Linguagem Orientada a Objetos

Arrays e Strings

Prof. Ms. Leonardo Rocha

- Unidade de Ensino: 4
- Competência da Unidade: Compreender e utilizar recursos de arrays e vetores e banco de dados
- Palavras-chave: Array, vetor, banco de dados, CRUD
- Título da Teleaula: Arrays e Strings
- Teleaula nº: 4

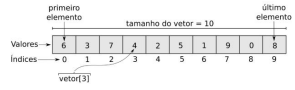
Contextualização

Compreender o uso de Arrays e vetores
Construir projeto de vetor de Strings
Leitura de arquivos
Criação de Banco de dados
Ferramentas para criação e manipulação de banco de dados

Arrays

Unidimensionais

A linguagem Java permite a criação de arrays unidimensionais também chamados de vetores. Um vetor agrupa um conjunto de dados de um mesmo tipo organizados em linha ou coluna, e por trabalhar com dados do mesmo tipo, falamos que o vetor é uma estrutura de dados homogênea.



Criação

Operação sobre Vetores	Exemplo de Código
Declaração de um vetor.	`tipo nomeVetor[];`
Alocação de espaço de um vetor.	`nomeVetor = new tipo[tamanho];`
Declaração e alocação de um vetor.	`tipo nomeVetor[] = new tipo[tamanho];`
Acesso de uma posição do vetor.	`nomeVetor[indice]`
Atribuição de um valor a uma posição.	`nomeVetor[indice] = valor;`
Acesso ao tamanho de um vetor.	`nomeVetor.length`

Exemplo - Vetor

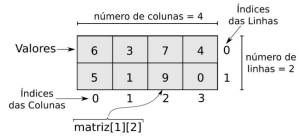
```
int vet[] = new int[5];
vet[0] = 6;
vet[1] = 3;
vet[2] = 7;
vet[3] = 4;
vet[4] = 2;
for (int i = 0; i < vet.length; i++) {
    System.out.println("Array[" + i + "]: " + vet[i]);
}
```


Formas de criar e inicializar

Tipo do Vetor	Exemplos usando a forma 1	Exemplos usando a forma 2
Inteiro	`int vet[] = {1, 5, 3, 4};`	`int[] vet = {1, 5, 3, 4};`
Real	`double vet[]={2.0, 5.3, 3.8};`	`double[] vet={2.0, 5.3, 3.8};`
Caractere	`char vet[] = {'a', 'e', 'i'};`	`char[] vet = {'a', 'e', 'i'};`

Multidimensionais

Os arrays multidimensionais, também chamados de matrizes, também são estruturas de dados muito importantes. Uma matriz agrupa um conjunto de dados de um mesmo tipo organizados em forma de linhas e colunas.



Criação

Operação sobre Matrizes	Exemplo de Código
Declaração de uma matriz.	`tipo nomeMat[][];`
Alocação de espaço de uma matriz.	`nomeMat = new tipo[tamX][tamY];`
Declaração e alocação de uma matriz.	`tipo nomeMat[][] =new tipo[tamX][tamY];`
Acesso de uma posição da matriz.	`nomeMat[indiceX][indiceY]`
Atribuição de um valor a uma posição.	`nomeMat[indiceX][indiceY] = valor;`
Acesso ao número de linhas.	`nomeMat.length`
Acesso ao número de colunas da i-ésima linha.	`nomeMat[i].length`

Exemplo - Matriz

```
double mat[][] = {{1.5, 5.2}, {3.6, 4.9}, {2.4, 8.1}};
for (int i = 0; i < mat.length; i++) {
    for (int j = 0; j < mat[i].length; j++) {
        System.out.println("M[" +i+ "][" +j+ "]: " + mat[i][j]);
    }
}
```


Exemplo - Utilizando arrays

```
package sp1_aula4;

public class SP1_aula4 {

    public static void main(String[] args) {


        String[] vet = new String[]{"leonardo", "João", "Maria"};
        System.out.println("Os nomes informados foram: ");
        for(int cont = 0; cont < vet.length; cont++)
        {
            System.out.println(vet[cont]);
        }
    }
}
```

Strings

Definição

A classe **String** possui uma importante característica, que é ser imutável. Isso quer dizer que uma vez atribuído um valor literal para a variável, existirá uma memória, só que sem nenhum objeto apontando para si. Assim, se você construir uma aplicação que modifique constantemente o valor da variável, você terá um desempenho ruim em termos de gastos de memória e de processamento .

StringBuilder

Outra classe que existe para manipular strings e é uma boa alternativa à classe String se desejar que o conteúdo da string seja mutável.

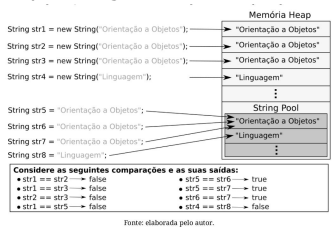
StringBuffer

é uma boa alternativa às outras duas classes se desejar que o conteúdo seja mutável e necessitar de uma aplicação thread safe.

Exemplo

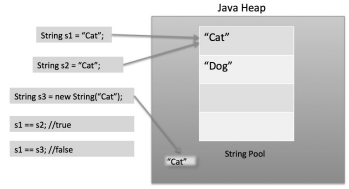
```
1 String str1 = new String("Orientação a objetos");
2 String str2 = "Orientação a objetos";
3 StringBuilder str3 = new StringBuilder("Orientação a objetos");
4 StringBuilder str4 = "Orientação a objetos"; //não é aceito (dá
  erro)
5 StringBuffer str5 = new StringBuffer("Orientação a objetos");
6 StringBuffer str6 = "Orientação a objetos"; //não é aceito (dá
  erro)
```


Heap x String Pool



Fonte: elaborada pelo autor.

