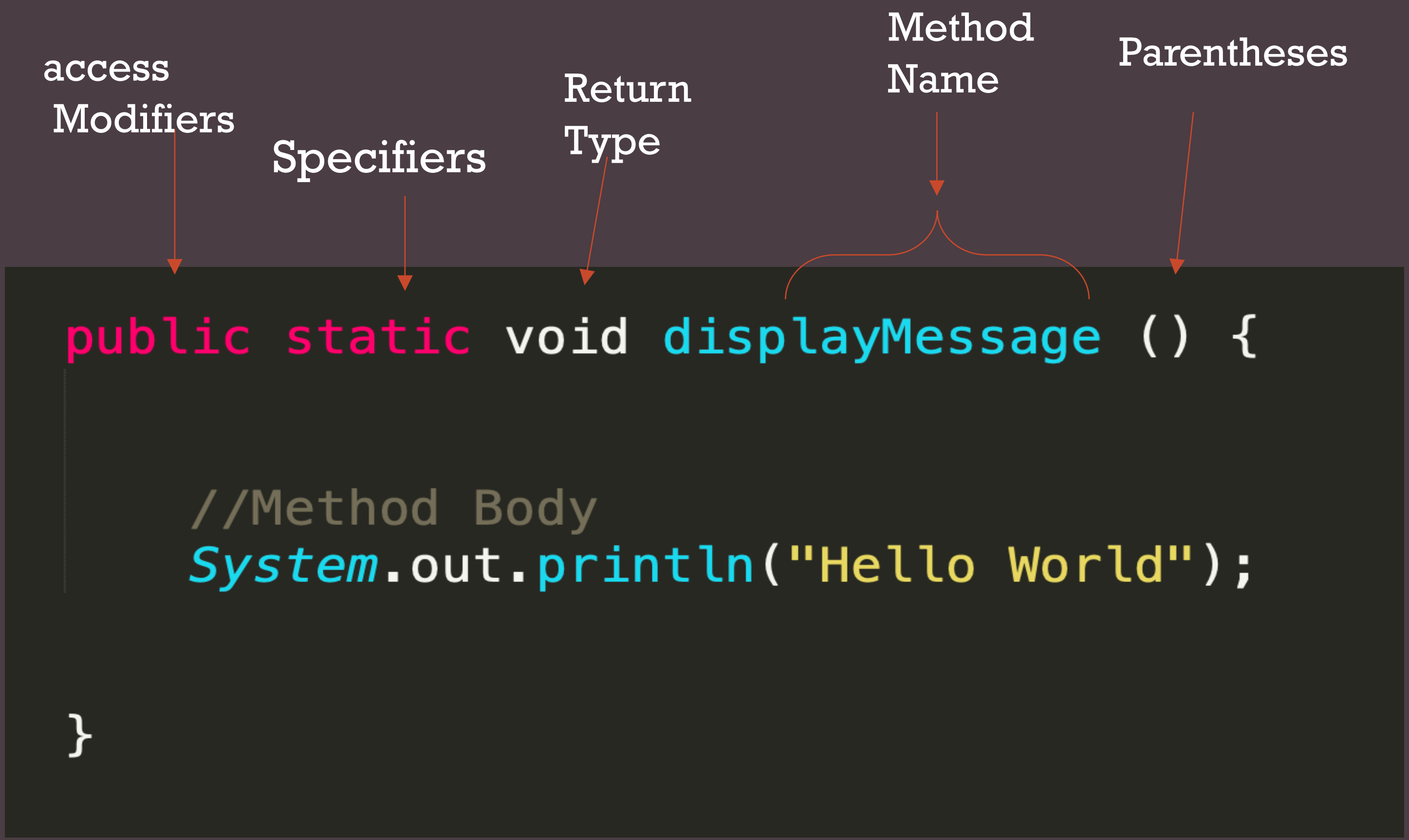
METHODS

Cybertek

What is Method?

Methods let you group a series of statements together to perform a specific task. If different parts of a script repeat the same task, you can reuse the method(rather than repeating the same set of statements.)

Declaring a Method



```
public static void displayMessage () {

    //Method Body
    System.out.println("Hello World");

}
```

Cybertek

- **Access Modifier and Specifiers:** For now, every method we write will begin with public static

- **Return Type:** When the keyword void appears, it means that the method is a void method, and does not return a value. We will see value-returning methods later.

- **Method Name:** Give each method a descriptive name. The same rules that apply to variable names also apply to method names.

