LINGUAGEM DE PROGRAMAÇÃO JAVA

# Java (Conceitual)

Java é uma linguagem de programação orientada à objetos (POO). Java é uma tecnologia que consiste tanto em uma linguagem de programação quanto em uma plataforma de software (IBM).

## Java Standard Edition

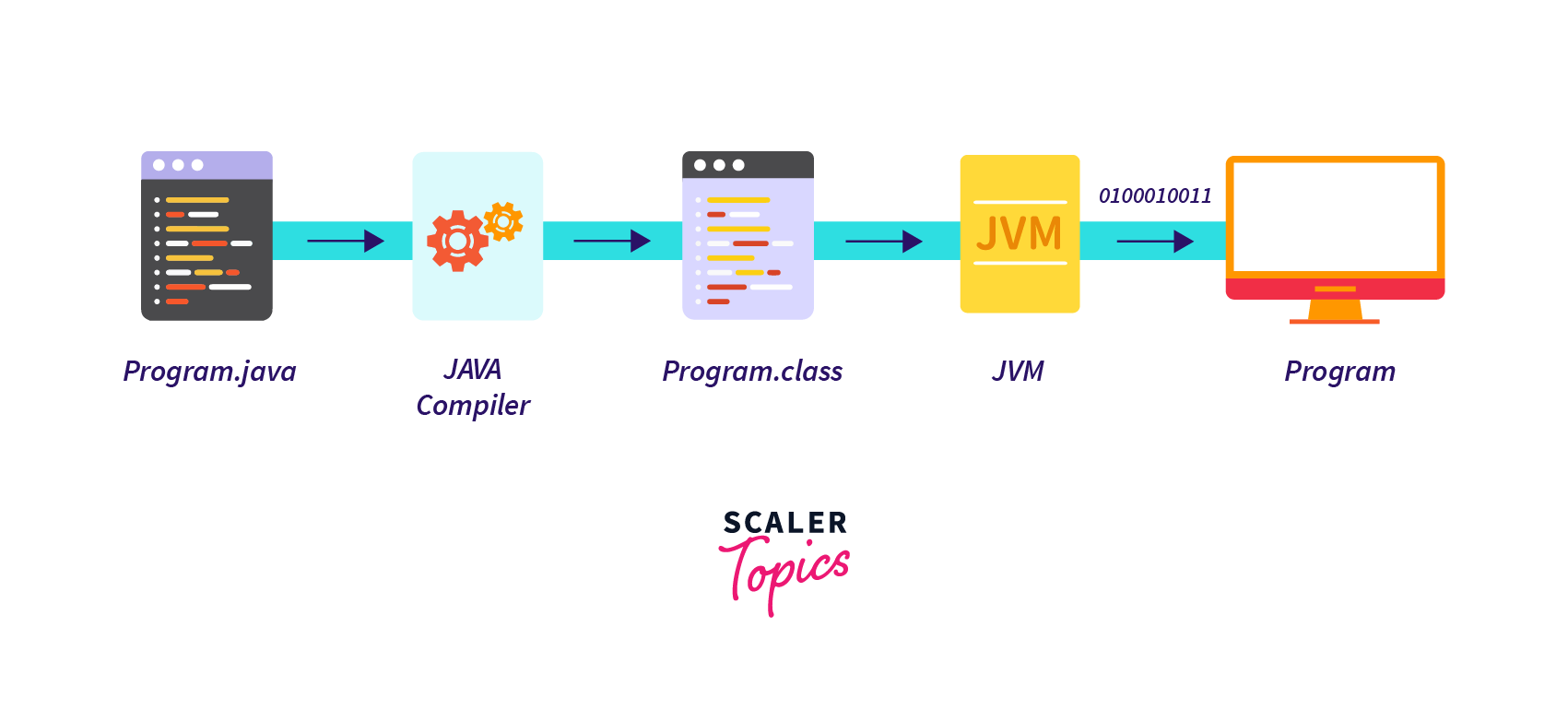
O Java Standard Edition é a plataforma central de programação Java. Ele contém todas as bibliotecas e APIs que qualquer programador precisa para o desenvolvimento java (AMAZON).

## Java versus Javascript

O Javascript não precisa ser compilado, enquanto que o código Java precisa. Além disso, o Javascript só pode ser executado em navegadores da web, ao passo que o Java pode ser executado em qualquer lugar (IBM).

## Compilação e Execução de um Programa Java

O processo de execução de um programa java.



(SCALER TOPICS)

Para criar um aplicativo usando Java, é necessário fazer o download do JDK (Java Development Kit), disponível para Windows, macOS e Linux. O programa é escrito na linguagem de programação Java e, em seguida, um compilador transforma o programa em bytecode Java, que é o conjunto de instruções do JVM (Java Virtual Machine) que faz parte do JRE (Java Runtime Environment). O bytecode Java é executado sem modificação em qualquer sistema que oferece suporte às JVMs, permitindo que seu código Java seja executado em qualquer lugar (IBM).

## Compilada e Interpretada

Java é uma linguagem compilada e interpretada (ChatGPT).

Primeiramente, o código fonte Java é compilado pelo compilador Java, que gera o bytecode Java. Esse bytecode é uma representação de baixo nível do código fonte Java que pode ser interpretada por uma máquina virtual Java (JVM - Java Virtual Machine) (ChatGPT).

Quando o bytecode é executado pela JVM, ele é interpretado e traduzido em código de máquina nativo do sistema em que a JVM está sendo executada. Esse processo é chamado de interpretação just-in-time (JIT), porque o bytecode é interpretado e traduzido em tempo de execução, conforme necessário (ChatGPT).

Assim, podemos dizer que Java é uma linguagem híbrida, pois envolve tanto a compilação quanto a interpretação durante o processo de execução do programa (ChatGPT).

