DOMAIN DRIVEN DESIGN

Prof. Rafael Desiderio

10 – COLLECTIONS FRAMEWORK

Collections Framework

- O que são Collections Framewok?
  - Localizadas no pacote “java.util”, **Collections Framework** (**Coleções)** são **estruturas de dados**, utilizadas para **agrupar objetos**, que permitem de **maneira eficiente e prática** o armazenamento e organização de **objetos;**
  - De forma **contrária ao vetor(array),** elas permitem que um número **arbitrário (não fixo)** de objetos seja armazenado numa estrutura;
  - **Coleções** podem ser utilizadas para representar **vetores, listas, pilhas, filas, mapas, conjuntos** e outras estruturas de dados.
  - A **escolha de um tipo de estrutura** depende dos requisitos do **problema** que se deseja resolver.

Collections Framework

- Várias aplicações necessitam utilizar as **coleções de objetos**, por **exemplo**: agendas pessoais, catálogos de bibliotecas, cadastro de funcionários;

- São amplamente utilizadas no **acesso a dados em bases de dados,** principalmente no **resultado de buscas**;

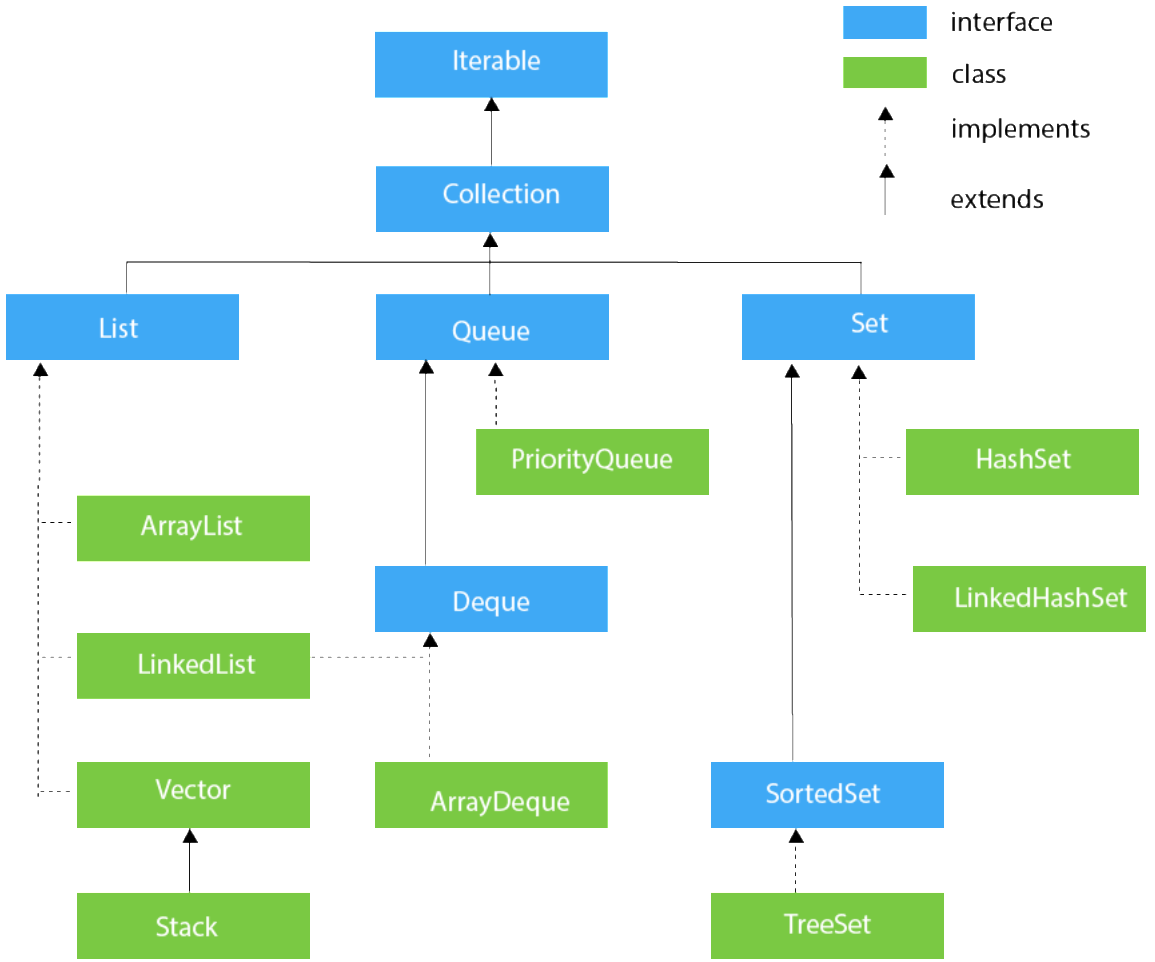
Collections Framework

- As **coleções** são definidas por meio de **interfaces**;

