

Introdução ao ecossistema JAVA

Java é uma linguagem de programação e plataforma computacional lançada pela primeira vez pela Sun Microsystems em 1995

Características:

- Compilada
- Interpretada
- Fortemente tipada
- Linguagem de alto nível
- Executada em uma máquina virtual

JVM (Java Virtual Machine) é uma máquina virtual responsável pela tradução dos ByteCodes oriundo do compilador Javac (Java Compiler) em códigos de máquina de cada sistema operacional.

Características:

- Execução das pilas
- Gerenciamento de memória
- Gerenciamento de threads
- Otimização de código (Compilação JIT - Just In Time)
- Garbage Collector (GC)

Quais as diferenças entre JRE e JDK?

- JRE (Java Runtime Environment) - responsável por executar os programas em Java.
- JDK (Java Development Kit) - utilitários que permite o desenvolvimento de programas em Java. Já possui a JVM para executar os programas.

Quais os tipos de plataformas Java?

- Java SE(Java Standard Edition) - contém as especificações do java e pode ser implementado por diversas empresas como Oracle, OpenJDK, Azul Zulu, etc.
- Java EE(Java Enterprise Edition) - contém todas as especificações do Java SE e um número de programas úteis para que executam em servidores. Em 2019 foi renomeado para Jakarta EE.
- Java ME (Java Micro Edition) - contém especificações para desenvolvimento de programas para dispositivos pequenos como celulares, PDAs, entre outros.

Quais as implementações Java SE:

- OpenJDK
- Oracle JDK
- AdoptOpenJDK
- Amazon Corretto
- GraalVm CE
- Azul Zulu

Comandos

- `jabba ls-remote` (para listar versões)
- `jabba install openjdk@1.11` (para instalar versão)
- `jabba use openjdk@1.11` (para usar versão)
- `java - version` (para ver versão que está sendo utilizada)
- `mkdir` (fazer pasta)
- `cd` (entrar na pasta)
- `/mnt/c/` (para abrir pasta do windows)
- `rmdir` (para poder apagar)
- `jabba use openjdk@1.11` (compilar)
- `javac -d target/ -sourcepath src nome da pasta` (compilar)
- `cd target/`
- `java nome da pasta` (para executar)

As 52 palavras reservadas do Java

Conheça o significado de 52 palavras reservadas do Java.

Modificadores de acesso

private: acesso apenas dentro da classe

protected: acesso por classes no mesmo pacote e subclasses

public: acesso de qualquer classe

Modificadores de classes, variáveis ou métodos

abstract: classe que não pode ser instanciada ou método que precisa ser implementado por uma subclasse não abstrata

class: especifica uma classe

extends: indica a superclasse que a subclasse está estendendo

final: impossibilita que uma classe seja estendida, que um método seja sobrescrito ou que uma variável seja reinicializada

implements: indica as interfaces que uma classe irá implementar

interface: especifica uma interface

native: indica que um método está escrito em uma linguagem dependente de plataforma, como o C

new: instancia um novo objeto, chamando seu construtor

static: faz um método ou variável pertencer à classe ao invés de às instâncias

strictfp: usado em frente a um método ou classe para indicar que os números de ponto flutuante seguirão as regras de ponto flutuante em todas as expressões

synchronized: indica que um método só pode ser acessado por uma thread de cada vez

transient: impede a serialização de campos

volatile: indica que uma variável pode ser alterada durante o uso de threads

Controle de fluxo dentro de um bloco de código

break: sai do bloco de código em que ele está

case: executa um bloco de código dependendo do teste do switch

continue: pula a execução do código que viria após essa linha e vai para a próxima passagem do loop

default: executa esse bloco de código caso nenhum dos teste de switch-case seja verdadeiro

do: executa um bloco de código uma vez, e então realiza um teste em conjunto com o while para determinar se o bloco deverá ser executado novamente

else: executa um bloco de código alternativo caso o teste if seja falso

for: usado para realizar um loop condicional de um bloco de código

if: usado para realizar um teste lógico de verdadeiro o falso

instanceof: determina se um objeto é uma instância de determinada classe, super classe ou interface

return: retorna de um método sem executar qualquer código que venha depois desta linha (também pode retornar uma variável)

switch: indica a variável a ser comparada nas expressões case

while: executa um bloco de código repetidamente enquanto a condição for verdadeira