Sem uma máquina virtual Java não é possível executar um programa escrito em Java diretamente no sistema operacional do Windows, porque o código não é compilado em uma linguagem de máquina nativa do sistema (ChatGPT).

O código Java é compilado em bytecode Java, que é uma linguagem intermediária que não é diretamente executada pelo sistema operacional do Windows. Para executar um programa Java, é necessário ter uma máquina virtual Java (JVM) instalada no sistema (ChatGPT).

A JVM é capaz de interpretar o bytecode Java e traduzi-lo em instruções executáveis em tempo de execução, permitindo que o programa Java seja executado em diferentes sistemas operacionais. Dessa forma, a JVM atua como uma camada de abstração que permite a portabilidade de programas Java para diferentes plataformas (ChatGPT).

Portanto, para executar um programa Java em um sistema operacional Windows, é necessário ter a JVM instalada nesse sistema. A maioria dos sistemas operacionais modernos já possui a JVM instalada por padrão, mas se não houver, o usuário precisará instalar a JVM manualmente antes de executar o programa Java (ChatGPT).

# Paradigmas da Programação

Paradigmas da programação são categorias de abordagens metodológicas na estruturação do código (algoritmo) para o desenvolvimento de programas computacionais (softwares) (AUTOR).

## Paradigma Orientado a Objetos

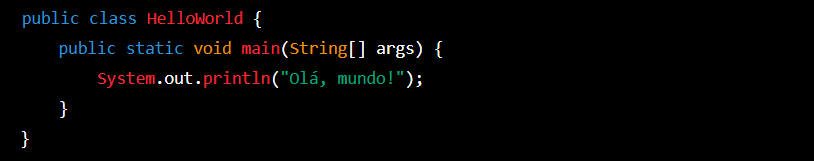
No paradigma orientado a objetos, os objetos são instâncias (entidades) de uma classe de dados com atributos específicos que descrevem as características da instância. Os métodos são instruções que manipulam os atributos pertencentes a essas classes para formar os objetos (AUTOR).

## Java vs. Paradigma Imperativo

Em Java é necessário criar uma classe para imprimir uma mensagem simples em um terminal. Isso ocorre porque Java é uma linguagem de programação orientada a objetos e tudo em Java é baseado em classes e objetos (ChatGPT).

No Java, todas as funções e variáveis devem estar definidas dentro de uma classe. Portanto, para imprimir uma mensagem simples em um terminal, você precisaria criar uma classe Java com um método que contém a instrução de impressão (ChatGPT).

Por exemplo, para imprimir a mensagem "Olá, mundo!" no terminal, deve-se criar uma classe, neste caso chamada "HelloWorld", com um método "main" que imprime uma mensagem na saída padrão:



O método "main" é o ponto de entrada para o programa Java, e é onde a execução do programa começa. Neste exemplo, o método "main" usa a classe System.out para imprimir a mensagem no terminal (ChatGPT).

Em resumo, em Java é necessário criar uma classe para realizar qualquer ação, até mesmo imprimir uma mensagem simples em um terminal, devido à natureza orientada a objetos da linguagem (ChatGPT).

Geralmente, o primeiro método de uma classe é chamado de método construtor, serve para gerar o objeto daquela classe, porém como Java é uma Linguagem OOP, ele é utilizado/criado como static main para possibilitar a execução do programa, porque todo OOP só trabalha com um objeto. Por isso a classe inicial do programa é chamada de classe de aplicativo (AUTOR).

# Elementos e Conceitos Fundamentais da Programação Orientada a Objetos

## Métodos e Classes

Para realizar uma tarefa em um programa é necessário um **método**. O método armazena as declarações do programa que, na verdade, executam as tarefas; além disso, ele oculta essas declarações do usuário. No Java, criamos uma unidade de programa chamada **classe** para armazenar o conjunto de métodos que executam as tarefas dela (DEITEL, pg. 9).

## Variáveis

Os programas lembram-se dos números e de outros dados na memória do computador e os acessam por meios de elementos de programa chamados variáveis (DEITEL, pg. 36).

Uma variável é uma posição na memória do computador na qual um valor pode ser armazenado para utilização posterior em um programa. Todas as variáveis Java devem ser declaradas com um nome e um tipo antes que elas possam ser utilizadas. O tipo de variável especifica o tipo de informação armazenada nessa posição da memória (DEITEL, pg. 37).

Principais tipos de variáveis em Java são String, int, float, char, boolean (W3Schools).

## Instanciação

Deve-se construir um objeto de uma classe antes que um programa possa executar as tarefas que os métodos da classe definem. O processo para fazer isso é chamado instanciação. Um objeto é então referido como uma **instância** da sua classe (DEITEL, pg. 9).

## Atributos e Chamadas de Método

Os atributos são especificados pelas **variáveis de instância** da classe (DEITEL, pg. 9).

## Encapsulamento

Classes encapsulam (contêm) seus atributos e métodos (DEITEL, pg. 9).

## Herança

Uma nova classe de objetos pode ser criada convenientemente por meio de herança, ela (chamada de subclasse) começa com as características de uma classe existente (chamada superclasse), possivelmente personalizando-as e adicionando aspectos próprios (DEITEL, pg. 9).

# Etapas de Desenvolvimento Java Típico

## Fase 1

Escrever um programa Java (em geral referido como código-fonte) utilizando o editor (IDE ou editor de texto). Arquivos de código-fonte Java recebem um nome que termina com a extensão .java (DEITEL, pg. 13).

## Fase 2

Utiliza-se o comando javac (o compilador Java) para compilar um programa através da janela de comando do sistema (DEITEL, pg. 14).



