

Introdução à Programação Orientada a Objetos



Jeferson Souza (thejefecomp), Ph.D. Candidate
thejefecomp@neartword.com



O que é Programação Orientada a Objetos?

Definição de Programação Orientada a Objetos

Pode-se dizer, de forma divertida, que **Programação Orientada a Objetos** é a arte de programar por meio do pensamento envolto no **Paradigma Orientado a Objetos**, onde todo o programa (software) é representado estaticamente por elementos denominados **Classes**, os quais são instanciados dinamicamente e materializados em elementos chamados de **Objetos**. Portanto, desenha-se o programa a pensar em **Classes**, e executa-se o programa a considerar sua materialização em **Objetos**.

Mas afinal, o que é uma Classe?

Definição de Classe

O conceito de **Classe** remete a um elemento com contexto e domínio muito bem definidos, o qual é composto de atributos (i.e., suas características intrínsecas), e métodos (i.e., suas operações) a representar o seu comportamento interno (i.e. dentro da classe) e externo (i.e. a interagir com elementos fora da classe). Uma **Classe** só possui estado quando materializada na forma de um **Objeto**.

Mas afinal, o que é uma Classe?

Definição de Classe

O conceito de **Classe** remete a um elemento com contexto e domínio muito bem definidos, o qual é composto de atributos (i.e., suas características intrínsecas), e métodos (i.e., suas operações) a representar o seu comportamento interno (i.e. dentro da classe) e externo (i.e. a interagir com elementos fora da classe). Uma **Classe** só possui estado quando materializada na forma de um **Objeto**.

Olha que curioso!

É preciso recordar que a Máquina Virtual Java tem a característica de **materializar até a “definição” das classes na forma de objetos**, a possibilitar, portanto, o armazenamento de estado e a execução de métodos diretamente das classes, durante a execução do programa.

Objeto, diga-me o que és!

Definição de Objeto

O conceito de **Objeto** relaciona-se com o conceito de **Classe** de forma direta, a implicar que um **Objeto** nada mais é do que a instância de uma **Classe** em um espaço de memória restrito, a possibilitar a definição de valores específicos para seus atributos e a execução de seus métodos de forma singular, ou seja, a execução de um método em um dado objeto não implica na execução do mesmo método em outro objeto da mesma classe, ou de qualquer outra classe.

Atributo, o que tu representas?

Definição de Atributo

O conceito de **Atributo** define uma característica intrínseca presente na classe, a qual é parte integrante de sua definição. A característica “**nome**” é um exemplo de atributo presente em uma classe denominada **Pessoa**. O valor de um atributo é parte da representação de estado de um determinado objeto.

Método, como estás a comportar-se?

Definição de Método

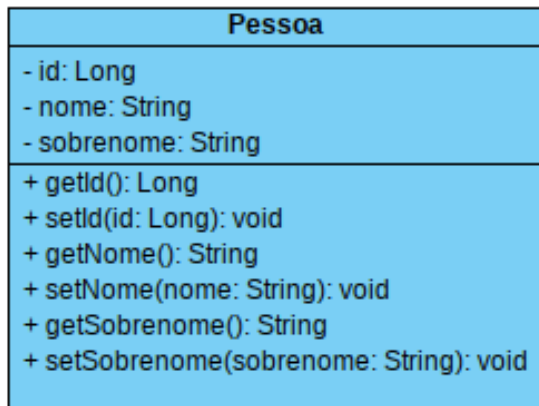
O conceito de **Método** define uma operação que pode ser executada dentro do domínio de uma classe/objeto, o qual define parte do comportamento observado em instâncias de uma dada classe.

Membro eu? Ou membro você? Atributos e Métodos

Membros de uma Classe

Atributos e métodos são conhecidos no paradigma de orientação a objetos como membros de uma classe.

Exemplos de Classes e Objetos - Classe 1



Exemplo de Classe Pessoa com três atributos

Exemplos de Classes e Objetos - Objeto 1

Instância de Pessoa (Objeto)
<ul style="list-style-type: none">- id = 1- nome = Jeferson- sobrenome = Souza
<ul style="list-style-type: none">+ getId(): Long+ setId(id: Long): void+ getNome(): String+ setNome(nome: String): void+ getSobrenome(): String+ setSobrenome(sobrenome: String): void

Exemplo de Instância da Classe Pessoa (Objeto) com três atributos

Exemplos de Classes e Objetos - Classe 2

Endereco
<ul style="list-style-type: none">- id: Long- cep: Long- logradouro: String- numero: Integer- bairro: String- cidade: Cidade
<ul style="list-style-type: none">+ getId(): Long+ setId(id: Long): void+ getCep(): Long+ setCep(cep: Long): void+ getLogradouro(): String+ setLogradouro(logradouro: String): void+ getCidade(): Cidade+ setCidade(cidade: Cidade): void