Heap x String Pool



Leitura de arquivos e banco de dados

Definição

Um BD pode ser pensado como uma forma de armazenar e consultar dados, assim, um simples arquivo pode ser entendido como uma forma primitiva/simples de se manter os dados. Dessa maneira, veremos em primeiro lugar, como ler um arquivo de texto (.txt) e, em seguida, manipular um arquivo CSV (Comma-Separated Values ou Valores Separados por Vírgula).

Leitura de texto

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class ManipulaArquivosTexto {
    public static void main(String[] args) {
        try {
            File arquivo = new File("dados/poema.txt");
            Scanner scanner = new Scanner(arquivo);
            while (scanner.hasNextLine()) {
                String linha = scanner.nextLine();
                System.out.printf("%s%n", linha);
            }
        } catch (FileNotFoundException ex) {
            System.out.printf("Erro abertura do arquivo: %s.%n",
                    ex.getMessage());
        }
    }
}
```


Leitura de CSV

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class ManipulaArquivosCSV {
    public final String DELIMITADOR_PONTO_VIRGULA = ";";
    public final String DELIMITADOR_VIRGULA = ",";
    public static void main(String[] args) {
        ManipulaArquivosCSV csv = new ManipulaArquivosCSV();
        List<List<String>> registroDados = csv.leitura();
        csv.imprimeDados(registroDados);
    }
```


Leitura de CSV

```java
    public List<List<String>> leitura() {
        List<List<String>> registroDados = new ArrayList<>();
        try {
            File arquivo = new File("dados/dadosclientes.csv");
            Scanner sc = new Scanner(arquivo);
            while (sc.hasNextLine()) {
                String linha = sc.nextLine();
                registroDados.add(getRegistroDaLinha(linha));
            }
        } catch (FileNotFoundException ex) {
            System.out.printf("Erro abertura do arquivo: %s.%n",
                    ex.getMessage());
            System.exit(0);
        }
        return registroDados;
    }
```


Leitura de CSV

```java
    private List<String> getRegistroDaLinha(String linha) {
        List<String> listValores = new ArrayList<String>();
        try (Scanner linhaScanner = new Scanner(linha)) {
            linhaScanner.useDelimiter(DELIMITADOR_PONTO_VIRGULA);
            while (linhaScanner.hasNext()) {
                listValores.add(linhaScanner.next());
            }
        }
        return listValores;
    }
    private void imprimeDados(List<List<String>> registroDados){
        for (List<String> lista : registroDados){
            for (String elemento : lista){
                System.out.printf("|%15s ", elemento);
            }
            System.out.println("|");
        }
    }
}
```

25

Banco de Dados

Os bancos de dados, geralmente, utilizam uma linguagem de consulta, e a principal linguagem de consulta utilizada é a SQL (Structured Query Language ou Linguagem de Consulta Estruturada), que é uma linguagem padrão para armazenamento, manipulação e recuperação de dados em BD, que pode ser utilizada para trabalhar com diversos BDs, como: MySQL, MariaDB, Microsoft SQL Server, Oracle, PostgreSQL, entre outros.

26

CRUD

Existem 4 operações básicas que podem ser realizadas sobre um banco de dados:

CREATE - Diz respeito à criação de dados
READ - Leitura ou consulta de dados
UPDATE - Atualização desses dados
DELETE - Deleção ou destruição dos dados.

Ferramentas

28

Ferramentas

[1]

[2]

[3]

29

```
public void inserirCliente(Connection conn, Cliente cliente)
{

try {
   Statement st = conn.createStatement();
   String sql = "INSERT INTO `clientes` (nome, " + "idade, profissao, cidade, estado) VALUES ('"+ cliente.getNome() +"', "
+ cliente.getIdade() + ", '" + cliente.getProfissao()+ "', '" + cliente.getCidade()+ "', '" + cliente.getEstado()+ "')";
   st.executeUpdate(sql);
   st.close();
   } catch (SQLException ex) {
     System.out.println(ex);
   }
}
```

Gravando dados no BD - Exemplo

30

Comandos SQL

Operações	Exemplo de Comando
Criar BD	`CREATE DATABASE 'nome_bd';`
Destruir BD	`DROP DATABASE 'nome_bd';`
Criar Tabela	`CREATE TABLE 'nome_bd'.'nome_tbl' ('col1' TIPO1, 'col2' TIPO2, ... 'colN' TIPON);`
Destruir Tabela	`DROP TABLE 'nome_bd'.'nome_tbl';` ou `DROP TABLE 'nome_tbl';`
Consultar Tudo	`SELECT * FROM 'nome_bd'.'nome_tbl';` ou `SELECT * FROM 'nome_tbl';`

31

Ferramentas

[1]

[2]

32

Aumentar as posições no Array

```java
package sp1_aula4;

public class pushArray {

    public static String[] push(String[] array, String push) {
    String[] nome = new String[array.length + 1];
    for (int i = 0; i < array.length; i++)
        nome[i] = array[i];
        nome[array.length] = push;
        return nome;

    }
}
```

33

Imprimindo um vetor

```java
package sp1_aula4;

public class SP1_aula4 {

    public static void main(String[] args) {

        String[] vet = new String[]{"leonardo", "João", "Maria"};
        vet = pushArray.push(vet, "Fernanda");
        System.out.println("Os nomes informados foram: ");
        for(int cont = 0; cont < vet.length; cont++)
        {
            System.out.println(vet[cont]);
        }
    }
}
```

Recapitulando

35

Compreender o uso de Arrays e vetores
Construir projeto de vetor de Strings
Leitura de arquivos
Criação de Banco de dados
Ferramentas para criação e manipulação de banco de dados