O compilador produz um arquivo com extensão .class que contém a versão compilada do código-fonte. IDEs fornecem um item de menu que faz o comando javac (DEITEL, pg. 14).

O compilador Java converte o código-fonte Java em bytecodes que representam as tarefas a serem executadas na fase de execução. O Java Virtual Machine (JVM), uma parte do JDK e a base da plataforma Java, executa bytecodes (DEITEL, pg. 14).

Diferentemente das instruções em linguagem de máquina, que são dependentes de plataforma (isto é, de hardware específico de computador), instruções bytecode são independentes de plataforma. Portanto, os bytecodes do Java são portáveis — sem recompilar o código-fonte, as mesmas instruções em bytecodes podem ser executadas em qualquer plataforma contendo uma JVM que entende a versão do Java na qual os bytecodes foram compilados. A JVM é invocada pelo comando java. Por exemplo, para executar um aplicativo Java chamado Welcome, digita-se java e o nome do programa



em uma janela de comando para invocar a JVM, que iniciaria os passos necessários a fim de executar o aplicativo (DEITEL, pg. 14).

## Fase 3

[...]

## Fase 4

[...]

## Fase 5

[...]

## Como Compilar e Executar um Programa Java pela Janela de Comando do Sistema

[...] (DEITEL, pg. 32)

# Linguagem de Programação Java – Características da Sintaxe

## Sintaxe de Instruções

A maioria das instruções acabam com ponto e vírgula, ; (DEITEL, pg. 31).

## Sensitive Case

Java faz distinção entre maiúsculas e minúsculas (DEITEL, pg. 30).

## Fortemente Tipada

[...]

# DESENVOLVIMENTO DE CÓDIGO EM JAVA

# Comentários

[...]

## Comentário de Fim de Linha

Comentário começa com //, indicando que é um comentário de fim de linha (DEITEL, pg. 29).

## Comentário Múltiplas Linhas

O Java também tem comentários tradicionais, que podem ser distribuídos ao longo de múltiplas linhas, que começam e terminam com delimitadores /\* e \*/ (DEITEL, pg. 29).

# Declarações

## Declarar uma Variável versus Inicializar uma Variável

Declarar uma variável significa informar ao compilador que uma variável com um determinado nome e tipo de dado será utilizada em um programa. A declaração de uma variável reserva espaço na memória para armazenar valores do tipo especificado (ChatGPT).

Inicializar uma variável é atribuir um valor inicial a uma variável declarada. A inicialização ocorre após a declaração e é feita usando o operador de atribuição '=' (ChatGPT).

## Declaração de Classe

Todo programa Java consiste em pelo menos uma classe. A palavra-chave class introduz uma declaração de classe e é imediatamente seguida pelo nome da classe. Palavras-chave são sempre escritas com todas as letras minúsculas (DEITEL, pg. 30).

O nome da classe deve ser exatamente igual ao nome do arquivo. Por convenção, os nomes de classes iniciam com letra maiúscula e apresentam a letra inicial de cada (DEITEL, pg. 30).

A chave esquerda, {, inicia o corpo de cada declaração de classe. A chave direita correspondente, }, deve terminar cada declaração de classe (DEITEL, pg. 30).

## Declaração de Método

Para um aplicativo Java, um dos métodos deve ser chamado *main*.

## Declarações *Import*

Classes predefinidas e relacionadas são agrupadas em pacotes e, coletivamente, são chamados de biblioteca de classes Java, ou Java Application Programming Interface - Java API (DEITEL, pg. 37).

# Estrutura de uma Classe

## Variável de Instância

Em Java, uma variável de instância é uma variável declarada dentro de uma classe, mas fora de qualquer método, construtor ou bloco estático. Ela é associada a cada instância (objeto) da classe e possui um valor único para cada objeto individual (ChatGPT).

É importante destacar que existem outros tipos de variáveis em Java, como variáveis locais (declaradas dentro de um método ou bloco) e variáveis estáticas (declaradas com o modificador static e compartilhadas por todas as instâncias da classe). Esses tipos de variáveis têm escopo e comportamento diferentes em relação às variáveis de instância (ChatGPT).

Resumindo, na prática, "atributo de um objeto" e "variável de instância" se referem à mesma coisa: uma variável declarada dentro de uma classe, fora de qualquer método, construtor ou bloco estático, que está associada a cada instância individual da classe (ChatGPT).

As variáveis ou métodos declarados com o modificador de acesso private só são acessíveis a métodos da classe em que isso ocorre (DEITEL, pg. 58).

## Método

[...] (DEITEL, pg. 58)

# Estruturas de Seleção e Repetição

## Instrução de Controle

[...]

## Operador Ternário

[...] (DEITEL, pg. 136).

# Método Main

## Assinatura do Método *Main*

"public static void main (String [] args)" é a assinatura do método principal (*main*) de uma aplicação Java. É por meio deste método que a execução do programa Java é iniciada (ChatGPT).

Cada aplicação Java deve ter um método *main* com essa assinatura exata, caso contrário o programa não será executado corretamente.

Aqui está uma explicação mais detalhada de cada parte da assinatura do método main:

* "public" é um modificador de acesso que indica que o método main é público e pode ser acessado por outras classes.
* "static" é um modificador de classe que indica que o método main é um método de classe (e não de objeto) e pode ser chamado diretamente pela classe, sem a necessidade de instanciar um objeto.
* "void" é um tipo de retorno que indica que o método main não retorna nenhum valor.
* "main" é o nome do método principal. Esse nome é fixo e não pode ser alterado.
* "(String [] args)" é a lista de parâmetros do método principal. "String []" é um array de strings que pode ser usado para passar argumentos para o programa Java no momento da execução. O argumento "args" é um nome de variável que pode ser alterado, mas é uma convenção usar o nome "args" para manter a consistência com outros programas Java.