Exemplo de Classe Endereco com seis atributos

Exemplos de Classes e Objetos - Objeto 2

Instância de Endereco (Objeto)
<ul style="list-style-type: none">- id = 1- cep = 86256139- logradouro = Rua das Canastras- numero: = 66- bairro = Amor Infinito- cidade = Instância de Cidade (nome = Paraíso Seixal)
<ul style="list-style-type: none">+ getId(): Long+ setId(id: Long): void+ getCep(): Long+ setCep(cep: Long): void+ getLogradouro(): String+ setLogradouro(logradouro: String): void+ getCidade(): Cidade+ setCidade(cidade: Cidade): void

Exemplo de Instância da Classe Endereco (Objeto) com seis atributos

Introdução à Linguagem de Programação Java

Classes são as construções base da linguagem

Java é uma linguagem de programação que usa o paradigma orientado a objetos, o que significa que as classes são as construções base para o desenvolvimento de programas.

Objetos são instâncias das classes representadas em memória

Cada classe em Java é materializada por uma representação/instância de sua definição alocada em memória.

Exemplo Simples de Classe [Boyarsky&Selikoff, 2015]

```
public class Animal {  
  
}
```

Exemplo simples de classe escrita na linguagem de programação
Java

Exemplo Simples de Classe com Atributo [Boyarsky&Selikoff, 2015]

```
public class Animal {  
  
    String nome;  
  
}
```

Exemplo simples de classe (com atributo) escrita na linguagem de programação Java

Exemplo Simples de Classe com Atributo e Métodos [Boyarsky&Selikoff, 2015]

```
public class Animal {  
    String nome;  
    public String getNome(){  
        return nome;  
    }  
    public void setNome(String novoNome){  
        nome = novoNome;  
    }  
}
```

Exemplo simples de classe (com atributo e métodos) escrita na linguagem de programação Java

Exemplo Simples de Classe com Atributo e Métodos: **this** está dentro de mim!

```
public class Pessoa {  
    String nome;  
    public String getNome(){  
        return this.nome;  
    }  
    public void setNome(String nome){  
        this.nome = nome;  
    }  
}
```

Exemplo simples de classe (com atributo e métodos) escrita na linguagem de programação Java a usar o **this**

Por que usar o **this**?

Dentro de mim, aponta com o **this**!

O **this** permite apontar, de dentro da classe, para qualquer um de seus membros (atributos ou métodos) de forma precisa, a permitir diferenciar tais membros de definições presentes em outros escopos.

Por que usar o **this**?

Dentro de mim, aponta com o **this**!

O **this** permite apontar, de dentro da classe, para qualquer um de seus membros (atributos ou métodos) de forma precisa, a permitir diferenciar tais membros de definições presentes em outros escopos.

Convenção de uso do **this**

No caso de métodos *getters* e *setters* é convenção utilizar o **this** para identificar unicamente o atributo da classe, a diferenciá-lo, e.g., do parâmetro (de mesmo nome) presente nos métodos *setters*.

Por que usar o **this**?

Sem o **this**

```
public void setNome(String novoNome){  
    nome = novoNome;  
}  
}
```

O parâmetro do método *setNome()* possui uma denominação distinta da utilizada no atributo da classe.

Por que usar o **this**?

Com o **this**

```
public void setNome(String nome){  
    this.nome = nome;  
}  
}
```

O parâmetro do método *setNome()* possui a mesma denominação utilizada no atributo da classe.

Comentários em Java

Algo a comentar?

Comentários podem ser especificados de duas formas:

- ▶ comentários de uma linha, por meio do uso de duas barras (`//`);
- ▶ comentários de múltiplas linhas, por meio do uso dos delimitadores `/*` e `*/`.

Exemplos de Comentários

```
// Exemplo de Comentário de uma linha
```

```
/*  
Exemplo de comentário  
com múltiplas linhas  
a descrever algum  
aspecto do código.  
*/  
public class Pessoa {}
```

Exemplos de comentários com uma linha (//) e com múltiplas linhas (/* */) na linguagem de programação Java.

HelloWorld.java - Método main() para que existes?

```
public class HelloWorld {  
  
    public static void main(String ... args){  
        System.out.println("Hello World");  
    }  
}
```

De principal não basta só a Máquina Virtual!

O método *main()* existe para que a Máquina Virtual Java possa utilizá-lo como ponto de entrada de execução de um dado programa. Todo programa Java *Standard Edition* deve possuir, ao menos, uma classe com a declaração do método *main()*. Mais detalhes nos modificadores de acesso [Encapsulamento].

Como Compilar Um Programa Java

Compilado é mais pomposo!