- **Parentheses:** The method name is always followed by a set of parentheses. Methods can be capable of receiving arguments. When this is the case, a list of one or more variable declarations will appear inside the parentheses. The method in this example does not receive any arguments, so the parentheses are empty.

Cybertek

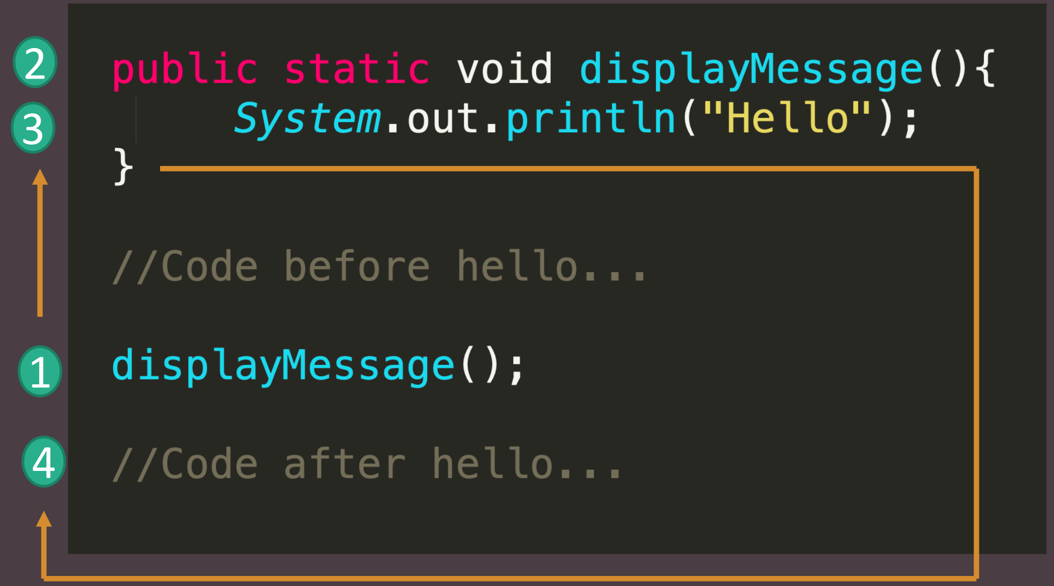
Calling a Method

- Having declared the method, you can then execute all of the statements between its curly braces with just one line of code. This is known as calling the method.

```
displayMessage();
```

Calling a Method

- The methods can store the instructions for a specific task.
- When you need the script to perform that task, you call the method.
- The method executes the code in that code block.
- When it has finished, the code continues to run from the point where it was initially called.



```
public static void displayMessage(){
        System.out.println("Hello");
}

//Code before hello...

displayMessage();

//Code after hello...
```

Task

1. Write a method that calculates the sum of 3 numbers

2. Write a method that shows the grater number from 2 numbers

Hierarchical Method Calls

- Method can also be called in a hierarchical, or layered fashion.

- In other words, method A can call method B, which can then call method C. When method C finishes, JVM returns to method B. When method B finishes , JVM returns to method A.

Passing Arguments to a Method

- Sometimes a method needs specific information to perform its task. In such cases, when you declare the method you give it parameters. Inside the method, the parameters act like variables.

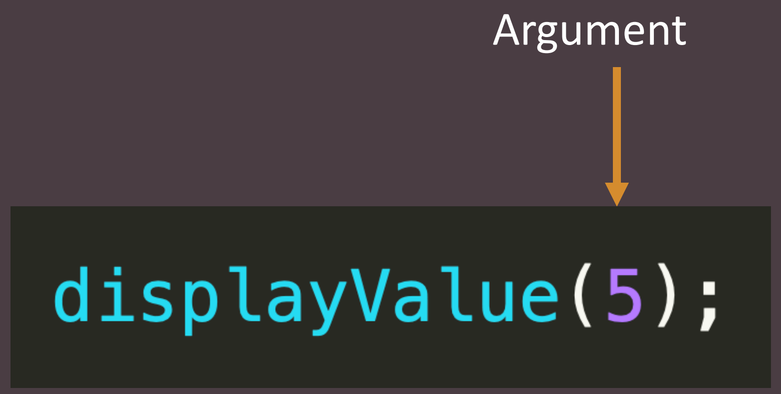
Parameter

```java
public static void displayValue(int num){

    System.out.println("The value is " + num);

}
```

Cybertek

Calling Method That Need Information

- When you call a method that has parameters, you specify the values it should use in the parentheses that follow its name. The values are called arguments, and they can be provided as values or as variables.



