Aula 11 - Collections Framework

O que são Collections?

- 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;

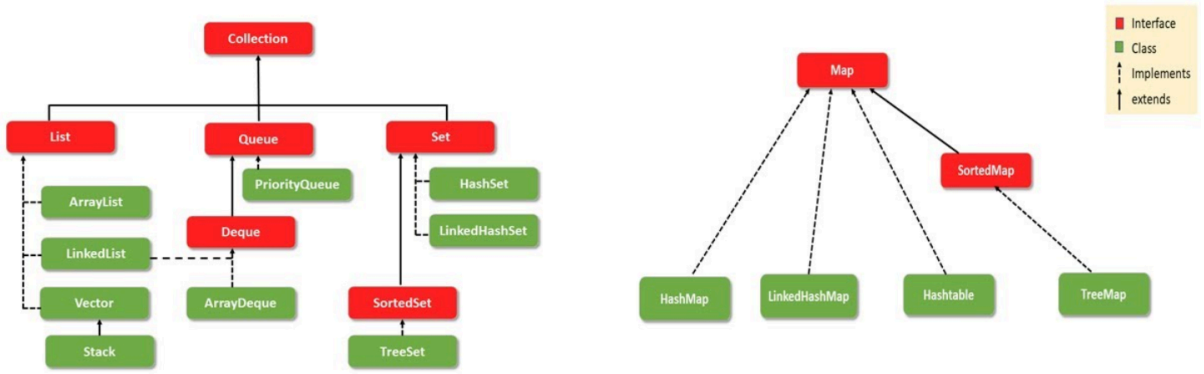
- Coleções podem ser utilizadas para representar vetores, listas, pilhas, filas, mapas, conjuntos e outras estruturas de dados.

- A escolha de um tipo de estrutura depende dos requisitos do problema que se deseja resolver.

- 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;

Arquitetura de classes e interfaces das Collections



1. HashMap

- Representa um objeto que mapeia chaves em valores.
- É semelhante ao Dicionário no Python;
- A classe HashMap implementa a interface Map;
- As entradas são armazenadas em uma tabela hash (em memória);
- Permite o uso de valores null;
- Não garante ordenação;

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

Exemplo de utilização:

```
import java.util.HashMap;

public class TesteHashMap {

    public static void main(String[] args) {

        HashMap<String, String> map = new HashMap<String, String>();

        map.put("nome", "Joseffe");
        map.put("email", "joseffe@gmail.com");
        map.put("funcao", "Professor");
        map.put("linkedin", "linkedin.com/joseffe");
        map.put("rg", null);

        map.remove("linkedin");

        System.out.println(map.get("funcao"));

        System.out.println(map.get("linkedin"));

        System.out.println(map);

        // For each com Lambda function
        map.forEach((key, value) -> {
            System.out.print(key + "=" + value + " ");
          });
```

```
        // For each convencional
        for (HashMap.Entry<String, String> m : map.entrySet()) {

            System.out.println(m.getKey() + " = " + m.getValue());
        }
    }
}
```

2. HashSet

- Uma coleção que não pode conter elementos duplicados;
- Corresponde à abstração de um conjunto;
- A classe HashSet implementa a interface Set;
- Armazena seus elementos em uma tabela hash;
- Elimina automaticamente objetos duplicados inseridos na coleção;

Principais métodos:

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

Exemplo de utilização:

```
import java.util.HashSet;
import java.util.Iterator;

public class TesteHashSet {

    public static void main(String[] args) {

        HashSet<String> set = new HashSet<String>();

        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());

        Iterator<String> itr = set.iterator();

        while (itr.hasNext()) {
            System.out.println(itr.next());
        }
    }
}
```

3. ArrayList

- 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;
- Implementação da interface List;
- Armazena seus elementos em um array que cresce dinamicamente;
- Ótimo desempenho para acesso em listas sem muitas modificações;

Principais métodos:

