1.3. Estrutura Básica de Programação em Java SE

1.3.1. Instruções de Entrada e Saída em Janelas

No modo console (texto puro), você já conhece o comando:

System.out.println();

Ele serve para exibir informações na tela, seguidas de uma quebra de linha.

Agora vamos introduzir um pouco de interface gráfica (GUI) usando a classe `JOptionPane`, que faz parte do framework Swing — o padrão atual para criação de janelas no Java.

A classe `JOptionPane` fornece métodos estáticos para exibir caixas de diálogo, ou seja, janelinhas com mensagens e campos para entrada de dados.

📤 Exibindo mensagens: `showMessageDialog()`

Esse método mostra uma janela com um texto e um botão "OK". Ele é usado para exibir informações ou avisos. Exemplo:

JOptionPane.showMessageDialog(null, "Olá, bem-vindo ao programa!");

✏️ Solicitando informações: `showInputDialog()`

Esse método exibe uma janela com um campo de texto para o usuário digitar uma informação. O valor digitado é retornado como uma `String`. Exemplo:

String nome = JOptionPane.showInputDialog(null, "Qual é o seu nome?");

Depois que o usuário digita o nome e clica em "OK", o texto digitado é armazenado na variável `nome`.

Você pode então exibir outra mensagem, usando esse valor:

JOptionPane.showMessageDialog(null, "Prazer em conhecê-lo(a), " + nome);

1.3.2. Tipos Primitivos, Strings e Conversão de Valores

✅ Tipos primitivos em Java:

Tipo	Descrição	Exemplo
`int`	Números inteiros	`int x = 10;`
`double`	Números com casas decimais	`double y = 5.7;`
`char`	Um único caractere	`char letra = 'A';`
`boolean`	Verdadeiro ou falso	`boolean ativo = true;`

- Números sem ponto são considerados `int`.
- Números com ponto são considerados `double`.

Exemplos:

produto.adicionarQuantidadeAoEstoque(5); // int
aluno.gravarMedia(9.5); // double

📌 Strings

Em Java, `String` não é um tipo primitivo — é uma classe, o que significa que suas variáveis são objetos.

Exemplo:

String nome = "Patrick";

Strings permitem concatenação com o operador `+`:

```
String nome = "João";
String sobrenome = "Silva";
JOptionPane.showMessageDialog(null, "Meu nome completo é " + nome + " " + sobrenome);

int idade = 22;
JOptionPane.showMessageDialog(null, "Tenho " + idade + " anos de idade.");
```

> 📌 **Observação: quando você concatena um número com uma `String`, a conversão para texto é feita automaticamente.**

🔁 Convertendo `String` para número

O método `showInputDialog()` sempre retorna uma `String`, mesmo se o usuário digitar um número. Para usá-lo em cálculos, você precisa converter essa String para um número.

✅ De `String` para `int`:

Use `Integer.parseInt()`:

```
String texto = JOptionPane.showInputDialog("Digite sua idade:");
int idade = Integer.parseInt(texto);
```

✅ De `String` para `double`:

Use `Double.parseDouble()`:

```
String texto = JOptionPane.showInputDialog("Digite sua altura:");
double altura = Double.parseDouble(texto);
```

✅ Resumo:

De	Para	Como converter
`String`	`int`	`Integer.parseInt()`
`String`	`double`	`Double.parseDouble()`
`int/double`	`String`	Automaticamente na concatenação

1.3.3. Conceitos Básicos de Memória, Comparação de Strings e Strings Mutáveis

Em Java, tanto objetos quanto variáveis ocupam espaços específicos na memória. Os nomes que damos a eles são apenas referências para esses espaços. Esse conceito é especialmente importante ao lidar com strings.

Comparação de Strings

Em vez de usar o operador `==` para comparar strings, devemos utilizar o método `.equals()`. Isso porque o `==` compara se duas variáveis apontam para o mesmo local de memória, enquanto o `.equals()` compara o conteúdo das strings.

java

CopiarEditar

```
String nome1 = "João";

String nome2 = "João";


System.out.println(nome1 == nome2);        // true (pode ser true, mas não é garantido)

System.out.println(nome1.equals(nome2));   // true (compara o conteúdo corretamente)
```

💡 **Dica: sempre use `.equals()` quando quiser comparar textos em Java.**

Imutabilidade das Strings

Em Java, objetos da classe `String` são imutáveis. Isso significa que, uma vez criado, o conteúdo da string não pode ser alterado. O que parece ser uma alteração, na verdade, é a criação de um novo objeto na memória.

Exemplo:

java

CopiarEditar

```
String resultado = "";
```

```
resultado = resultado + "Nome do candidato 1\n";

resultado = resultado + "Nome do candidato 2\n";
```

Nesse caso, a cada concatenação com `+`, uma nova string é criada na memória, e a variável `resultado` passa a apontar para essa nova string. Isso pode causar problemas de desempenho se for feito muitas vezes, como dentro de um laço de repetição.

Usando StringBuilder para Melhor Desempenho

Para evitar esse problema, utilizamos a classe `StringBuilder`, que permite a modificação do conteúdo da string de forma eficiente, sem criar novos objetos a cada operação.

Exemplo com `StringBuilder`:

java

CopiarEditar

```
StringBuilder resultado = new StringBuilder();

resultado.append("Nome do candidato 1\n");

resultado.append("Nome do candidato 2\n");


String stringFinal = resultado.toString(); // converte para String
"normal"
```

Esse é o jeito mais indicado e performático para construir strings dinamicamente, especialmente em estruturas de repetição.

✅ Resumo

- **Use `.equals()` para comparar o conteúdo de strings.**
- **Strings são imutáveis em Java: cada modificação cria um novo objeto.**

- **Para desempenho, use `StringBuilder` ao concatenar strings várias vezes.**

1.3.4. Uso do Operador Ternário

O operador ternário `? :` é uma forma simples e elegante de tomar decisões em Java. Ele pode ser usado no lugar de estruturas `if...else`, deixando o código mais curto e fácil de ler, especialmente em situações simples.

Sintaxe
resultado = condição ? valorSeVerdadeiro : valorSeFalso;

- **Se a condição for verdadeira, `resultado` recebe o `valorSeVerdadeiro`.**
- **Se for falsa, `resultado` recebe o `valorSeFalso`.**

Exemplo prático

```
int nota = 7;
String resultadoDoAluno = (nota >= 6) ? "Aprovado" : "Reprovado";
System.out.println("Resultado: " + resultadoDoAluno);
```

Saída:

Resultado: Aprovado

Comparando com if...else

O código acima com `if...else` seria assim:

```
int nota = 7;
String resultadoDoAluno;
if (nota >= 6) {
   resultadoDoAluno = "Aprovado";
} else {
   resultadoDoAluno = "Reprovado";
}
```

Veja como o operador ternário torna o código mais enxuto.

✅ Quando usar?

- **Quando você quiser atribuir um valor com base em uma condição simples.**
- **Quando a legibilidade não for prejudicada.**
- **Evite usar quando houver lógica mais complexa — nesse caso, `if...else` pode ser mais claro.**

Se quiser, posso te mostrar outros usos interessantes do operador ternário, como para decisões com números, booleanos ou até com `JOptionPane`. Quer ver algum desses?