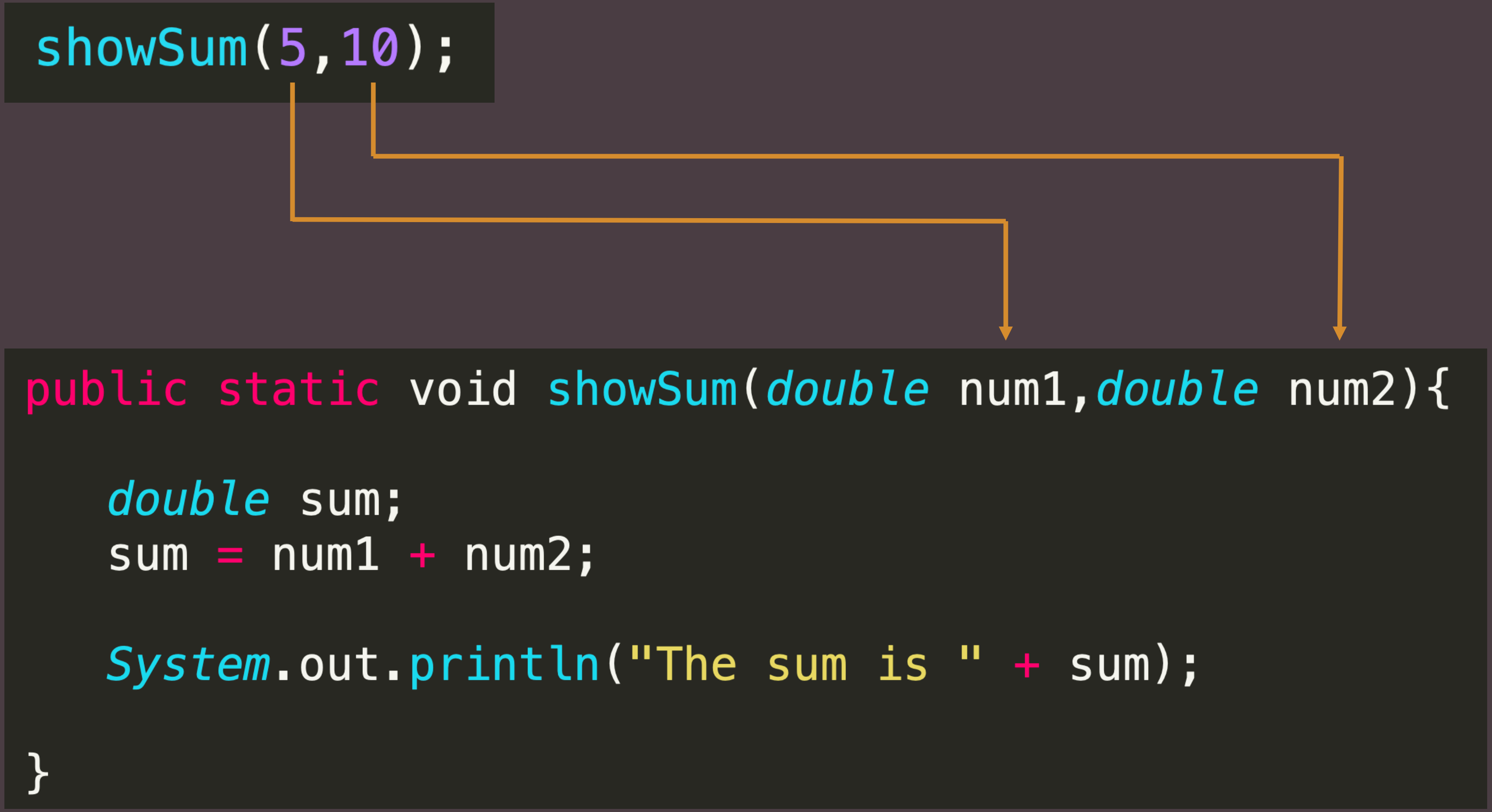
```
displayValue(5);
```

```java
displayValue(5);
```

The argument 5 is copied into
the parameter variable num

```java
public static void displayValue(int num){

    System.out.println("The value is " + num);

}
```

Passing Multiple Arguments



```java
showSum(5,10);
```

```java
public static void showSum(double num1,double num2){

    double sum;
    sum = num1 + num2;

    System.out.println("The sum is " + sum);

}
```

Cybertek

Task

Write a method that accepts 3 parameters:

1- number

2- number

3- operator(-,+,*,/)

Sample output:

calculator(6,3, “+”)   -- > 9

calculator(6,3, “-”)   -- > 3

calculator(6,3, “*”)   -- > 18

calculator(6,3, “/”)   -- > 2