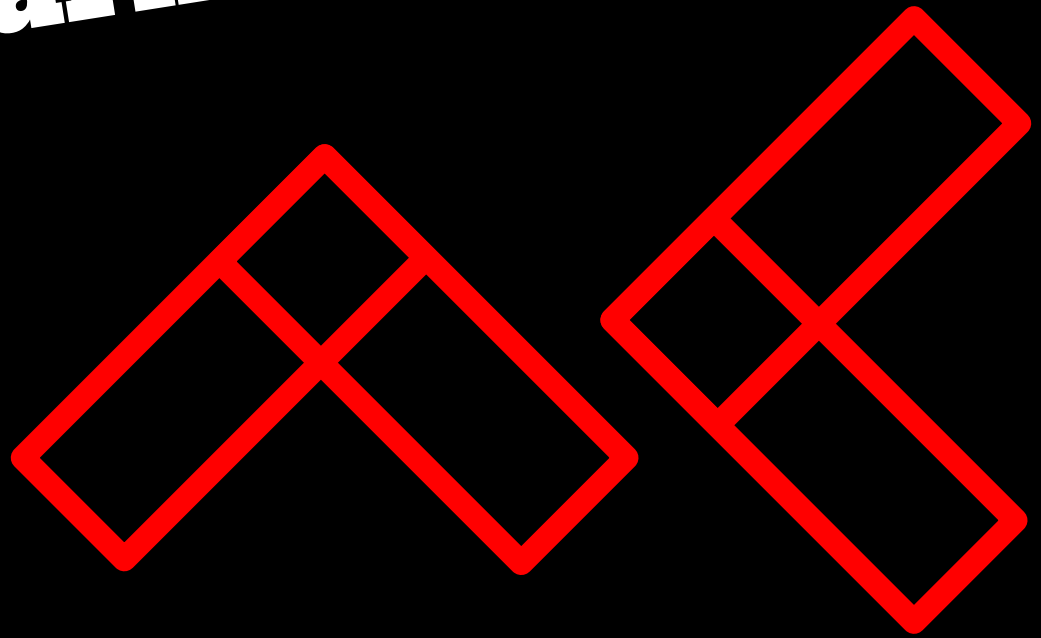
summarizer 0xA



Sponsored by: ACADEMIADECODIGO

Produced by: Paulo Gomes @16_stormrooters 15/05/2018

Last class was crazy!!

Scary room

New neighbours

Constructor

private

Objects

Lots of information!!

Encapsulation

Instances

Methods

Void

New definitions!!

public

New conventions!!

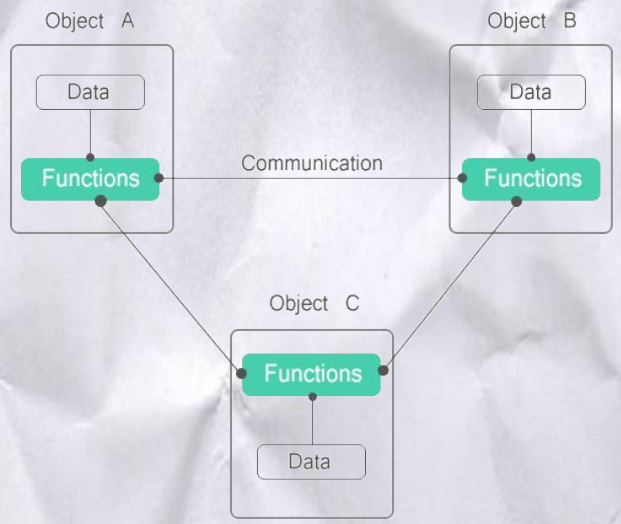
OOP

OOP
(Object Oriented Programming)

Programming paradigm where the data type of the data structure is defined, as well as the operations we want to apply to these structures

Objects become **data structures**, that include data and functions

OOP
(Object Oriented Programming)
Program



An object oriented program consists of many objects interacting with each other through functions (methods).