Para compilar um programa java é necessário utilizar a ferramenta *javac*. A sintaxe básica é a seguinte:

`javac [opções] [arquivos com código-fonte java]` onde:

[opções] representam as opções adicionais a especificar parâmetros de compilação. Exemplo: `-classpath .`, o qual especifica que o diretório corrente (.) é utilizado como local de busca dos arquivos fonte especificados.

[arquivos com código fonte java] especificam os arquivos com código-fonte a ser compilado.

Mais informações [javac da versão 11:] <https://docs.oracle.com/en/java/javase/11/tools/javac.html>

Como Compilar Um Programa Java (Continuação)

Exemplo

```
javac -classpath . HelloWorld.java
```

O resultado da compilação gera um arquivo com extensão **.class**, o qual contém um código intermediário (i.e., java *bytecode*) para ser interpretado pela Máquina Virtual Java, e executado na arquitetura de *hardware* na qual a referida Máquina Virtual está a ser executada.

Como Executar Um Programa Java

Vamos lá que está na hora da execução!

Para executar um programa java é necessário utilizar a ferramenta *java*. A sintaxe básica é a seguinte:

java [opções] [nome da classe com o método main()] [argumentos]
onde:

[opções] representam as opções adicionais a especificar parâmetros de execução. Exemplo: `-classpath .`, o qual especifica que o diretório corrente (.) é utilizado como local de busca das classes necessárias para execução.

[nome da classe com o método main()] especifica o nome da classe que contém o método `main()`.

Como Executar Um Programa Java (Continuação)

Vamos lá que está na hora da execução!

[**argumentos**] especificam os argumentos passados como parâmetro na execução do método *main()*.

Mais informações [java da versão 11:] <https://docs.oracle.com/en/java/javase/11/tools/java.html>

Exemplo

```
java -classpath . HelloWorld
```

O resultado é a interpretação do arquivo **HelloWorld.class** e a respectiva execução das instruções presentes no método *main()* da referida classe.

Tipos Primitivos em Java

Palavra-resevada	Tipo	Exemplo
boolean	true ou false	true
byte	inteiro de 8-bits	8
short	inteiro de 16-bits	16
int	inteiro de 32-bits	32
long	inteiro de 64-bits	64
float	ponto-flutuante de 32-bits	123.45f
double	ponto-flutuante de 64-bits	123.456
char	caractere unicode de 16-bits	'a'

Tabela de Tipos Primitivos em Java [Boyarsky&Selikoff, 2015]

Palavras Reservadas da linguagem Java

Tabela de Palavras Reservadas

abstract	continue	for	new	switch
assert	default	if	package	synchronized

Declaração de Identificadores

Regras de Declaração de Identificadores

Existem três regras básicas que devem ser seguidas na declaração de identificadores, ou seja, nomes de classes, atributos, métodos, e variáveis:

1. Um nome deve começar com uma letra, ou com um dos símbolos \$ ou _;
2. Os caracteres posteriores ao primeiro podem incluir letras, números, e alguns caracteres especiais tais como o \$ e o _;
3. Nenhum nome pode ser idêntico a uma palavra reservada utilizada pela linguagem Java.

Declaração de Identificadores

Regras de Declaração de Identificadores

Existem três regras básicas que devem ser seguidas na declaração de identificadores, ou seja, nomes de classes, atributos, métodos, e variáveis:

1. Um nome deve começar com uma letra, ou com um dos símbolos \$ ou _;
2. Os caracteres posteriores ao primeiro podem incluir letras, números, e alguns caracteres especiais tais como o \$ e o _;
3. Nenhum nome pode ser idêntico a uma palavra reservada utilizada pela linguagem Java.

Bibliografia

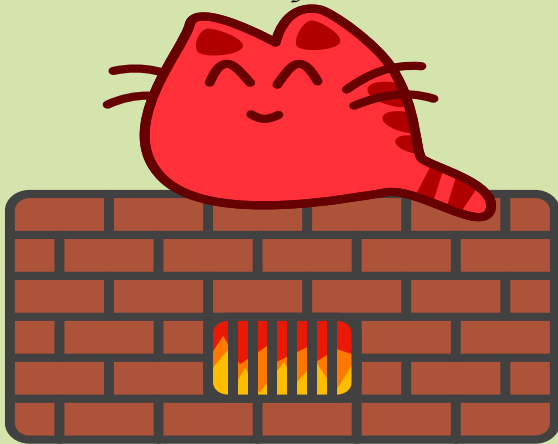


BOYARSKY, J. and SELIKOFF, S. *“Oracle Certified Associate Java SE 8 Programmer I: Study Guide”*. Sybex. Indianápolis, Indiana. 2015.



BOYARSKY, J. and SELIKOFF, S. *“Oracle Certified Associate Java SE 8 Programmer II: Study Guide”*. Sybex. Indianápolis, Indiana. 2016.

That's it folks!



Thank you for your attention!