Em resumo, a assinatura do método main em Java é uma convenção bem estabelecida que indica ao compilador e ao interpretador Java qual método deve ser executado primeiro ao iniciar a aplicação (ChatGPT).

## Um Método *Main* (principal) pode ser chamado sem a Necessidade de Instanciar um Objeto

Um método principal em Java é definido com o modificador "static", que significa que ele pertence à classe em que está definido, e não a um objeto específico criado a partir dessa classe. Isso significa que ele pode ser chamado diretamente pela classe, sem a necessidade de criar uma instância da classe (ChatGPT).

Quando o método principal é chamado pelo sistema operacional, o interpretador Java carrega a classe que contém o método principal e o executa. Como o método principal é estático, ele pode ser executado sem que uma instância da classe seja criada, pois ele não depende de nenhum estado de objeto específico (ChatGPT).

Isso torna o método principal um ponto de entrada natural para a execução de programas Java, pois ele pode ser executado diretamente pela linha de comando, sem a necessidade de criar objetos ou inicializar o estado de uma classe. Além disso, o método principal é geralmente o primeiro método a ser executado em um programa Java, e é usado para iniciar a execução do programa e configurar qualquer estado inicial necessário (ChatGPT).

## Utilizando Argumentos de Linha de Comando

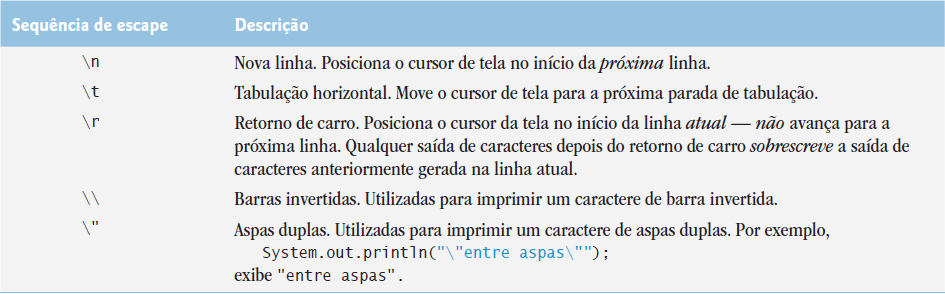
É possível passar argumentos da linha de comando para um aplicativo por meio do parâmetro *String*[] do método *main*, que recebe um *array* de *Strings*. Por convenção, esse parâmetro é chamado *args*.

[...] (DEITEL, pg.221).

# Exibição de Texto

## Caracteres de Nova Linha e Caracteres de Escape

[...] (DEITEL, pg. 34).



(DEITEL, pg. 35)

## Exibindo Texto com *Printf*

O método *System.out.printf* (f significa “formato”) exibe os dados formatados. argumentos. Quando um método exige múltiplos argumentos, estes são colocados em uma lista separada por vírgulas (DEITEL, pg. 35).

## Especificadores de Formato

[...] (DEITEL, pg. 35)

## Explicação do Comando *System.out.println*

Em Java, *System* é uma classe predefinida fornecida pelo JDK (Java Development Kit) que fornece acesso a várias funcionalidades e recursos do sistema, como entrada/saída, propriedades do sistema e muito mais. É parte da biblioteca padrão do Java e está localizada no pacote *java.lang* (ChatGPT).

*Out* é um objeto estático da classe *System* que representa a saída padrão do programa. É uma instância da classe *PrintStream*, que fornece métodos para exibir dados no console ou em outro fluxo de saída. O objeto *out* é usado para realizar operações de impressão no console (ChatGPT).

Dessa forma, quando usamos a expressão *System.out* em Java, estamos referenciando o objeto out da classe System, que nos permite realizar operações de saída no console. (ChatGPT).

*Println* é um método da classe *PrintStream*, que é acessado por meio da variável estática *out* da classe System. O método *println* é usado para imprimir uma linha de texto no console, seguido por uma quebra de linha (ChatGPT).

Ao usar *System.out.println("Hello, World!");*, estamos invocando o método *println* da classe *PrintStream* por meio da variável estática out da classe System. Isso resultará na impressão da mensagem "Hello, World!" no console, seguido por uma quebra de linha (ChatGPT).

## Placeholders e Especificadores de Formato

[...]

## Concatenar textos com o operador +

Também pode-se utilizar o operador + para concatenar *strings* com *placeholders*. Por exemplo:

String nome = "Carlos";

int valor = 100;

String mensagem = "Olá, " + nome + "! O valor é: " + valor;

System.out.println(mensagem);

Neste exemplo, os valores de nome e valor são concatenados às partes fixas da *string* usando o operador +.

[...] (DEITEL, pg. 163)

# Leitura de Dados

## Classe Scanner

O objeto da classe Scanner é usado apenas para ler dados de entrada inseridos pelo usuário e não armazenar, portanto depois são transferidos para uma variável ou outro objeto.

O objeto Scanner em Java não armazena dados, ele apenas inicializa a leitura de dados de entrada a partir de um determinado fluxo de entrada, que pode ser o console (System.in) ou um arquivo, por exemplo (ChatGPT).

Antes de utilizar um Scanner é necessário criá-lo e verificar a origem dos dados (DEITEL, pg. 38).

Após criar o objeto Scanner e associá-lo ao fluxo de entrada desejado, é possível utilizar os métodos de leitura disponíveis para ler os dados que foram digitados pelo usuário ou que estão armazenados no arquivo (ChatGPT).

Em Java, para criar um objeto de uma classe é necessário utilizar o operador new e chamar um dos construtores da classe. No caso da classe Scanner, o construtor utilizado para ler dados de entrada do console é o Scanner(System.in).