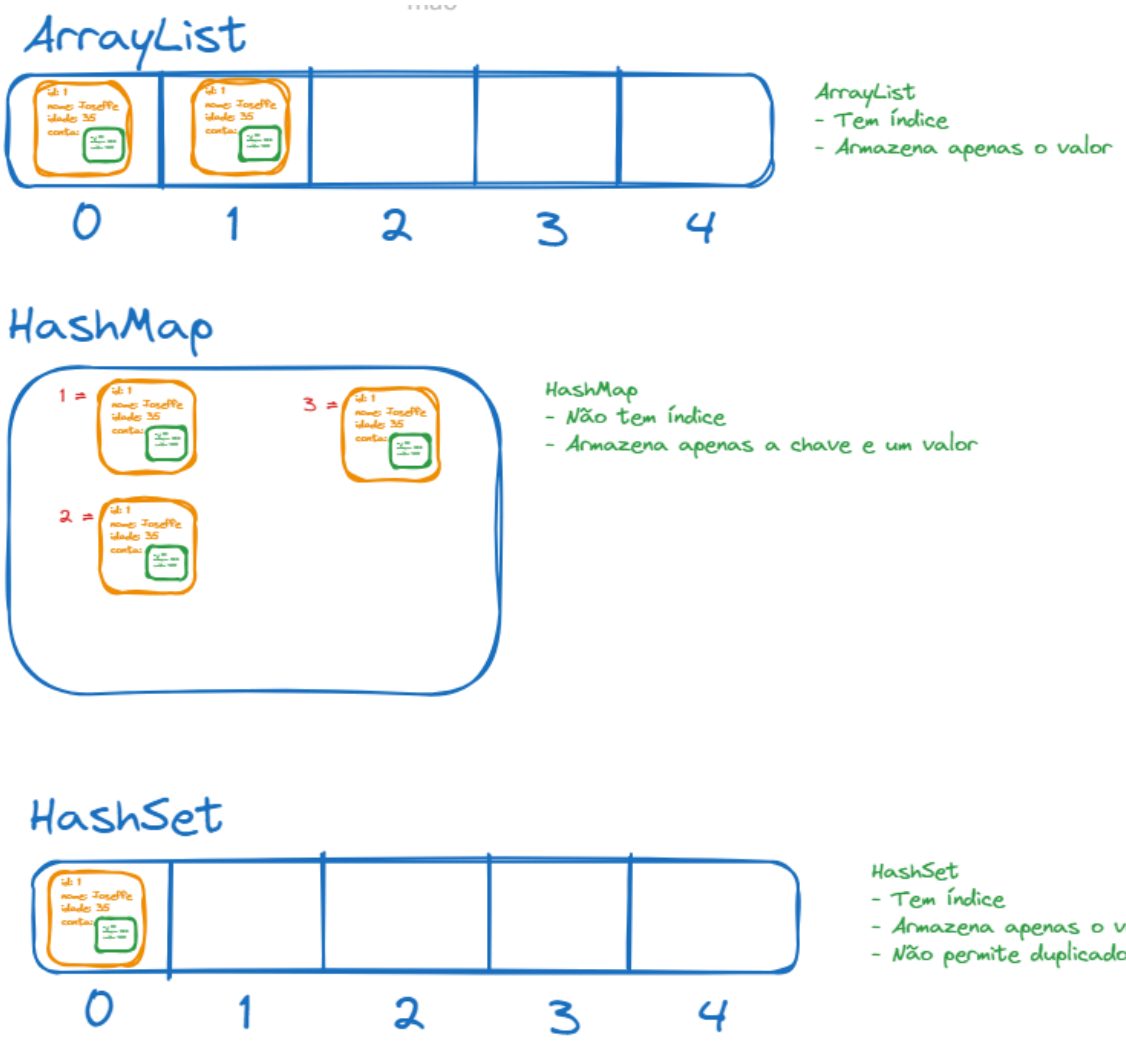
Método	Descrição
add*	Adiciona um objeto numa determinada posição
get	Retorna o objeto localizado numa determinada posição
remove*	Remove um objeto localizado numa determinada posição
set	Coloca um objeto numa determinada posição (substitui objetos)
indexOf	Retorna a posição de um objeto na lista
lastIndexOf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