Tratamento de erros

assert: testa uma expressão condicional para verificar uma suposição do programador

catch: declara o bloco de código usado para tratar uma exceção

finally: bloco de código, após um try-catch, que é executado independentemente do fluxo de programa seguido ao lidar com uma exceção

throw: usado para passar uma exceção para o método que o chamou

throws: indica que um método pode passar uma exceção para o método que o chamou

try: bloco de código que tentará ser executado, mas que pode causar uma exceção

Controle de pacotes

import: importa pacotes ou classes para dentro do código

package: especifica a que pacote todas as classes de um arquivo pertencem

Primitivos

boolean: um valor indicando verdadeiro ou falso

byte: um inteiro de 8 bits (signed)

char: um caracter unicode (16-bit unsigned)

double: um número de ponto flutuante de 64 bits (signed)

float: um número de ponto flutuante de 32 bits (signed)

int: um inteiro de 32 bits (signed)

long: um inteiro de 64 bits (signed)

short: um inteiro de 16 bits (signed)

Variáveis de referência

super: refere-se a superclasse imediata

this: refere-se a instância atual do objeto

Retorno de um método

void: indica que o método não tem retorno

Palavras reservadas não utilizadas

const: Não utilize para declarar constantes; use public static final

goto: não implementada na linguagem Java por ser considerada prejudicial

Declaração de classes

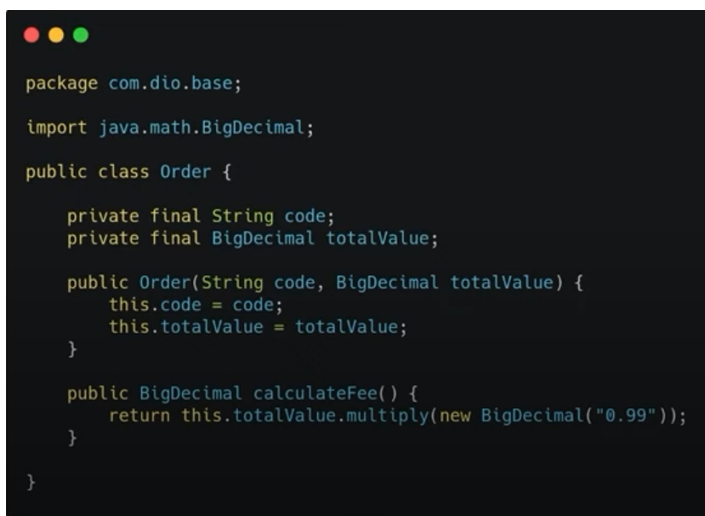
Estrutura básica de uma classe:

```
public class Order {  
    // atributos e metodos  
}
```

Estrutura com atributos e métodos:

```
public class Order {  
    // atributos  
    private final String code;  
    private final BigDecimal totalValue;  
  
    // metodos  
    public BigDecimal calculateFee() {  
    }  
}
```

Estrutura completa de uma classe:

A screenshot of a code editor with a dark background and light-colored text. The code is in Java and defines a class named 'Order'. It includes a package declaration, an import statement for 'BigDecimal', and the class definition with two private final attributes, a constructor, and a method to calculate a fee.

```
package com.dio.base;

import java.math.BigDecimal;

public class Order {

    private final String code;
    private final BigDecimal totalValue;

    public Order(String code, BigDecimal totalValue) {
        this.code = code;
        this.totalValue = totalValue;
    }

    public BigDecimal calculateFee() {
        return this.totalValue.multiply(new BigDecimal("0.99"));
    }

}
```

Modificadores de acesso

Tipos de modificadores:

- public - Qualquer classe de qualquer pacote poderá acessar o atributo ou método.
- protected - Qualquer classe definida no mesmo pacote ou subclasse.
- Sem modificador - Apenas classes definidas no mesmo pacote.
- private - Apenas a própria classe.

