

Introdução a Java

//Programação Orientada a Objetos

IT BOARDING

BOOTCAMP



Índice



01 POO

02 Classes e Objetos

03 Métodos

04 Construtores



IT BOARDING

BOOTCAMP

Programação Orientada a Objetos

// O que é?

IT BOARDING

BOOTCAMP



“ É um paradigma de programação onde o código é organizado em unidades chamadas classes, a partir das quais são criados objetos relacionados entre si, simplificando o desenvolvimento de software e a manutenção de aplicações. ”



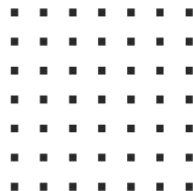
Ele permite que você modele facilmente conceitos do mundo real no nível de programação.



Oferece um conjunto de ferramentas que permitem grande flexibilidade no desenho de uma solução.



Também chamado de POO ou OOP em inglês.





POO vs Programação estruturada

Programação Orientada a Objetos	Programação Estruturada
Forma de programação mais próxima de como você expressaria as coisas na vida real	Visa resolver um problema do início ao fim em uma única estrutura de código
Análise orientada a objetos	Análise orientada aos processos do sistema
Existem diferentes maneiras de lidar com a solução de um problema	Técnicas estruturadas básicas: sequência, seleção, repetição
Funções e dados são encapsulados em uma entidade	Funções e dados são tratados como entidades separadas

Classes e Objetos

IT BOARDING

BOOTCAMP



// Classes

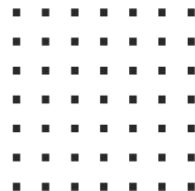
Grupo de variáveis e operações agrupadas em módulos coerentes. Uma classe pode ter atributos, construtores e métodos. Pode-se dizer que é um template que define a forma de um objeto.

```
public class Libro {  
    String nombre;  
    String autor;  
}
```

// Objetos

São instâncias de uma classe, ou seja, quando você cria um objeto, ele pertence a uma determinada classe e há uma representação física dessa classe na memória.

```
Libro libro = new Libro();
```



Então...

Como criamos uma classe em Java?

Os métodos e variáveis que constituem uma classe são chamados de **membros de uma classe**. Os membros de dados são conhecidos como **variáveis de instância**.



É definido pela palavra-chave **class**. Abaixo podemos ver a estrutura básica e simplificada de uma classe

```
public class NombreClase {  
  
    //Declaración variables de instancia  
    tipoDato variable1;  
    tipoDato variable2;  
  
    //Declaración de métodos  
    ...  
}
```



E como criamos um objeto?

Para criar efetivamente um objeto, usaremos a seguinte instrução:

Definimos o tipo de dado, ou seja, a classe a qual o objeto pertencerá

```
Libro libro = new Libro();
```

Usamos a nova palavra-chave

Indicamos um nome significativo para a variável

Chamamos o construtor da classe



Você não sabe o que queremos dizer? **Não se preocupe!** Veremos mais tarde



Métodos e Construtores

IT BOARDING

BOOTCAMP



Métodos

Também chamados de
funções ou
procedimentos em
outras linguagens.

// Conjunto de instruções definidas em uma classe que executam uma determinada tarefa.

Como se escreve um método?

Eles são identificados por ter parênteses após seu nome. Um método terá:

- *Um modificador de acesso*
- *Um tipo de retorno*
- *O nome (deve ser descritivo)*
- *Pode ou não ter parâmetros.*



{Vejam os um exemplo da sintaxe}

Modificador de Acesso

O método pode ser acessado de qualquer lugar do programa

Tipo de retorno

O tipo de dados retornado pelo método é especificado

Parâmetros

São variáveis que recebem o valor dos argumentos passados ao método, caso o método não possua parâmetros a lista ficará vazia

```
{ public int sumarNumero(int numero1, int numero2) {  
    //Cuerpo del método  
}
```

Nome

Deve ser um nome descritivo para o que o método executa, pode ser qualquer identificador válido que não tenha sido usado no contexto atual



Você sabia que a combinação do nome do método e a lista de parâmetros constituem a **assinatura do método**?





Método `main()`

Os programas Java iniciam sua execução pelo método `main ()`, ou seja, é o ponto de entrada da aplicação.