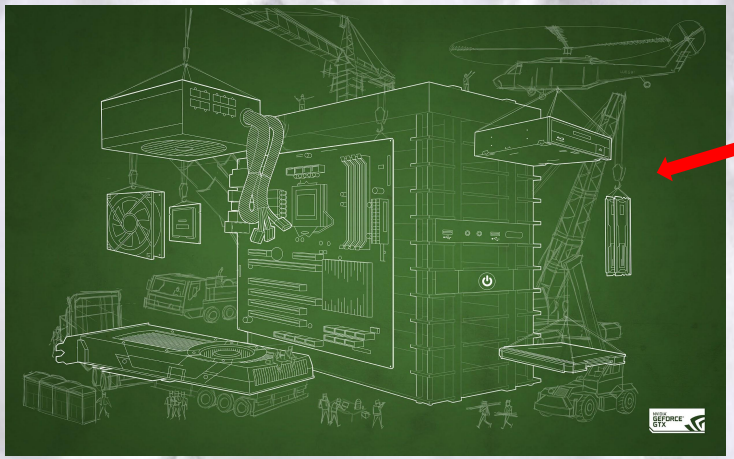
Object

Container of properties and its behaviours

Properties / States

Methods / Behaviours

Create Objects



To created objects we need:

classes

instances

Create a new class

Calculator.java × | Main.java ×

```java
package org.academiadecodigo.bootcamp.calculator;

//create a new class
public class Calculator {

    //properties or instance variables
    public String brand;
    public String color;

}

```

Create calculator instances

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

public class Main {

    public static void main(String[] args) {

        //create new instance (an object of the Calculator class)
        Calculator calc = new Calculator();

        //attribute values to the properties of the instance we just created
        calc.brand = "Casio";
        calc.color = "Deep Dark";

        //create a second instance, and attribute values to its properties
        Calculator calc2 = new Calculator();

        calc2.brand = "Xiaomi";
        calc2.color = "Escaping Donkey Color";

        //output
        System.out.println("Calculator1 is a " + calc.brand + " and it color is " + calc.color);
        System.out.println("Calculator2 is a " + calc2.brand + " and it color is " + calc2.color);
    }

}
```

```
Calculator1 is a Casio and it color is Deep Dark
Calculator2 is a Xiaomi and it color is Escaping Donkey Color

Process finished with exit code 0
```

Method

Group of statements, that together perform an operation

Function defined in a class

Provide functionality to our objects

Declaring a method

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

//create a new class
public class Calculator {

    //properties or instance variables
    public String brand;
    public String color;

    //declare the method add
    public void add() {

        int num1 = 4;
        int num2 = 7;

        //print the result
        System.out.println(num1 + num2);

    }

}
```

Call the method

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

public class Main {

    public static void main(String[] args) {

        //create new instance (an object of the Calculator class)
        Calculator calc = new Calculator();

        //attribute values to the properties of the instance we just created
        calc.brand = "Casio";
        calc.color = "Deep Dark";

        //call the method
        //we don't need to pass a value to it
        //because we already did that when declaring it
        calc.add();