## Método nextLine()

O método nextLine() é então utilizado para ler a linha de entrada digitada pelo usuário e atribuí-la à variável especificada.

Na declaração de um objeto Scanner, exemplo:

Scanner input = new Scanner(System.in);

O primeiro Scanner é referência a classe.

O segundo, ao método construtor.

System.in é um objeto.

Essa linha de código em Java cria um objeto da classe Scanner e o associa à entrada padrão do sistema (System.in).

O objeto Scanner é utilizado para ler entradas a partir do console (também conhecido como linha de comando) ou de outro fluxo de entrada. Ao associá-lo à entrada padrão do sistema, podemos ler os dados que são digitados pelo usuário no console.

Após criar o objeto Scanner, é comum utilizá-lo para ler os dados de entrada digitados pelo usuário. Por exemplo, podemos utilizar o método nextLine() do objeto Scanner para ler uma string digitada pelo usuário:

# Palavras-chave *this*

*public class Pessoa {*

*private String nome; //variável de instância (atributo)*

*public void setNome(String nome) {*

*this.nome = nome;*

*}*

*}*

Neste exemplo, *this.nome* refere-se à variável de instância nome, e nome refere-se ao parâmetro do método. Ao usar *this.nome*, estamos indicando que queremos atribuir o valor do parâmetro à variável de instância nome (ChatGPT).

User

Por que o objeto System.out não foi instanciado como o Scanner?

ChatGPT

O objeto System.out não é instanciado com o operador new, como ocorre com o objeto da classe Scanner, porque ele não é uma instância de uma classe.

Na verdade, System.out é um objeto predefinido (ou pré-instanciado) da classe PrintStream, que representa a saída padrão do sistema. Em outras palavras, ele é um objeto que já existe na memória e que pode ser utilizado diretamente pelo programa.

Dessa forma, ao utilizar o objeto System.out, não é necessário criar uma nova instância da classe PrintStream com o operador new, como ocorre com outras classes que precisam ser instanciadas para serem utilizadas. O objeto System.out já está disponível para uso em qualquer lugar do programa.

User

number1 = input.nextInt();

ChatGPT

Este trecho de código em Java utiliza o método nextInt() da classe Scanner para ler um número inteiro da entrada do usuário e armazená-lo na variável number1. O método nextInt() faz parte da interface java.util.Scanner, que é usada para ler entradas de diversos tipos de dados a partir do teclado ou de outras fontes de entrada.

O método nextInt() retorna o próximo número inteiro disponível na entrada, aguardando até que o usuário insira um valor válido. Se a entrada não for um número inteiro válido, uma exceção será lançada. Para evitar isso, é recomendado o uso do método hasNextInt() para verificar se o próximo valor na entrada é um inteiro válido antes de chamar o método nextInt().

User

O que são objetos predefinidos em um código Java?

ChatGPT

Em Java, objetos predefinidos são objetos que são criados automaticamente pelo sistema e estão disponíveis para uso em todos os programas Java, sem a necessidade de criar instâncias desses objetos explicitamente. Esses objetos são fornecidos como parte das classes padrão da linguagem Java e são frequentemente referidos como objetos padrão ou objetos embutidos.

Alguns exemplos de objetos predefinidos em Java incluem:

System: uma classe que fornece acesso aos recursos do sistema, como entrada e saída de dados e a hora do sistema.

String: uma classe que representa sequências de caracteres.

Math: uma classe que fornece métodos para operações matemáticas, como raiz quadrada, potência, seno e cosseno.

Boolean: uma classe que representa valores booleanos (verdadeiro ou falso).

Integer: uma classe que representa números inteiros.

Esses objetos predefinidos são usados ​​com frequência em programas Java e são importantes para o desenvolvimento eficiente e eficaz de aplicativos.

User

Por que um objeto não pode ser usado para armazenar um dado de entrada do usuário lido pelo próprio objeto?

ChatGPT

Na verdade, um objeto pode ser usado para armazenar um dado de entrada do usuário lido pelo próprio objeto, desde que a classe do objeto seja definida para incluir um atributo para armazenar o valor. No entanto, geralmente é mais comum usar um objeto Scanner para ler o valor de entrada do usuário e armazená-lo em uma variável separada. Isso ocorre porque a classe Scanner é projetada especificamente para ler entradas do usuário e não para armazenar valores lidos.

Além disso, o objeto Scanner é projetado para ser reutilizável e pode ser usado para ler várias entradas do usuário sem precisar criar um novo objeto Scanner para cada entrada. Se um objeto fosse usado para armazenar um valor lido, ele teria que ser atualizado ou recriado sempre que uma nova entrada do usuário fosse lida, o que pode tornar o código menos eficiente e mais complicado de manter.

# Método Construtor

Ao declarar uma classe, você pode fornecer seu próprio construtor a fim de especificar a inicialização personalizada para objetos de sua classe (DEITEL, pg. 64).

Um construtor deve ter o mesmo nome que a classe (DEITEL, pg. 65).

O método construtor é um método especial que tem o mesmo nome da classe e é usado para inicializar objetos quando eles são criados. Ele é responsável por garantir que as variáveis da classe sejam inicializadas com valores apropriados quando um objeto é criado. Em outras palavras, o construtor é usado para instanciar um objeto com um estado inicial definido pelo programador (ChatGPT).

Mesmo na construção do código do método construtor, os parâmetros são variáveis locais que precisam ser atribuídas as variáveis de instância (AUTOR).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Pergunta ao GPT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Em Java, um método construtor obriga o objeto ser declarado com um parâmetro para a variável de instância?

Em Java, um método construtor não obriga necessariamente que um objeto seja declarado com um parâmetro para uma variável de instância. A existência ou não de parâmetros no construtor depende da implementação específica da classe.