- Como as **interfaces** apenas fornecem **um contrato entre o implementador** e **os clientes em relação às funcionalidades** de uma abstração, toda **implementação** de uma coleção é realizada numa classe concreta;

- Algumas das **principais interfaces** usadas em coleções são:
  - **Collection**
    - **Set**
    - **List**
  - **Map**

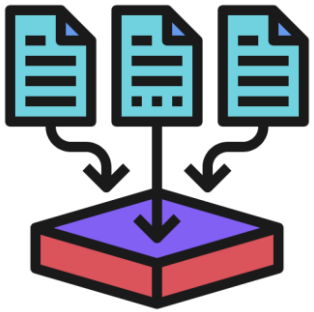
Collections Framework



Tipos de Coleção

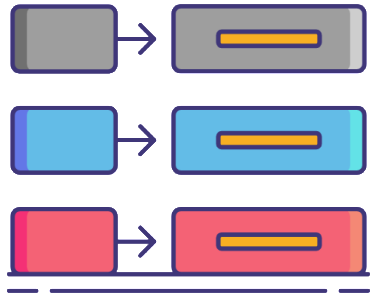
- No **Java**, as **coleções** podem ser classificadas em **duas categorias**: as que implementam a interface **Collection** e as que implementam a interface **Map;**

- **As subinterfaces principais de Collection são:**
  - **List** - representa uma lista de objetos.
    - Implementação mais utilizada: **ArrayList**.
  - **Set** - representa um conjunto de objetos únicos (os objetos não podem se repetir).
    - Implementação mais utilizada: **HashSet**.



Tipos de Coleção

- A interface **Map** representa uma tabela Hash, que **guarda valores compostos por [chave, valor],** sua principal subinterface é:
    - **SortedMap** - representa um mapa ordenado;
        - Implementação mais utilizada: **HashMap;**



Interface Collection

- Denominador comum de **todas as coleções**:
  - **Não há implementações** “diretas” dessa interface;

- Implementações dos tipos **básicos de coleções** tem construtores que recebem um parâmetro do tipo **Collection;**

- **Permite a criação de uma coleção contendo inicialmente todos os elementos da coleção especificada como parâmetro**, independentemente de seu tipo ou implementação;

Principais Métodos

Método	Descrição
add	Adiciona um objeto a coleção
clear	Remove todos objetos da coleção
contains	Verifica se a coleção contém o objeto determinado
isEmpty	Verifica se a coleção está vazia
remove	Remove um objeto da coleção
size	Retorna a quantidade de objetos na coleção
toArray	Retorna um array contendo os elementos da coleção

Interface List

- **Representa uma sequência de elementos** - ordered collection - pode conter elementos **duplicados**;
- Permite controlar a **posição de inserção** de um elemento e acessar elementos por sua posição;
- Permite procurar por um **objeto específico na lista e retornar sua posição numérica;**
- A classe **ArrayList** implementa esta interface;

Principais Métodos

Método	Descrição
add*	Adiciona um objeto numa determinada posição
get	Retorna o objeto localizado numa determinada posição
remove*	Remove um objeto localizado numa determinada posição
set	Coloca um objeto numa determinada posição (substitui objetos)
indexOf	Retorna a posição de um objeto na lista
lastIndexOf	Retorna a última posição de um objeto na lista
subList	Retorna parte de uma lista

**método sobrecarregado*

Implementação - ArrayList

- Implementação da interface **List;**
- Armazena seus elementos em um **array** que cresce **dinamicamente;**
- Ótimo **desempenho para acesso em listas** sem muitas modificações no início e meio;

```
ArrayList lista = new ArrayList();
lista.add("A");
lista.add("C");
lista.add("B");
lista.remove(1);
System.out.println(lista.get(1)); //B
```

List – ArrayList - Exemplo

- **Passo 1 – No pacote beans elaboramos a classe Produto:**

```
package br.com.fiap.beans;
public class Produto {
private String descricao;
private String marca;
private double valor;
private int quantidade;
public Produto() {
super();
}
public Produto(String descricao, String
marca, double valor, int quantidade) {
super();
this.descricao = descricao;
this.marca = marca;
this.valor = valor;
this.quantidade = quantidade;
}
// Continua com Getters e Setters...
```

List – ArrayList - Exemplo

- **Passo 2 – Na classe TesteLista do pacote main, podemos deixar prontos os metodos (static) para facilitar a entrada de dados (JOptionPane):**

```
public class TesteLista {

static String texto(String j) {
return JOptionPane.showInputDialog(j);
}

static double decimal(String j) {
return Double.parseDouble(JOptionPane.showInputDialog(j));
}

static int inteiro(String j) {
return Integer.parseInt(JOptionPane.showInputDialog(j));
}

public static void main(String[] args) {
// continua no próximo Slide...
```

