



INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Módulo 1: Programação em Linguagem Java

Aula 6 Objetos



Programação Orientada a Objetos

O paradigma de programação **dominante** hoje em dia é o da **programação orientada por objetos**.

Neste paradigma, o código manipula **objetos com estado e comportamento próprios**. Um objeto representa tipicamente uma **entidade** do mundo real (física ou não).

O universo dos **objetos de determinado tipo** é designado por **classe de objetos** (descrição dos objectos de um dado tipo).

Classes de Objetos

- Uma classe de objetos representa um **tipo de objetos**
- O nome da classe deve refletir o que os objetos são (no singular)
 - Exemplos: Scanner, Math, String
 - Convenção: começar com letra maiúscula
- Um objeto é uma instância de uma classe

Classes de Objetos

- A definição de uma classe de objetos é essencialmente composta por:
 - **Atributos:** variáveis que definem o estado de um objeto
 - **Métodos construtores:** métodos particulares que têm como objetivo criar objetos da classe
 - **Métodos:** definições de operações sobre os objetos
 - Funções
 - Procedimentos

Classe de Objetos: Person

```
public class Person {  
  
    private String firstName;  
    private String lastName;  
  
    public Person(String firstName, String lastName){  
        this.firstName = firstName;  
        this.lastName = lastName;  
    }  
  
    public String getFirstName(){  
        return firstName;  
    }  
  
    public String getLastName() {  
        return lastName;  
    }  
  
    public void setFirstName(String  
newFirstName){  
        this.firstName = newFirstName;  
    }  
  
    public void setLastName(String newLastName) {  
        this.lastName = newLastName;  
    }  
}
```



Atributos

Atributos são variáveis cujos valores caracterizam um objeto

- Representam o estado do objeto
- Cada objeto guarda valores para os seus atributos

```
public class Person {  
  
    private String firstName;  
    private String lastName;  
  
    //...  
}
```

firstName = “João”

lastName = “Santos”

joao

Métodos Construtores

- Um método **construtor** de uma classe é o método particular cujo propósito é **criar objetos** dessa classe
- O papel de um construtor é **inicializar os atributos do objeto criado**
- Podem haver **vários construtores** numa mesma classe, porém com parâmetros diferentes
- Caso não sejam definidos construtores numa classe, existe **por omissão um construtor sem parâmetros**

Métodos Construtores

Definição

```
public class Person {  
  
    private String firstName;  
    private String lastName;  
  
    public Person(String firstName, String lastName){  
        this.firstName = firstName;  
        this.lastName = lastName;  
    }  
  
}
```

Invocação

```
public class Main {  
  
    public static void main(String[] args) {  
        Person joao = new Person("João", "Silva");  
    }  
}
```

Métodos

```
public class Person {  
  
    private String firstName;  
    private String lastName;  
  
    public Person(String firstName, String lastName){  
        this.firstName = firstName;  
        this.lastName = lastName;  
    }  
  
    public String getFirstName(){  
        return firstName;  
    }  
  
    public String getLastName() {  
        return lastName;  
    }  
  
    public void setFirstName(String newFirstName){  
        this.firstName = newFirstName;  
    }  
  
    public void setLastName(String newLastName) {  
        this.lastName = newLastName;  
    }  
}
```

- As **operações** disponíveis num objeto são definidas em **métodos**, os quais têm acesso aos atributos.
- O **this** é uma referência ao objeto atual. O this pode ser usado em qualquer membro do objeto (atributos, métodos ou construtor).

Métodos **Get** e **Set**

```
public class Person {  
  
    private String firstName;  
    private String lastName;  
  
    public Person(String firstName, String lastName){  
        this.firstName = firstName;  
        this.lastName = lastName;  
    }  
  
    public String getFirstName(){  
        return firstName;  
    }  
  
    public String getLastname() {  
        return lastName;  
    }  
  
    public void setFirstName(String newFirstName){  
        this.firstName = newFirstName;  
    }  
  
    public void setLastName(String newLastName) {  
        this.lastName = newLastName;  
    }  
}
```

É comum existirem métodos de inserção e de retorno dos atributos. Por norma, o nome destes métodos respeita a seguinte nomenclatura:

- **getAttribute()**
- **setAttribute(...)**

Os atributos de uma classe devem ser sempre **acedidos** e **definidos** através dos **getters** e **setters**, e nunca diretamente!

Exercício 1

Implemente a classe Person:

- Atributos: primeiro nome, último nome, idade, morada, nacionalidade
- Defina *getters* e *setters* para todos os atributos
- O construtor deve receber o primeiro nome, o último nome e a idade

Crie um teste de forma a validar que a implementação funciona. Pode utilizar como base o exemplo abaixo:

```
public class Main {  
    public static void main(String[] args){  
        Person joao = new Person("João", "Silva", 20);  
        System.out.println(joao.getFirstName());  
        joao.setLastName("Santos");  
        System.out.println(joao.getLastName());  
    }  
}
```

Método ***toString()***

- A operação *String toString()* é uma função standard do Java cujo objetivo é **devolver uma representação textual do objeto**. A operação **existe por omissão** para todos os objetos. Contudo, caso a queiramos redefinir, terá que ser definido um método na classe em questão.
- A execução desta operação, retornaria algo assim se o método *toString* não fosse implementado:

System.out.println(joao);  Person@1b6d3586

Método **toString()**

- Como redefinir a função `toString()` de forma a devolver um outro valor?

```
@Override  
public String toString(){  
    return "O nome é "+firstName+" "+lastName+" e tenho "+age+" anos.";  
}
```

Agora, ao imprimir o valor do objeto, temos um resultado diferente:

System.out.println(joao);  “O meu nome é João Santos e tenho 20 anos”

Exercício 2

Implemente na classe Person o método `toString()` que deve retornar a seguinte informação:

- Primeiro nome
- Último nome
- Idade

Exercício 3

Desenvolva uma classe para representar retângulos, tendo em conta a sua largura e comprimento. Os objetos retângulo deverão ser imutáveis, isto é, uma vez criado as suas dimensões não podem ser alteradas.

- Implemente o construtor e métodos que permitem obter o comprimento e a largura do retângulo
- Defina um método construtor adicional que cria um quadrado.
- Defina as funções que permitem obter as seguintes informações:
 - área
 - perímetro
 - comprimento da diagonal
 - se o retângulo é um quadrado

Exercício 4

Crie um sistema de gestão de alunos numa sala. Para tal, deverá criar as classes Room e Student. Cada sala tem a sua capacidade, o nome do bloco (ex: A, B, C, D) e o número da sala. Cada aluno terá o número de aluno, o nome e o curso. Deve ser possível realizar as seguintes operações:

- adicionar alunos a uma sala (até ao limite da sua capacidade)
- remover um aluno específico, com base no seu número
- listar todos os alunos que estão na sala



O futuro profissional começa aqui

