**Entrada e saída de dados**

**Introdução ao Conceito de Entrada e Saída de Dados em Java**

No desenvolvimento em Java, a entrada e saída de dados permitem que um programa interaja com o usuário e manipule informações. Compreender como capturar dados fornecidos pelo usuário e como apresentar informações no console ou em arquivos é essencial para criar aplicações funcionais e interativas. Aqui, exploraremos os conceitos básicos de entrada e saída em Java, utilizando classes e métodos que facilitam essas operações.

**Fundamentos de Entrada e Saída de Dados**

Os fundamentos de entrada e saída de dados em Java são cruciais para a comunicação entre um programa e o usuário, bem como para a manipulação de dados em arquivos e outros dispositivos. Este processo envolve a captura de informações fornecidas pelo usuário e a exibição de resultados ou a gravação de dados em diversos formatos.

**Entrada de Dados**

A entrada de dados refere-se à obtenção de informações do usuário ou de fontes externas e pode ser realizada de várias maneiras. Em Java, as abordagens mais comuns para entrada de dados são usando a classe Scanner, usando *BufferedReader e InputStreamReader*.

A classe Scanner fornece uma maneira prática e flexível de ler dados do console. Ela suporta diferentes tipos de dados, como inteiros, strings e números de ponto flutuante.

Com métodos como *nextLine(), nextInt(),* e *nextDouble()*, a *Scanner* facilita a leitura e a conversão de entradas para tipos específicos. Exemplo:

*Scanner scanner = new Scanner(*[*System.in*](http://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();*

O *BufferedReader*, em combinação com o *InputStreamReader*, é utilizado para ler dados de entrada de maneira eficiente, especialmente para leitura de texto de arquivos ou do console. *BufferedReader* fornece métodos como *readLine()* para ler linhas completas, enquanto *InputStreamReader* converte bytes em caracteres. Exemplo:

*BufferedReader reader = new BufferedReader(new InputStreamReader(*[*System.in*](http://system.in/)*));  
System.out.print("Digite seu nome: ");  
String nome = reader.readLine();  
System.out.print("Digite sua idade: ");  
int idade = Integer.parseInt(reader.readLine());*

**Saída de Dados**

A saída de dados refere-se ao processo de exibição de informações para o usuário ou de gravação de dados em um arquivo. Em Java, as principais ferramentas para saída de dados são usando *System.out ou usando PrintWriter.*

*System.out*é um fluxo de saída padrão que permite imprimir dados no console. É uma das formas mais simples e diretas de exibir informações, utilizando métodos como *print()* e *println()* para formatar e apresentar dados. Exemplo:

*System.out.println(“Hello, World!”); // imprime uma linha de texto*

A classe *PrintWriter*é utilizada para realizar operações de escrita de forma mais avançada. Ela oferece métodos adicionais para formatar a saída, como *printf()* e *format()*, que permitem um controle mais detalhado sobre a apresentação dos dados. Exemplo:

*PrintWriter writer = new PrintWriter(System.out, true);  
writer.println(“Hello, World!”); // imprime uma linha de texto  
writer.printf(“Nome: %s, Idade: %d%n”, “João”, 30); // imprime com formatação*

Esses fundamentos estabelecem a base para a interação com o usuário e para a manipulação de dados em Java, sendo essenciais para o desenvolvimento de aplicações que requerem entrada e saída de dados eficazes.

**Manipulação de Arquivos**

Manipular arquivos é uma parte fundamental do desenvolvimento de software, permitindo que você armazene, leia e modifique dados de forma persistente.

Em Java, a manipulação de arquivos é facilitada por um conjunto robusto de classes que oferece diversas ferramentas para trabalhar com arquivos de texto e binários.

Com essas ferramentas, você pode criar, ler, escrever e gerenciar arquivos de maneira eficiente, tornando possível o armazenamento e a recuperação de informações entre execuções do programa.

Para ler e escrever arquivos, Java utiliza classes do pacote ‘[java.io](http://java.io/)’ como *FileReader, FileWriter, BufferedReader* e *BufferedWriter.*

Exemplo para escrever arquivo com *BufferedWriter*:

*public class FileWriteExample {  
      public static void main(String[] args) {  
          try (BufferedWriter writer = new BufferedWriter(new FileWriter(“arquivo.txt”))) {  
              writer.write(“Olá, Mundo!”);  
              writer.newLine();  
              writer.write(“Segunda linha.”);  
          } catch (IOException e) {  
              e.printStackTrace();  
          }  
      }  
  }*

Exemplo para ler um arquivo com BufferedReader:

*public class FileReadExample {  
      public static void main(String[] args) {  
          try (BufferedReader reader = new BufferedReader(  
new FileReader(“arquivo.txt”))) {  
              String linha;  
              while ((linha = reader.readLine()) != null) {  
                  System.out.println(linha);  
              }  
          } catch (IOException e) {  
              e.printStackTrace();  
          }  
      }  
  }*

Este resumo abrange as principais técnicas e classes usadas para entrada e saída de dados em Java, que são fundamentais para criar programas que interagem com o usuário e manipulam arquivos de forma eficiente.

**Conteúdo Bônus**

**Título**: Java para iniciantes: Entrada e Saída de Dados em Java

**Canal**: Cryswerton Silva

**Plataforma**: YouTube

**Descrição**: Neste vídeo, eu mostro como usar entrada e saída de dados utilizando a linguagem de programação Java. Se você é iniciante em programação Java, uma das primeiras coisas que você precisa aprender é como usar os recursos de entrada e saída de dados em Java.

**Referências Bibliográficas**

ASCENCIA, A. F. G.; CAMPOS, E. A. V. de. **Fundamentos da programação**: algoritmos, Pascal, C/C++ e Java. 2. ed. Pearson, 2007.

DEITEL, H. M.; DEITEL, P. J.; CHOFFNES, D. R. **Sistemas operacionais**. 3. ed. Pearson, 2005.

FORBELLONE, A. L. V.; EBERSPACHER, H. F. **Lógica de programação**: a construção de algoritmos e estruturas de dados. 3. ed. Pearson, 2005.

GUEDES, S. (Org.). **Lógica de programação algorítmica**. Pearson, 2014.

LEE, V.; SCHNEIDER, H.; SCHELL, R. **Aplicações móveis**: arquitetura, projetos e desenvolvimento. Pearson, 2005.

MELO, A. C. V. de; SILVA, F. S. C. da. **Princípios de linguagens de programação**. Blucher, 2014.

MENEZES, A. M. de. **Os paradigmas de aprendizagem de algoritmo computacional**. Blucher, 2015.

PUGA, S.; RISSETTI, G. **Lógica de programação e estruturas de dados, com aplicações em Java**. Pearson, 2016.