A palavra-chave **`static`** indica que um objeto desta classe não precisa ser criado para chamar esse método

Este método não retorna nenhum valor, este tipo de retorno é denominado **`void`**

O nome do método deve ser **`main`**

Ele deve aceitar um **array de `String`**. O nome dos parâmetros ou argumentos do método pode ser qualquer identificador válido

A palavra reservada **`public`** indica que ele pode ser acessado de qualquer lugar do programa

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hola Mundo");  
    }  
}
```



Você pode escrever **`String [] args`**, **`String args []`** ou **`String... args`**. O compilador aceita qualquer uma dessas alternativas



Métodos de instância vs Métodos de Classe

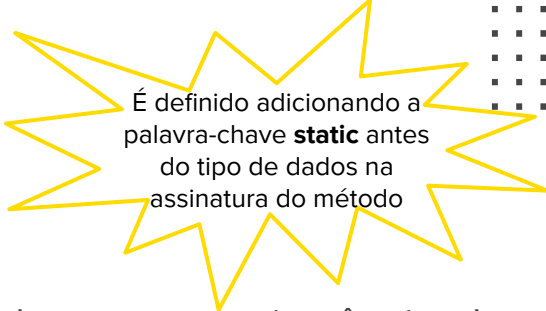
Os **métodos de instância** podem ser acessados por meio de uma instância ou objeto de uma classe.

```
public class Livro {  
  
    private String nomeLivro;  
    private String autor;  
  
    public String mostrarNomeComAutor() {  
        System.out.println(nomeLivro + " " + autor);  
    }  
  
    public String mostrarNomeSemAutor() {  
        System.out.println(nomeLivro);  
    }  
}
```

//Sintaxe da chamada:
instância.método(parâmetros)

```
Livro livro = new Livro()  
livro.mostrarNomeComAutor();  
livro.mostrarNomeSemAutor();
```

Métodos de classe



É definido adicionando a palavra-chave **static** antes do tipo de dados na assinatura do método

Um **método de classe** é aquele que pode ser chamado sem uma instância da classe. Um exemplo típico de uso é a classe `java.lang.Math`

```
package java.lang;

public class Math {

    //Retorna o menor valor de dois valores int.
    public static int min(int a, int b) {...}

    //Retorna o maior de dois valores int.
    public static int max(int a, int b) {...}

    //Retorna o valor do primeiro argumento elevado à
    potência do segundo argumento.
    public static double pow(double a, double b) {...}
}
```

//Sintaxe da chamada:

`Classe.método(parâmetros)`

`int menor = Math.min(2,3)`

`int maior = Math.max(5,8)`

`double potencia = Math.pow(2,2)`

// CONSTRUTORES



Você se lembra na criação do objeto livro quando chamamos o construtor da classe? ... Isso é o que queríamos dizer! Vamos lembrar o código ...

```
Livro livro = new Livro();
```

Neste caso, **Livro()** é um método, pois é seguido por parênteses. É um tipo **especial de método** chamado construtor, que nos permite criar um novo objeto.



Declaração de um construtor

- O nome do construtor deve corresponder ao nome da classe
- Não tem tipo de retorno
- A existência de parâmetros é opcional
- Pode haver mais de um, embora apenas um será executado quando a classe for criada

```
//Constructor sin parámetros  
public Libro() {  
  
}
```

```
//Constructor con parámetros  
public Libro(String nombre, String autor) {  
    this.nombre = nombre;  
    this.autor = autor;  
}
```



Construtores



Cada classe em Java tem um construtor, mesmo se não o escrevermos. Caso contrário, o Java criará um **Construtor padrão** sem parâmetros. O construtor é responsável por inicializar o valor de cada atributo da nova instância.



Atributos do tipo primitivo são inicializados com **0** ou **false** por padrão, enquanto os atributos do tipo objeto (referência) são inicializados com **null**.



Java não permite que variáveis de membro de uma nova instância sejam não inicializadas, portanto, se não forem inicializadas, apontam para nulo.



Sobrecarregando um construtor (Overloading)

“

A sobrecarga permite que vários construtores sejam declarados para a mesma classe (deve ter o mesmo nome da classe), desde que tenham um tipo e / ou número de parâmetros diferente. Isso permite que você construa um objeto de maneiras diferentes.

”





Vejamos um exemplo de sobrecarga ...

```
public class Pessoa {  
    private int id;  
    private String nome;  
    private int idade;  
  
    public Pessoa(int id, String nome) {  
        this.id = id;  
        this.nome = nome;  
    }  
  
    public Pessoa(int id, String nome, int idade) {  
        this.id = id;  
        this.nome = nome;  
        this.idade = idade;  
    }  
  
    public void mostrarPessoa() {  
        System.out.println(id+ " " + nome + " " + idade);  
    }  
}
```

```
public static void main(String[] args) {  
  
    Pessoa pessoa1 = new Pessoa(1, "Martin");  
    Pessoa pessoa2 = new Pessoa(2, "Graciela", 26);  
    pessoa1.mostrarPessoa();  
    pessoa2.mostrarPessoa();  
}
```

Qual será o resultado deste programa?
(Clique para descobrir)

```
1 Martin 0  
2 Graciela 26
```

Agora ... A prática leva à perfeição

Vá em frente!

IT BOARDING

BOOTCAMP



Obrigado

IT BOARDING

BOOTCAMP