Um construtor é um método especial que é invocado durante a criação de um objeto e é responsável por inicializar os valores das variáveis de instância daquela classe. Ele pode ter ou não parâmetros, dependendo das necessidades do projeto.

Se um construtor não tiver parâmetros, ele é chamado de construtor padrão. Ele é declarado sem nenhum parâmetro e pode ser usado para criar um objeto sem a necessidade de fornecer argumentos específicos durante a instanciação.

Aqui está um exemplo de uma classe com um construtor padrão:

java

Copy code

public class Exemplo {

private int numero;

// Construtor padrão

public Exemplo() {

numero = 0; // Inicialização padrão

}

// Outros métodos e atributos...

}

Nesse caso, o construtor padrão Exemplo() não possui parâmetros. Ele simplesmente inicializa o valor da variável de instância numero com o valor padrão de zero.

No entanto, é possível definir construtores com parâmetros para permitir que os valores das variáveis de instância sejam especificados durante a criação do objeto. Aqui está um exemplo de uma classe com um construtor com parâmetros:

java

Copy code

public class Exemplo {

private int numero;

// Construtor com parâmetros

public Exemplo(int valor) {

numero = valor;

}

// Outros métodos e atributos...

}

Nesse caso, o construtor Exemplo(int valor) possui um parâmetro valor que é usado para inicializar a variável de instância numero com um valor específico fornecido durante a instanciação do objeto.

Portanto, em resumo, a presença ou ausência de parâmetros em um construtor depende da implementação específica da classe e das necessidades do projeto. Um construtor pode ser definido com ou sem parâmetros, dependendo se você deseja que o objeto seja criado com valores específicos para as variáveis de instância ou com valores padrão.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Pergunta ao GPT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

No segundo exemplo, ao instanciar um objeto da classe exemplo é necessário passar um parâmetro?

No segundo exemplo, ao definir um construtor com parâmetros na classe Exemplo, ao instanciar um objeto da classe, é necessário passar um argumento correspondente ao parâmetro do construtor. Caso contrário, ocorrerá um erro de compilação.

Por exemplo, considerando o construtor com parâmetros Exemplo(int valor):

java

Copy code

public class Exemplo {

private int numero;

public Exemplo(int valor) {

numero = valor;

}

// Outros métodos e atributos...

}

Ao criar um objeto Exemplo, é necessário fornecer um valor inteiro para o construtor. Caso contrário, o código não será compilado.

Exemplo de uso correto:

java

Copy code

Exemplo obj = new Exemplo(10);

Nesse exemplo, estamos criando um objeto Exemplo e passando o valor 10 como argumento para o construtor. O construtor, por sua vez, inicializará a variável de instância numero com esse valor.

Exemplo de uso incorreto:

java

Copy code

Exemplo obj = new Exemplo(); // Erro de compilação!

Nesse exemplo incorreto, não estamos fornecendo o argumento necessário para o construtor com parâmetros. Portanto, ocorrerá um erro de compilação.

Em resumo, ao definir um construtor com parâmetros em uma classe, é necessário fornecer um argumento correspondente ao parâmetro ao instanciar um objeto da classe. Caso contrário, ocorrerá um erro de compilação.

# Tecnologia Swing – Classe JOptionPane

A classe JOptionPane é uma classe estática (ou seja, seus métodos podem ser chamados diretamente na classe, sem a necessidade de criar um objeto) e é usada para exibir diálogos de mensagem em um aplicativo Java. Portanto, não é necessário criar um objeto com new para utilizá-la (ChatGPT).

# “Estaticismo”

## Classe Estática

Em Java, uma classe estática é uma classe que não precisa ser instanciada para ser usada. Ao contrário de classes regulares, que exigem a criação de um objeto antes que seus métodos e campos possam ser acessados, as classes estáticas podem ser referenciadas diretamente pelo nome da classe (ChatGPT).

As classes estáticas podem ser usadas para criar funções utilitárias que não precisam acessar ou alterar o estado de um objeto. Por exemplo, a classe Math em Java é uma classe estática que fornece métodos para realizar operações matemáticas comuns, como calcular a raiz quadrada ou o seno de um número (ChatGPT).

## Variável Estática

Uma variável estática em Java é uma variável que pertence à classe em vez de pertencer a instâncias individuais (objetos) dessa classe. Ela é compartilhada por todas as instâncias da classe e pode ser acessada diretamente usando o nome da classe, sem a necessidade de criar um objeto da classe (ChatGPT).

As variáveis estáticas são declaradas usando o modificador static na sua declaração (ChatGPT).

## Método Estático

Em Java, métodos estáticos pertencem à classe em vez de uma instância específica da classe. Isso significa que eles podem ser chamados sem a necessidade de criar um objeto da classe, o que pode ser útil em algumas situações. No entanto, como eles pertencem à classe em si, e não a uma instância específica, não faz sentido chamá-los usando uma variável de referência para um objeto (ChatGPT).

Assim, é necessário usar o nome da classe ao chamar um método estático em Java, para indicar que o método está sendo chamado diretamente na classe em si, em vez de em uma instância específica. A sintaxe para chamar um método estático é a seguinte: NomeDaClasse.nomeDoMetodo() (ChatGPT).

[...] (DEITEL, pg. 164).

# Modificadores de Acesso *Public* e *Private*

Ser um método público significa que ele pode ser chamado de dentro de uma classe que não seja sua classe de “declaração” (de origem, em que foi declarado junto ao seu objeto padrão. Ele não é exclusivo dessa classe), porém isso não significa que ele pode ser chamado diretamente por uma variável ou por um objeto (instância de classe) de outra classe (classe diferente da que ele foi criado), até porque as instruções contidas dentro dele não seriam aplicáveis sendo que ele foi criado para manipular os atributos (ou variáveis de instâncias) do objeto padrão dele. Por isso um método geralmente é público e uma variável de instância, privada. E, por isso que sempre que ele for chamado, ele deve ser chamado por uma instância (ou pela própria classe no caso de static) da mesma classe em que ele foi declarado.

