APOSTILA 2

Tipos de dados

Créditos:

Autor: Vitor Dutra.
Revisão Técnica: Prof. Jeferson Leon e Prof. Éder Oliveira de Rosso.
Curso de Java.
VC Ensinos.

Curso de Java | Logica | VC Ensinos

Autor: Vitor Fernando Dutra
Revisão Técnica: Prof. Jeferson Faleiro Leon e Prof. Éder Oliveira de Rosso

Sumário:

Autor: Vitor Fernando Dutra
Revisão Técnica: Prof. Jeferson Faleiro Leon e Prof. Éder Oliveira de Rosso

Tipos de Dados em Java

1. Apresentação dos Tipos de Dados

Os tipos de dados em Java são fundamentais para armazenar informações e determinar como os dados serão manipulados durante a execução do programa. Eles são categorizados em dois grupos principais:

- **Tipos Primitivos:** Usados para representar valores simples, como números inteiros, decimais, caracteres e valores booleanos.
- **Tipos Não Primitivos:** Representam estruturas mais complexas, como strings, arrays, classes e enums.

A escolha adequada do tipo de dado é essencial para garantir eficiência e organização no código.

---

2. Configurações Iniciais

Para iniciar o desenvolvimento em Java, é necessário preparar o ambiente. O uso do NetBeans IDE simplifica esse processo.

Instalação do JDK

O **Java Development Kit (JDK)** é um conjunto de ferramentas que inclui o compilador, bibliotecas e a máquina virtual (JVM) necessários para o desenvolvimento.

1. Baixe o JDK no site oficial da Oracle, OpenJDK ou distribuições como Amazon Corretto.
2. Após a instalação, configure a variável de ambiente `JAVA_HOME` para que o sistema reconheça o JDK.

Instalação do NetBeans IDE

1. Faça o download do **NetBeans IDE** no site https://netbeans.apache.org/.
2. Durante a instalação, associe o IDE ao JDK previamente instalado.

Configurando o Primeiro Projeto

Autor: Vitor Fernando Dutra
Revisão Técnica: Prof. Jeferson Faleiro Leon e Prof. Éder Oliveira de Rosso

1. Abra o NetBeans e clique em `File > New Project`.
2. Selecione `Java with Ant > Java Application`.
3. Escolha um nome para o projeto, defina o local onde ele será salvo e clique em **Finish**.

Após esses passos, o ambiente estará pronto para o desenvolvimento de aplicativos Java.

---

3. Criando um Novo Documento Java

Em Java, cada classe é definida em um arquivo separado, com extensão `.java`. Para criar um novo arquivo no NetBeans:

1. No painel lateral, clique com o botão direito sobre o pacote principal do projeto.
2. Escolha `New > Java Class`.
3. Dê um nome à classe e clique em **Finish**.

Estrutura Básica de um Programa Java

Um arquivo Java contém, no mínimo, uma classe e um método principal (`main`).

```
public class MinhaClasse {
    public static void main(String[] args) {
        System.out.println("Bem-vindo ao Java!");
    }
}
```

Neste exemplo:

- A palavra-chave `class` define a classe pública `MinhaClasse`.
- O método `main` é o ponto de entrada do programa, onde a execução começa.
- O comando `System.out.println` exibe uma mensagem no console.

---

4. Tipos de Dados Primitivos e Não Primitivos

Tipos Primitivos

Os tipos primitivos são os mais simples em Java e utilizados para armazenar valores básicos. Existem oito tipos:

Autor: Vitor Fernando Dutra
Revisão Técnica: Prof. Jeferson Faleiro Leon e Prof. Éder Oliveira de Rosso

- O tipo **byte** é usado para armazenar números inteiros pequenos, com valores que variam de -128 a 127.
- O tipo **short** permite armazenar números inteiros um pouco maiores, com valores de -32.768 a 32.767.
- O tipo **int** é o mais comum para números inteiros e suporta valores de -2^{31} a $2^{31}-1$.
- O tipo **long** é usado para números inteiros muito grandes, variando de -2^{63} a $2^{63}-1$.
- O tipo **float** é utilizado para números decimais de precisão simples, com até 7 dígitos de precisão.
- O tipo **double** armazena números decimais com maior precisão, suportando até 15 dígitos.
- O tipo **char** é usado para armazenar um único caractere, como 'A' ou '1', seguindo o padrão Unicode.
- O tipo **boolean** representa valores lógicos, sendo `true` (verdadeiro) ou `false` (falso).

Exemplo de Uso:

```
public class TiposPrimitivos {
    public static void main(String[] args) {
        int idade = 30;
        double altura = 1.85;
        boolean ativo = true;

        System.out.println("Idade: " + idade);
        System.out.println("Altura: " + altura);
        System.out.println("Ativo: " + ativo);
    }
}
```

Tipos Não Primitivos

Os tipos não primitivos são mais complexos e permitem representar coleções de dados ou objetos.

- **Strings** são usadas para representar texto, como `"Olá, mundo!"`. Apesar de serem tratadas como tipos básicos, elas são objetos de uma classe.
- **Arrays** são estruturas que armazenam múltiplos valores do mesmo tipo, como números ou strings.
- **Classes Personalizadas** permitem organizar atributos e métodos em um único local, sendo amplamente usadas em projetos maiores.
- **Enums** são conjuntos fixos de constantes, como os dias da semana ou estados de um processo.

Exemplo de Uso:

Autor: Vitor Fernando Dutra
Revisão Técnica: Prof. Jeferson Faleiro Leon e Prof. Éder Oliveira de Rosso

```
public class TiposNaoPrimitivos {
    public static void main(String[] args) {
        String nome = "Programação Java";
        int[] numeros = {10, 20, 30, 40};

        System.out.println("Nome: " + nome);
        for (int num : numeros) {
            System.out.print(num + " ");
        }
    }
}
```

---

5. Conversões de Tipos de Dados

As conversões de tipos ajustam o formato de uma variável para outro.

Conversão Implícita (Widening)

Ocorre automaticamente quando o tipo de destino suporta valores maiores. Por exemplo, ao converter de `int` para `double`, não há risco de perda de dados:

```
int inteiro = 10;
double decimal = inteiro;
System.out.println(decimal);  // Saída: 10.0
```

Conversão Explícita (Casting)

É necessária quando há risco de perda de dados, como ao converter de `double` para `int`. Deve ser feita manualmente:

```
double valor = 10.75;
int truncado = (int) valor;
System.out.println(truncado);  // Saída: 10
```

Conversão entre Primitivos e Não Primitivos

O Java oferece as **wrapper classes** para converter tipos primitivos em objetos e vice-versa. Esse processo é conhecido como **autoboxing** e **unboxing**.

```
int numero = 100;
```

Autor: Vitor Fernando Dutra
Revisão Técnica: Prof. Jeferson Faleiro Leon e Prof. Éder Oliveira de Rosso

```
Integer objeto = numero;  // Autoboxing
int novoNumero = objeto;  // Unboxing
```

---

6. Referências Biográficas

- Oracle. *Java Platform, Standard Edition Documentation.* Disponível em: https://docs.oracle.com/javase/.
- Apache NetBeans. *Official NetBeans IDE Documentation.* Disponível em: https://netbeans.apache.org/.
- Deitel, Paul; Deitel, Harvey. *Java: How to Program*. 11ª edição. Pearson, 2017.