Entrada e Saída de dados

A Entrada e Saída dos dados permitem que um programa faça interação com o usuário. É possível utilizar o teclado do computador ou notebook para inserir dados a uma variável, ou usar arquivos de disco, entre outras formas.

Para a Entrada de dados, usamos as classes que permitem fazer a leitura dos dados, sendo elas: Scanner, BufferedReader e inputStreamReader.

A Saída de dados trata-se da exibição em tela do valor da variável, gravado na Entrada dos dados. As classes que utilizamos para realizar a Saída dos dados, e que fornecem métodos para envio dos dados são: System.out, PrintWriten e FileOutputStream.

Utilizando as classes de Entrada e Saída de dados, podemos implementar maneiras diferentes de exibir os dados; e até podemos fazer operações matemáticas utilizando a entrada e saída de dados.

Veremos neste conteúdo como podemos fazer uma Entrada e Saída de dados simples, utilizando algumas das classes citadas acima. Tenha o objetivo de entender como a linguagem precisa ser direcionada para realizar alguma operação de entrada ou saída.

* 1. Lembre-se

O processamento de dados é desenvolvido em três etapas, que são: Entrada, Processamento e Saída.

**Processamento**

**Saída**

**Entrada**

Entrada: armazena o dado que o usuário digitar pelo teclado, sendo um Comando de Entrada de Dados.

Processamento: podem ser processamentos de cálculo ou leitura dos dados.

Saída: mostra o resultado do processamento na tela do computador, sendo um Comando de Saída.

1.2 Entrada de dados

Utilizamos a classe Scanner para obter informações do usuário mediante o seu teclado. Esta classe pertence ao Pacote java.util, e precisamos instanciar um objeto desta classe para utilizarmos a Classe.

Veja a sintaxe do Scanner no exemplo abaixo:

Scanner nomeScanner = new Scanner(System.in);

Na aula anterior aprendemos sobre o método main(), e será dentro dele que vamos implementar o Scanner.

Public static void main(String[] args){

// Implementando o Scanner

Scanner leia = new Scanner(System.in);

}

No exemplo acima foi criado um objeto chamado leia, e este objeto pertence a Classe Scanner.

Este objeto foi instanciado com o objetivo de ler o que for inserido no teclado, usando o padrão de entrada (System.in). Na Classe System existe uma variável chamada in, onde a sua função é armazenar o dado que dar entrada no teclado ou em outro arquivo de texto.

Então, vamos resumir o código que temos até o momento, conforme o próximo quadro:

* Temos a importação do pacote java.util.Scanner,
* Temos a nossa Classe EntradaDados,
* Temos o método main definido,
* E temos instanciado o objeto Scanner.

import java.util.Scanner;

public class EntradaDados{

Public static void main(String[] args){

// Implementando o Scanner

Scanner leia = new Scanner(System.in);

}

}

1.3 Método de Entrada da Classe Scanner

Veremos alguns dos métodos de entrada mais usados da classe Scanner.

nextInt(): Faz a leitura de números inteiros.

Int num = Scanner.nextInt();

nextDouble(): Faz a leitura de números de ponto flutuante de precisão dupla.

double num = Scanner.nextDouble();

nextFloat(): Faz a leitura de números de ponto flutuante.

float num = Scanner.nextFloat();

nextLong): Faz a leitura de números inteiro longo.

double num = Scanner.nextDouble();

nextBoolean(): Faz a leitura de valor booleano.

boolean bool = Scanner.nextBoolean();

next(): Faz a leitura de valores tipo String.

String str = Scanner.next();

nextLine(): Faz a leitura de uma linha inteira como uma String.

String line = Scanner.nextLine();

nextByte(): Faz a leitura de valores como byte.

byte b = Scanner.nextByte();

nextShort(): Faz a leitura de valores como short.

short s = Scanner.nextShort();

nextBigInteger(): Faz a leitura de valores como BigInteger.

BigInteger bigInt = Scanner.nextBigInterger();

nextBigDecimal(): Faz a leitura de valores como BigDecimal.

BigDecimal bigDec = Scanner.nextBigDecimal();

hasNext(): Declara verdadeiro se houver outro valor na entrada.

if (Scanner.hasNext()){

// Processa o próximo valor

}

hasNextInt(): Retorna verdadeiro se o valor de entrada puder ser interpretado como inteiro.

if (Scanner.hasNextInt()){

int num = scanner.nextInt();

}

hasNextDouble(): Retorna verdadeiro se o próximo valor de entrada puder ser interpretado como um número de ponto flutuante de precisão dupla.

if (Scanner.hasNextDouble()){

double num = scanner.nextDouble();

}

1.4 Exemplo do uso da Classe Scanner

Veja abaixo o exemplo de uso da Classe Scanner implementada com os métodos: nextInt(), nextDouble(), next() e nextLine().

import java.util.Scanner;

public class ScannerExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Digite um número inteiro: ");

int numInt = scanner.nextInt();

System.out.print("Digite um número de ponto flutuante: ");

double numDouble = scanner.nextDouble();

System.out.print("Digite uma palavra: ");

String word = scanner.next();

System.out.print("Digite uma linha de texto: ");

scanner.nextLine(); // Consumir a nova linha restante

String line = scanner.nextLine();

System.out.println("Número inteiro: " + numInt);

System.out.println("Número de ponto flutuante: " + numDouble);

System.out.println("Palavra: " + word);

System.out.println("Linha de texto: " + line);

scanner.close();

}

}

Mas o que foi feito no código acima?

Utilizamos os métodos nextInt(), nextDouble(), next() e nextLine() para armazenar os valores que o usuário escreveu via teclado:

nextInt(): realiza a leitura do número inteiro fornecido pelo usuário.

nextDouble(): realiza a leitura de um número ponto flutuante fornecido pelo usuário.

next(): realiza a leitura de uma única palavra fornecida pelo usuário.

nextLine(): realiza a leitura de texto fornecida pelo usuário. Note que o scanner.nextLine() adicional antes da leitura da linha de texto é necessário para consumir a nova linha restante após next() ou nextInt().

Ao utilizar a Classe Scanner, é importante utilizar o comando scanner.close() para terminar de utilizar a classe e liberar os recursos associados.

2.0 Saída de Dados

Referências

[Drops] Java – I/O: Entrada e Saída de dados

https://www.linkedin.com/pulse/drops-java-io-entrada-e-sa%C3%ADda-gabriel-d-emery/

Java: Como ler e apresentar dados com os comandos de entrada e saída

https://www.devmedia.com.br/java-como-ler-e-apresentar-dados-com-os-comandos-de-entrada-e-saida/22266

Entrada de Dados: Classe Scanner

https://www.devmedia.com.br/entrada-de-dados-classe-scanner/21366

Entrada e Saída de dados

https://github.com/Leon4rdoalves/CookBook-Java/blob/main/04.md