Uma variável de instância pública seria acessível no código de outra classe, porém sempre através do objeto da mesma classe que a variável, ou seja, um objeto deve ser instanciado.

COMPILA COM javac [ ].java

EXECUTA COM java [ ]

Placeholders x Escapes x Especificadores de formato

# FOR APRIMORADO

**Pergunta:** O for aprimorado em Java "percorre" os valores do array e não o indexador ([0], [1]) de cada elemento?

**ChatGPT**

Sim, você está correto. O loop for aprimorado em Java (também conhecido como "enhanced for" ou "for-each") é projetado para percorrer os valores de um array ou de uma coleção, em vez de iterar pelos índices.

A sintaxe do loop for aprimorado em Java é a seguinte:

for (tipo valor : array) {

// bloco de código a ser executado

}

Nesse tipo de loop, a variável valor recebe cada elemento do array ou coleção um por um, e o bloco de código dentro do loop é executado para cada valor.

Aqui está um exemplo de uso do loop for aprimorado para percorrer os valores de um array:

int[] numeros = {1, 2, 3, 4, 5};

for (int numero : numeros) {

System.out.println(numero);

}

Nesse exemplo, cada elemento do array numeros é atribuído à variável numero em cada iteração do loop. Então, dentro do bloco de código do loop, imprimimos o valor de numero.

A saída será:

1

2

3

4

5

Observe que, ao usar o loop for aprimorado, você não tem acesso direto aos índices dos elementos. O loop itera automaticamente pelos valores do array ou da coleção, tornando o código mais conciso e legível quando você só precisa dos valores e não dos índices.

# Argumentos de Linha de Comando

[...] (DEITEL, pg. 221).

É possível usar os argumentos de linha de comando sem o método main?

ChatGPT

Não, os argumentos de linha de comando só podem ser acessados através do método main em Java. O método main é o ponto de entrada do programa e é onde os argumentos de linha de comando são recebidos como um parâmetro do tipo String[] args.

Os argumentos de linha de comando são passados para o programa no momento da execução, quando o comando java é chamado juntamente com o nome da classe a ser executada e os argumentos, se houver. O método main é responsável por receber e processar esses argumentos.

Se você deseja usar os argumentos de linha de comando em outras partes do programa além do método main, você pode passá-los como parâmetros para outros métodos ou classes. O método main pode receber os argumentos de linha de comando e, em seguida, chamar outros métodos, passando os argumentos como necessário. Mas a captura e o processamento inicial dos argumentos devem ocorrer no método main.

Portanto, para acessar e utilizar os argumentos de linha de comando em um programa Java, é necessário fazer uso do método main.

# Dúvidas

1. **Diferença entre variável, objeto e atributos?**

**Resposta:** Variável é um elemento que armazena um dado e não, necessariamente, tem vínculo com um objeto. Objeto é a instanciação de uma classe. Atributos são “variáveis” que guardam valores dos objetos. Métodos realizam ações e manipulam os valores dos atributos (AUTOR).

1. **Por que é necessário criar um objeto da classe scanner para “habilitar” a leitura de dados, sendo que ele não vai receber nenhum atributo e nem armazenar nenhum dado?**

**Resposta:** A função de ler um dado de entrada é executada por uma classe por se tratar de uma linguagem orientada a objeto, ou seja, não existe um comando ou operador isolado que executa essa função, ela foi implementada na linguagem como uma classe (AUTOR).

1. **Os métodos da classe Scanner são públicos, então podem ser diretamente atribuídos (chamados) em outras variáveis?**

**Resposta:** Não, o padrão é ler o dado de entrada através de uma instância criada da classe Scanner e depois transferir para uma variável ou outro objeto.

1. **É criado um objeto da classe Scanner, através do construtor new, apenas para habilitar os métodos da classe (~~que são público, podem ser aplicados em outras variáveis ou classes~~), porque como Java é uma linguagem orientada a objeto não existe um comando de leitura de dados “isolado” como uma linguagem imperativa, portanto esses métodos (“comandos”) tem que ser criados dentro de uma classe e para habilitá-los é necessário instanciar (chamar o construtor) a classe. A operação de leitura é camuflada em uma classe por Java ser uma LOO.**

**Resposta:**

1. **Por que o objeto System.out não foi instanciado como os outros (Ex.: Scanner)?**

**Resposta:** O objeto System.out não é instanciado com o operador new, como ocorre com o objeto da classe Scanner, porque ele não é uma instância de uma classe (ChatGPT).

Na verdade, System.out é um objeto predefinido (ou pré-instanciado) da classe PrintStream, que representa a saída padrão do sistema. Em outras palavras, ele é um objeto que já existe na memória e que pode ser utilizado diretamente pelo programa (ChatGPT).

Dessa forma, ao utilizar o objeto System.out, não é necessário criar uma nova instância da classe PrintStream com o operador new, como ocorre com outras classes que precisam ser instanciadas para serem utilizadas. O objeto System.out já está disponível para uso em qualquer lugar do programa (ChatGPT).

1. **O que são objetos predefinidos no código em Java?**

**Resposta:** Em Java, objetos predefinidos são objetos que são criados automaticamente pelo sistema e estão disponíveis para uso em todos os programas Java, sem a necessidade de criar instâncias desses objetos explicitamente. Esses objetos são fornecidos como parte das classes padrão da linguagem Java e são frequentemente referidos como objetos padrão ou objetos embutidos (ChatGPT).

