
FIAP

Domain Driven Design
using Java

AGENDA

1. Entrada e Saída de Dados
2. Exercícios

Entrada e Saída de Dados

-> Introdução

- A entrada e saída de dados são aspectos fundamentais de qualquer linguagem de programação, incluindo Java.
- Este capítulo abordará como lidar com entrada de dados do usuário e saída de informações em diferentes contextos, como o console e arquivos.

Entrada de Dados com Scanner

- A entrada de dados em Java é geralmente realizada através da classe Scanner do pacote *java.util*.
- Fornece métodos para ler diferentes tipos de dados a partir do teclado (entrada padrão) ou de outras fontes de entrada.

```java
import java.util.Scanner;

public class EntradaDeDados {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Digite seu nome: ");
        String nome = scanner.nextLine();
        System.out.print("Digite sua idade: ");
        int idade = scanner.nextInt();
        System.out.println("Olá, " + nome + "! Você tem " + idade + " anos.");
        scanner.close();
    }
}
```

Entrada de Dados com Scanner

- Considerações adicionais:
    - **Encerrando o Scanner:** Sempre é importante encerrar o objeto Scanner após sua utilização chamando o método *close()*.
    - **Tratamento de Outros Tipos de Dados:** Além de *nextLine()* e *nextInt()*, a classe Scanner fornece métodos para ler outros tipos de dados como *nextDouble()*, *nextBoolean()*, entre outros.
    - **Validação de Entrada:** Valide os dados de entrada do usuário para garantir que estejam no formato esperado antes de utilizá-los.

Saída de Dados com *System.out.println()*

- A saída de dados em Java refere-se ao processo de exibir informações para o usuário ou para outros sistemas externos.
- O método *println()* da classe *System.out* é amplamente utilizado para exibir informações no console.

```java
public class SaidaDeDados {
    public static void main(String[] args) {
        String nome = "Alice";
        int idade = 25;
        System.out.println("Nome: " + nome);
        System.out.println("Idade: " + idade);
    }
}
```

Formatação de Saída

- Além de exibir informações, é possível formatar a saída de dados para torná-la mais legível e apresentável.
- O método *printf()* pode ser utilizado para formatar a saída.

```
public class FormatacaoDeSaida {
    public static void main(String[] args) {
        String nome = "João";
        int idade = 30;
        System.out.printf("Nome: %s, Idade: %d%n", nome, idade);
    }
}
```

Saída de Dados em Arquivos

- Além de exibir informações no console, é possível direcionar a saída de dados para arquivos utilizando as classes *FileWriter* e *PrintWriter*.

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class SaidaEmArquivo {
    public static void main(String[] args) {
        try (PrintWriter arquivo = new PrintWriter(new FileWriter("saida.tx
            arquivo.println("Este é um exemplo de saída em arquivo.");
        } catch (IOException e) {
            System.out.println("Erro ao escrever no arquivo: " + e.getMessa
        }
    }
}
```

Manipulação de Strings

- A manipulação de strings é uma parte fundamental da programação em Java.
- Concatenação de Strings: Utiliza o operador **+** para combinar duas ou mais strings em uma única string.

```
String nome = "João";
int idade = 30;
String mensagem = "Olá, meu nome é " + nome + " e tenho " + idade + " anos.
System.out.println(mensagem);
```

Manipulação de Strings

- Java fornece uma variedade de métodos para manipular strings, permitindo operações como extrair substrings, converter maiúsculas para minúsculas, verificar o comprimento da string, e muito mais.

```
String texto = "Exemplo de manipulação de strings em Java";
int comprimento = texto.length();
String maiusculas = texto.toUpperCase();
String minusculas = texto.toLowerCase();
String substring = texto.substring(8, 18);
boolean contem = texto.contains("manipulação");
String substituida = texto.replace("Java", "JavaScript");
```

Formatação de Strings

- Além da concatenação simples, Java oferece recursos avançados de formatação de strings usando a classe *String.format()* e a classe *Formatter*.

```
String nome = "Maria";
int idade = 25;
String mensagemFormatada = String.format("Olá, meu nome é %s e tenho %d anos.", nome, idade);
System.out.println(mensagemFormatada);
```

Formatação de Strings

- Exemplo com *Formatter*:

```java
import java.util.Formatter;

public class FormatacaoComFormatter {
    public static void main(String[] args) {
        String nome = "Maria";
        int idade = 25;
        Formatter formatter = new Formatter();
        formatter.format("Olá, meu nome é %s e tenho %d anos.", nome, idade
        String mensagem = formatter.toString();
        System.out.println(mensagem);
        formatter.close();
    }
}
```

Comparação de Strings

- Ao comparar strings em Java, utilize os métodos *equals()* ou *equalsIgnoreCase()* em vez do operador **==,** que compara referências de objeto, não os conteúdos das strings.

```java
String str1 = "hello";
String str2 = "Hello";
if (str1.equals(str2)) {
    System.out.println("As strings são iguais.");
} else {
    System.out.println("As strings são diferentes.");
}
```

Conversões entre Tipos e Casting

- Em Java, as conversões entre tipos podem ser classificadas em conversões implícitas e explícitas:
  - **Conversões Implícitas:** Ocorrem automaticamente pelo compilador quando não há perda de dados.
  - **Conversões Explícitas (Casting):** Realizadas manualmente pelo programador e podem resultar em perda de dados.
- Exemplo de conversão implícita:

```
int numero = 10;
double valor = numero; // Conversão implícita de int para double
```

Conversões entre Tipos e Casting

- O casting é usado para converter explicitamente um tipo de dado em outro.

- Exemplo de casting

```
double valorDouble = 10.5;
int valorInteiro = (int) valorDouble; // Casting de double para int
System.out.println("Valor Inteiro: " + valorInteiro); // Saída: 10
```

Cuidados ao Utilizar Casting

- Ao utilizar casting em Java, esteja ciente das possíveis perdas de dados e garanta que a conversão seja realizada de forma segura.
- Exemplo

```java
double valorDouble = 10.5;
if (valorDouble % 1 == 0) {
    int valorInteiro = (int) valorDouble; // Casting de double para int
    System.out.println("Valor Inteiro: " + valorInteiro);
} else {
    System.out.println("O valor double não pode ser convertido para inteiro sem perda de dados.");
}
```

Exercícios

1. Implemente um programa que solicite ao usuário seu nome, idade e altura, e depois exiba essas informações formatadas no console. Utilize a classe Scanner para ler as entradas do usuário.

2. Implemente um programa que solicite ao usuário seu nome, idade e salário, e depois exiba essas informações formatadas no console utilizando o método *printf()*. Utilize especificadores de formato para garantir que a saída seja legível e apresentável.

3. Crie um programa que solicite ao usuário uma frase e realize as seguintes operações:

    - Converta a frase para maiúsculas.
    - Converta a frase para minúsculas.
    - Exiba o comprimento da frase.
    - Verifique se a frase contém a palavra "Java".
    - Substitua a palavra "Java" por "Python" na frase, se presente.

Exercícios

4. Crie um programa que solicite ao usuário um número decimal e o converta para um número inteiro utilizando casting.

5. Implemente um programa que solicite ao usuário duas strings e compare seus conteúdos utilizando os métodos equals() e equalsIgnoreCase().


FIAP