        //print 11
    }

}
```

A method using two attributes wasn't enough

So we created one were we could add values as arguments

Adding parameters to a method

Calculator.java | Main.java

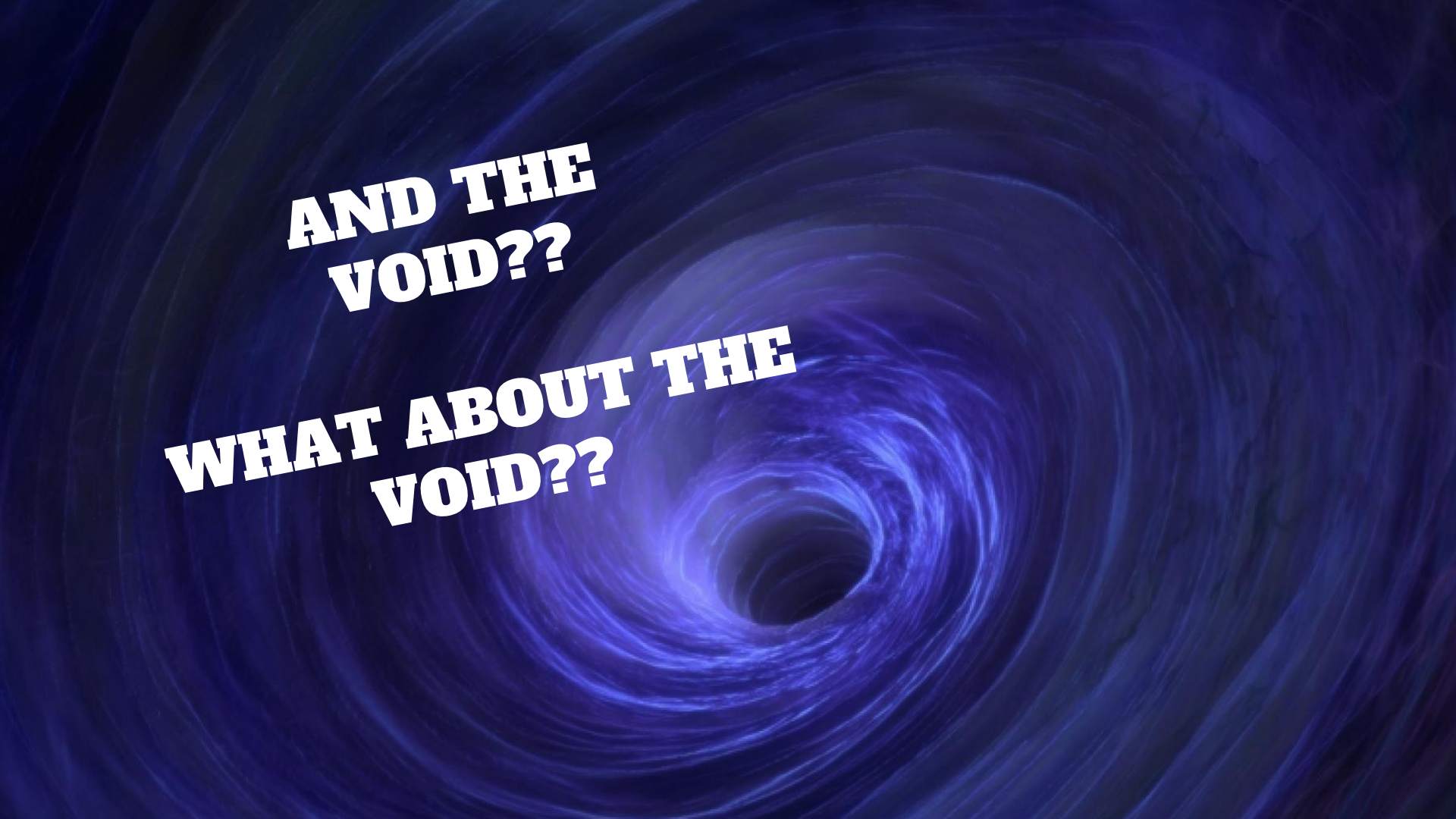
```java
package org.academiadecodigo.bootcamp.calculator;

//create a new class
public class Calculator {

    //properties or instance variables
    public String brand;
    public String color;

    //declare the method add with two parameters
    public void add(int num1, int num2) {

        //print the result
        System.out.println(num1 + num2);

    }

}
```

Calling a method with parameters

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

public class Main {

    public static void main(String[] args) {

        //create new instance (an object of the Calculator class)
        Calculator calc = new Calculator();

        //attribute values to the properties of the instance we just created
        calc.brand = "Casio";
        calc.color = "Deep Dark";

        //call the method add, and giving it to parameters
        calc.add(7,9);

        //print 16
    }

}
```

```
16

Process finished with exit code 0
```


AND THE VOID??
WHAT ABOUT THE VOID??

