Welcome to the

Java Course

Module 3 - OOP

Let’s resume!

```
if (condition) {
    //condition is true;
} else {
    //condition is false;
}
```

Let’s resume!

```
while (condition) {
        // Code to be executed
}
```

Let’s resume!

```
for (initialization;condition;update) {
        // Code to be executed
}
```

Let’s resume!

```
public static void function_name(){
    //portion of code
}
```

Let’s resume!

type *array_name*[ ] = { *val_0* , … , *val_n*}

Let’s resume!

type *matrix*[ ][ ] = { {…}, {…}, {…} , … };

Let’s practice!

OOP

Object Oriented Programming is a different way to concept the code and the world around us.

OOP

Everything can be an object, even an abstract concept. The point is considering anything as an object.

OOP

The **CLASS** is a generic object that represents a category of things.

OOP

For example:

DOG

Not a specific dog, but the concept of a generic dog.

OOP

```
public class MyClassName {

}
```

OOP

```
public class Dog {

}
```

OOP

A Class has 2 things:

attributes

methods

OOP

For example:

DOG

An attribute could be the size or the color or the number of tales!

OOP

```
public class Dog {
    //definition of the attributes
    private double size;
    private String color;
    private int tail_n;
}
```

OOP

```
public class Dog {
    //definition of the attributes
    private double size;
    private String color;
    private int tail_n;
}
```

OOP

```
public class Dog {
    //definition of the attributes
    private double size;
    private String color;
    private int tail_n;
}
```

Let’s practice!

OOP

Functions defined into a class are called **Methods**. They are the actions that the object can execute.

OOP

The method aimed to the attributes initialization is called **constructor**. Using it, you can define the attribute values.

OOP

```
public class Dog {

    // Constructor

public Dog(double size, String color, int tail_n) {
    this.size = size;
    this.color = color;
    this.tail_n = tail_n;
  }
```

OOP

```
public class Dog {

    // Constructor

public Dog(double size, String color, int tail_n) {
    this.size = size;
    this.color = color;
    this.tail_n = tail_n;
  }
```

OOP

```
public class Dog {

    // Constructor

public Dog(double size, String color, int tail_n) {

    this.size = size;

    this.color = color;

    this.tail_n = tail_n;
  }
```

OOP

```
public class Dog {

    // Constructor

public Dog(double size, String color, int tail_n) {
    this.size = size;
    this.color = color;
    this.tail_n = tail_n;
  }
```

OOP

```
public class Dog {

    // Constructor

public Dog(double size, String color, int tail_n) {
    this.size = size;
    this.color = color;
    this.tail_n = tail_n;
  }
```

OOP

For example:

DOG

A method that just perform an action could be to bark.

OOP

```
public void bark() {
    System.out.println(“woof!”);
}
```

OOP

For example:

DOG

A method that uses passed parameters could be to eat.

OOP

```
public void eat(double food) {
        if (food > 10){
            this.full = true ;
        }
}
```

OOP

For example:

DOG

A method that returns a value could be to poop!

OOP

```
public int poop() {
        if (this.full) {
            return 200;
        }
}
```

Let’s practice!

OOP

An **instance** is a specific object of a class, with specific property values. It's not a generic object, but an object with a name and an identity.

OOP

To create an instance:

Class *insanceName* = new Class()

OOP

To create an instance:

Class *insanceName* = new Class()

OOP

To create an instance:

Class *insanceName* = new Class()

OOP

To create an instance:

Class *insanceName* = new Class()

OOP

When you create an instance, you can pass attribute values within the parentheses to directly utilize the constructor.

OOP

To pass attribute values:

Class *insanceName* = new Class(attr1, attr2, ...)

OOP

```
Dog bobby = new Dog("yellow");
```

OOP

To use a method within an instance, simply place a period

“.” after the instance name.

OOP

To use a method:

instance.method_name()

OOP

To use a method:

```
instance.method_name()
```

OOP

To use a method:

instance.method_name()

OOP

```
String dog_color = bobby.getColor();
System.out.println(dog_color);
```

Let’s practice!