Exemplo de utilização:

```java
import java.util.ArrayList;

public class TesteArrayList {

    public static void main(String[] args) {

        ArrayList<String> lista = new ArrayList<String>();

        lista.add("A");
        lista.add("C");
        lista.add("B");

        lista.remove(1);

        System.out.println(lista.get(1)); //B

        for(String a: lista) {
            System.out.println(a);
        }
    }
```

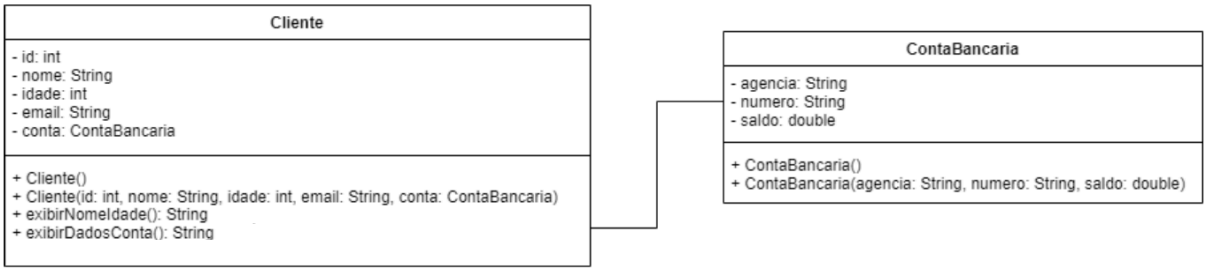
```
}
```

Conclusão: Se compararmos o HashMap e o HashSet com o ArrayList, temos o HashMap e o HashSet mais rápidos que o ArrayList. Embora o HashMap e o HashSet sejam mais lentos no início e ocupem mais memória, eles são mais rápidos para grandes volumes de dados.



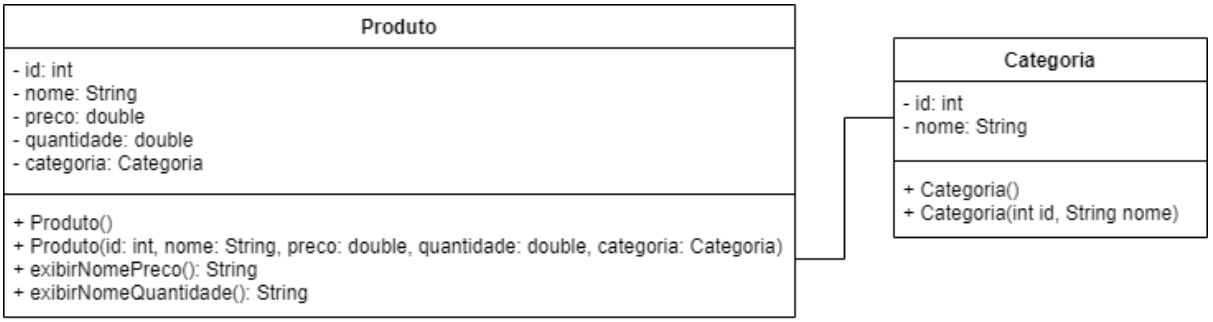
Exercícios

Utilizando **HashMap,** crie um sistema com as classes conforme o Diagrama de Classe (UML) abaixo. Crie Getters e Setters para todos os atributos das classes. Crie um programa que utilize essas classes para cadastrar clientes e pergunte para cada cliente se ele tem ou não conta bancária. Caso o cliente tenha, permita ele cadastrar os dados da conta bancária. Ao final, exibir todos os clientes e suas respectivas contas bancárias, se houver.



Correção:
https://drive.google.com/drive/u/1/folders/1EGJl1AjSZGRPfwWPMCyj22b2HJruMzr_

Utilizando **HashMap,** crie um sistema com as classes conforme o Diagrama de Classe (UML) abaixo. Crie Getters e Setters para todos os atributos das classes. Crie um programa que utilize essas classes para cadastrar produtos e pergunte para cada produto se ele tem ou não uma categoria. Caso o produto tenha, permita ele cadastrar os dados da categoria. Ao final, exibir todos os produtos e suas respectivas categorias, se houver.



Correção:
https://drive.google.com/drive/u/1/folders/1Fu3hmJhBieq1WjFOwAB3LMCbPVBYA0JV