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

//create a new class
public class Calculator {

    //properties or instance variables
    public String brand;
    public String color;

    //declare the method add
    public void add() {

        int num1 = 4;
        int num2 = 7;

        //print the result
        System.out.println(num1 + num2);

    }

}
```

The keyword <u>void</u> means that the method doesn't return anything

Besides many things, it can print to the console

The <u>return type</u> defines the data type of the value the method returns when called

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

//create a new class
public class Calculator {

    //properties or instance variables
    public String brand;
    public String color;

    //change from return type void to return type int
    public int add(int num1, int num2) {

        //print the result
        return num1 + num2;

    }

}
```

Using the return value

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

public class Main {

    public static void main(String[] args) {

        //create new instance (an object of the Calculator class)
        Calculator calc = new Calculator();

        //attribute values to the properties of the instance we just created
        calc.brand = "Casio";
        calc.color = "Deep Dark";

        //get return values , and attribute its value to the result variable
        int result = calc.add(6, 9);//15

        //get the return value of the addition (result + 3)
        result = calc.add(result, 3);//15 + 3

        //print the returned value
        System.out.println("The result of this addition is: " + result);
    }

}
```

Method Overloading

When a class has two or more methods with the same name, but different arguments

We need to be aware of the <u>method signature</u>, and it is constituted by:

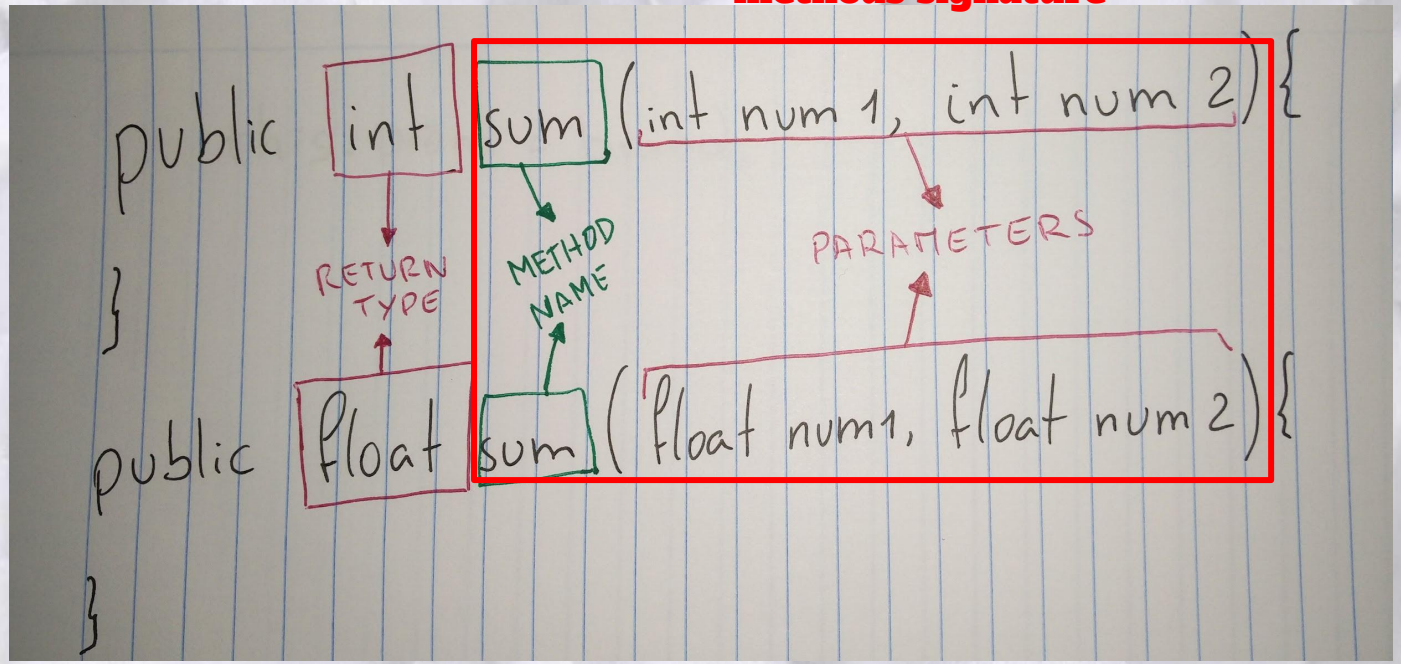
name of method

number of arguments

types of arguments

order of arguments

methods signature



We can't have methods with the same signature

Constructor

Called when a new object is created

Responsible for initializing the object

Called only once for each object

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

public class Main {

    public static void main(String[] args) {

        //create new instance (an object of the Calculator class)
        //here we are using the default constructor
        Calculator calc = new Calculator();

        //attribute values to the properties of the instance we just created
        calc.brand = "Casio";
        calc.color = "Deep Dark";

        //get return values , and attribute its value to the result variable
        int result = calc.add(6, 9);//15

        //print the returned value
        System.out.println("The result of this addition is: " + result);
    }

}
```