List – ArrayList - Exemplo

- **Passo 3 – Instanciamos o ArrayList para Produto, e preparamos o objProduto para ser instanciado dentro do (do/while) para fazer a entrada do que será digitado :**

```
public static void main(String[] args) {
List<Produto> listaProdutos = new ArrayList<Produto>();
Produto objProduto;
do {
// Entrada
objProduto = new Produto();
objProduto.setDescricao(texto("Digite a Descrição"));
objProduto.setMarca(texto("Digite a marca"));
objProduto.setValor(decimal("Digite o Valor"));
objProduto.setQuantidade(inteiro("Digite a quantidade"));
listaProdutos.add(objProduto);
// Escolhemos entre ecerrar ou adicionar mais um produto no carrinho de compras
} while (JOptionPane.showConfirmDialog(null, "Mais produto no carrinho?",
"Carrinho de Compras",
JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE) == 0);
// Continua no próximo Slide...
```

List – ArrayList – Exemplo

- **Passo 4 – Após o (do/ while) obteremos as saídas dos produtos que foram inseridos, através do foreach que percorrerá todas as entradas dos produtos que foram digitados:**

```
        // O foreach percorre todos os elementos do Produto para serem exibidos
        for (Produto p : listaProdutos) {
            System.out.println("Descrição: " + p.getDescricao() + "\n" +
            "Marca: " + p.getMarca() + "\n" +
            "Valor: " + p.getValor() + "\n" +
            "Quantidade: " + p.getQuantidade());
        }
    }
}
```

Interface Set

- Uma coleção que **não pode conter elementos duplicados;**
- Corresponde à abstração de um **conjunto;**
- Contém somente **os métodos herdados** da interface **Collection;**
- A classe **HashSet** implementa esta interface;

Método	Descrição
add	Adiciona um objeto no Set
clear	Remove todos objetos do Set
contains	Verifica se o Set possui um objeto determinado
isEmpty	Verifica se o set está vazio
remove	Remove um objeto do Set
size	Retorna a quantidade de objetos no Set
toArray	Retorna um array contendo os objetos do Set

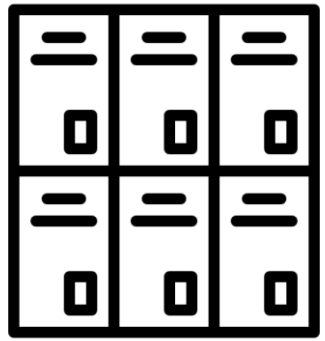
Implementação - HashSet

- Implementação da interface **Set**.
- Armazena seus elementos em uma **tabela *hash;***
- Elimina **automaticamente objetos duplicados** inseridos na coleção;

```
HashSet set = new HashSet();
set.add("um");
set.add("dois");
set.add("tres");
set.add("dois"); //será eliminado da coleção
set.add("quatro");
set.remove("tres");
System.out.println("Tamanho do set: " + set.size());
```

Interface Map

- Representa um objeto que **mapeia chaves em valores**;
- Um objeto **Map não** pode conter **chaves duplicadas;**
  - cada chave é **mapeada para um único valor**;
- A classe **HashMap** implementa esta interface;



Principais métodos

Método	Descrição
clear	Remove todos os mapeamentos
containsKey	Verifica se uma chave já está presente no mapeamento
containsValue	Verifica se um valor já está presente no mapeamento
get	Retorna o valor associado a uma chave determinada
isEmpty	Verifica se o mapeamento está vazio
keySet	Retorna um Set contendo as chaves
put	Adiciona um mapeamento
remove	Remove um mapeamento
size	Retorna o número de mapeamentos
values	Retorna uma Coleção contendo os valores dos mapeamentos

Implementação – HashMap

- Implementação da interface **Map;**
  - as entradas são armazenadas em uma tabela *hash;*
  - permite o uso de valores **null;**
- **Não** garante ordenação;

```
HashMap map = new HashMap();
map.put("1", "um");
map.put("2", "dois");
map.put("3", "tres");
map.put("4", "quatro");
map.remove("4");
System.out.println(map.get("1"));
```

Vamos à Prática!

- Seguindo como exemplo o projeto completo que vimos anteriormente, faça um projeto que cadastre alunos com **nome**, **rm** e **turma** utilizando o **Arraylist** da **interface List**, com os laços de repetição **while** e **do/while.**

FIAP

"Conhecer a si mesmo é o começo de toda a sabedoria". - Aristóteles