1. **Por que um objeto não pode ser usado para armazenar um dado de entrada do usuário lido pelo próprio objeto?**

**Resposta:** Na verdade, um objeto pode ser usado para armazenar um dado de entrada do usuário lido pelo próprio objeto, desde que a classe do objeto seja definida para incluir um atributo para armazenar o valor. No entanto, geralmente é mais comum usar um objeto Scanner para ler o valor de entrada do usuário e armazená-lo em uma variável separada. Isso ocorre porque a classe Scanner é projetada especificamente para ler entradas do usuário e não para armazenar valores lidos (ChatGPT).

Além disso, o objeto Scanner é projetado para ser reutilizável e pode ser usado para ler várias entradas do usuário sem precisar criar um novo objeto Scanner para cada entrada. Se um objeto fosse usado para armazenar um valor lido, ele teria que ser atualizado ou recriado sempre que uma nova entrada do usuário fosse lida, o que pode tornar o código menos eficiente e mais complicado de manter (ChatGPT).

1. **O que significa dizer que um método é público? Não é possível chamar esse método em objetos de outras classes?**

**Resposta:**

1. **Em Java, objeto é um conceito (abstração) de um conjunto de variáveis (atributos) referenciadas por uma outra única variável primitiva que contém apenas o endereço de memória?**

**Resposta:** Em Java, um objeto é uma instância de uma classe que encapsula um conjunto de atributos e métodos relacionados. Um objeto é criado alocando-se espaço na memória e inicializando-o com valores padrão ou personalizados por meio de um construtor (ChatGPT).

Em relação à variável que faz referência ao objeto, ela armazena o endereço de memória do objeto, que é um valor de tipo por referência. Essa variável não contém diretamente os valores dos atributos do objeto, mas sim uma referência a esse conjunto de valores (ChatGPT).

Assim, é possível acessar os atributos e métodos de um objeto por meio da variável de referência, que é uma forma de interagir com o objeto na memória. Quando duas variáveis de referência apontam para o mesmo objeto, elas compartilham o mesmo endereço de memória e, portanto, referenciam o mesmo objeto (ChatGPT).

1. **Como seria a chamada de uma variável de instância pública, chamada name, no código de uma classe diferente da qual a variável name foi declarada?**

**Resposta:** Para acessar uma variável de instância pública chamada name de uma classe diferente da qual ela foi declarada, é necessário criar um objeto da classe que possui a variável e então acessá-la através desse objeto.

Suponha que a classe que declara a variável name seja a classe Person. Se quisermos acessar a variável name em outra classe, podemos fazer o seguinte:

Person p = new Person();

String nome = p.name;

Nesse exemplo, criamos um objeto p da classe Person e usamos esse objeto para acessar a variável name da classe Person. Note que como name é uma variável pública, podemos acessá-la diretamente usando o operador de ponto (.).

1. **Por que o tipo do dado de retorno de um método é especificado na linha de declaração do método e não como um parâmetro ou na linha da instrução de retorno?**

**Resposta:** O tipo de dado de retorno de um método é especificado na linha de declaração do método porque é uma parte essencial da assinatura do método, que é usada pelo compilador para verificar a corretude do código que está chamando o método. A especificação do tipo de retorno permite que o compilador verifique se o valor retornado pelo método é compatível com o tipo esperado pelo código que está chamando o método (ChatGPT).

Além disso, a especificação do tipo de retorno na linha de declaração do método ajuda a documentar o código e a torná-lo mais legível para outros programadores que podem estar trabalhando com o código. Ao ver a assinatura do método, é possível determinar rapidamente qual é o tipo de dado que o método retorna, o que pode ser útil para entender a funcionalidade do método e para evitar erros de programação (ChatGPT).

1. **Qual a diferença entre passagem por referência e passagem por valor?**

# Comandos Básicos

*System.out.print*

//exibe ou imprime uma linha de texto na janela de comando (DEITEL, pg. 31).

# Links de Interesse

Java Platform Standard Edition 8 Documentation

<https://docs.oracle.com/javase/8/docs/>

Java™ Platform, Standard Edition 8 API Specification

<https://docs.oracle.com/javase/8/docs/api/>

The Java™ Tutorials

<https://docs.oracle.com/javase/tutorial/>

Getting Started with Java

<https://dev.java/learn/getting-started/>

Java Tutorial

<https://www.scaler.com/topics/java/>

Processo de Execução de Programa Java

<https://simplesnippets.tech/execution-process-of-java-program-in-detail-working-of-just-it-time-compiler-jit-in-detail/>

Java Language Basics

<https://dev.java/>

<https://dev.java/learn/language-basics/>

Oracle Java Treinamento e Certificação

<https://education.oracle.com/software/java/pFamily_48>

O que é Java? (AWS - Amazon)

<https://aws.amazon.com/pt/what-is/java/>

Problemas Visual Studio Code

<https://www.clubedohardware.com.br/forums/topic/1602751-o-que-esse-x-significa/>

# JAVA - Resumo Tabular

Operadores Matemáticos

Operadores Lógicos

Operadores de Igualdade

Operadores Relacionais

Operadores de Atribuição Compostos

Operadores de Incremento e Decremento

Tipos de Variáveis

Especificadores de Formato

Caracteres de Escape

# Principais Tópicos para Aprender o Básico de uma Linguagem de Programação

1. Entender a sintaxe, estrutura geral da linguagem e em qual paradigma da programação ela se encaixa melhor.
2. Tipos de variáveis e como declarar.
3. Operadores.
4. Estruturas de condição e de repetição.
5. “Interação” com o usuário (input e print).
6. Vetores de dados.
7. Manipulação de strings.
8. Tratativa de erros.