Default Constructor is an automatically generated no-argument constructor method

Declaring a new Constructor

When we define a new Constructor, the default is not created

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

//create a new class
public class Calculator {

    //properties or instance variables
    public String brand;
    public String color;

    //declaring a new constructor
    public Calculator (String brand, String color) {
        this.brand = brand;
        this.color = color;
    }

    //change from return type void to return type int
    public int add(int num1, int num2) {

        //print the result
        return num1 + num2;

    }

}
```

Using a new Constructor

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

public class Main {

    public static void main(String[] args) {

        //create new instance (an object of the Calculator class)
        //using the new constructor
        Calculator calc = new Calculator("Casio", "Deep Dark");

        //calc.brand = "Casio";
        //calc.color = "Deep Dark";

        //get return values , and attribute its value to the result variable
        int result = calc.add(6, 9);//15

        //print the returned value
        System.out.println("The result of this addition is: " + result);
    }

}
```

```
this.brand = brand;
```

In the Scary Room we heard that it can have two uses:

It can be used to differentiate the <u>local variables</u> from an <u>object's properties</u>

||

It can be used when we want to <u>refer to an objects' properties</u>
(HIGHLY RECOMMENDED BY THE MASTERS)

Encapsulation

This concept is used to hide the internal representation, or state, of an object from the outside

It provides read/write methods to access and use the hidden data

(GETTER && SETTER)

Hiding Data

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

//create a new class
public class Calculator {

    //properties or instance variables
    public String brand;
    public String color;

    //declaring a new constructor
    public Calculator (String brand, String color) {
        this.brand = brand;
        this.color = color;
    }

    public String getBrand() {
        return this.brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }
}
```

Accessing Data

Calculator.java | Main.java

```java
package org.academiadecodigo.bootcamp.calculator;

public class Main {

    public static void main(String[] args) {

        //create new instance (an object of the Calculator class)
        //using the new constructor
        Calculator calc = new Calculator("Casio", "Deep Dark");

        //using the SETTER
        calc.setBrand("Texas Instrumentals");

        //using the GETTER
        String brand = calc.getBrand();

        //get the return value, and attribute its value to result variable
        int result = calc.add(7,3);

        //print the returned value
        System.out.println("\nThe result of this addition is: " + result +
                " and the brand of the calculator is: " + brand);
    }

}
```

```
The result of this addition is: 10 and the brand of the calculator is: Texas Instrumentals

Process finished with exit code 0
```


THANK YOU
FOR YOUR ATTENTION