Métodos

São funções que definem o comportamento de uma classe

Tipos:

- Métodos construtores - Definem como uma classe será instanciada "construída"
- Métodos comuns - Definem comportamentos que podem ou não estar atribuídos às regras de negócio. Ex: calcular taxas de um pedido, etc.

Estrutura de condição

Estrutura de condição é responsável por fazer o desvio do fluxo de execução do código de acordo com uma dada condição.

Tipos de estrutura de condição:

if - else:

```
// imports

public class Order {

    // outros atributos

    private final double totalValue;

    // outros metodos

    public double calculateFee() {
        if (this.totalValue > 100) {
            return this.totalValue * 0.99;
        } else {
            return this.totalValue;
        }
    }
}
```

switch - case :

```
// imports

public class Order {

    // outros atributos

    private final int totalValue;

    // outros metodos

    public double calculateFee() {
        switch (this.totalValue) {
            case 100:
                return totalValue * 0.99;
            case 200:
                return totalValue * 1.99;
            default:
                return totalValue;
        }
    }
}
```

Estrutura de repetição.

Estrutura de repetição é responsável por executar repetitivamente uma instrução ou um bloco de instruções até que uma condição esteja sendo satisfeita.

Tipos de estrutura de repetição:

- while

```
// imports

public class Order {

    // outros atributos

    private String[] items;

    // outros metodos

    public void printItems() {
        int i = 0;

        while (i < items.length) {
            System.out.println(items[i]);
            i++;
        }
    }
}
```


- do - while

```
// imports

public class Order {

    // outros atributos

    private String[] items;

    // outros metodos

    public void printItems() {
        int i = 0;

        do {
            System.out.println(items[i]);
            i++;
        } while (i < items.length);
    }
}
```

- for

```
// imports

public class Order {

    // outros atributos

    private String[] items;

    // outros metodos

    public void printItems() {

        for (int i = 0; i < items.length; i++) {
            System.out.println(items[i]);
        }

    }

}
```

- enhanced for

```
// imports

public class Order {

    // outros atributos

    private String[] items;

    // outros metodos

    public void printItems() {

        for (String i : items) {
            System.out.println(i);
        }

    }

}
```

Documentação java

Comentários em linha

```
public class Order {  
  
    // esse é um comentário de linha para informar que esse atributo representa o código do pedido  
    private final String code;  
    private final BigDecimal totalValue;  
  
    public Order(String code, BigDecimal totalValue) {  
        this.code = code;  
        this.totalValue = totalValue;  
    }  
}
```

Comentário em bloco

```
public class Order {  
  
    /**  
     * Esse é um comentário em bloco para informar que esse atributo representa o código do pedido  
     */  
    private final String code;  
    private final BigDecimal totalValue;  
  
    public Order(String code, BigDecimal totalValue) {  
        this.code = code;  
        this.totalValue = totalValue;  
    }  
}
```

Javadoc

É uma ferramenta para documentação no formato HTML que se baseia nos comentários do código-fonte. Os comentários precisam conter tags para que a documentação fique legível.

Tipos de tags :

- `@author` - Especifica o autor da classe ou do método.
- `@deprecated` - Identifica classes ou métodos obsoletos.
- `@link` - Possibilita a definição de um link para um outro documento local ou remoto através de um URL.
- `@param` - Descreve um parâmetro que será passado a um método.
- `@return` - Descreve qual o tipo de retorno de um método.
- `@see` - Associa a outras classes ou métodos.
- `@since` - Descreve desde quando uma classe ou métodos foi adicionado.
- `@throws` - Descreve os tipos de exceções que podem ser lançadas por um método.
- `@version` - Descreve a versão da classe ou método.

