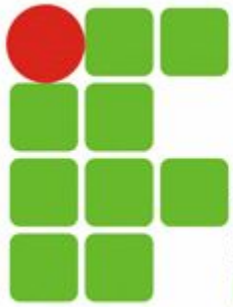


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
FLUMINENSE

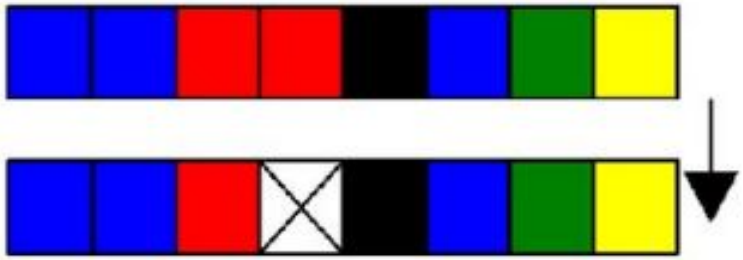
API
Collections Framework

Arrays x Problemas

Manipular arrays é bastante trabalhoso. Essa dificuldade aparece em diversos momentos:

- Não conseguimos saber quantas posições do array já foram "populadas" sem criar, para isso, métodos auxiliares.
- É impossível buscar diretamente por um determinado elemento cujo índice não se sabe;
- Depois de criado, **o array não pode ser redimensionado em Java**.

Arrays x Problemas



Retire a quarta Conta

conta[3] = **null**;

Supondo que os dados armazenados representem contas, o que acontece quando precisarmos inserir uma nova conta no banco?

- Precisaremos procurar por um espaço vazio?
- Guardaremos as posições vazias em alguma estrutura de dados externa?
- E se não houver espaço vazio?
- Teríamos de criar um array maior e copiar os dados do antigo para ele?

Arrays x Problemas

Além dessas dificuldades que os arrays apresentavam, faltava um conjunto robusto de classes para suprir a necessidade de estruturas de dados básicas, como listas ligadas e tabelas de espalhamento.

Para resolver essas limitações com o uso de ARRAYS, foi criado um conjunto de classes e interfaces conhecido como **Collections Framework**, que reside no pacote **java.util**.

- A API Collections framework é robusta e possui diversas classes que representam estruturas de dados avançadas.
- A API do Collections é uma arquitetura unificada para representação e manipulação de coleções, permitindo que elas sejam manipuladas independentemente dos detalhes de sua implementação.

Collections Framework

Collections Framework

- Conjunto de interfaces, implementações e algoritmos
- Vantagens
  - Reduz esforço de programação
  - Aumenta velocidade e qualidade na programação
  - Permite interoperabilidade entre API´s
  - Reduz esforço para aprender uma nova API
  - Reduz esforço para criar uma nova API
  - Promove reuso

Atenção

Collections Framework contem várias classes e interfaces.

Repare que temos:

- **Classe Collections** (java.util.Collections): esta classe consiste exclusivamente de métodos estáticos que manipulam coleções.

- **Interface Collection**: interface raiz da hierarquia de coleções.

Veja mais em:
https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html
https://docs.oracle.com/javase/8/docs/technotes/guides/collections/index.html

Conceito de Coleções

- Coleção:
  - É um objeto que representa um grupo de objetos.
  - Os objetos são armazenados em uma estrutura de dados.
  - Utilizadas para armazenar, recuperar e manipular elementos.

Coleções

O que é uma coleção?

- Tipicamente representam itens de dados que naturalmente formam um grupo.
- Exemplo:
  - Uma coleção de cartas de um baralho
  - Uma coleção de e-mails
  - uma agenda telefônica (coleção de nomes e telefones)

Coleções

O que é uma coleção?

Uma coleção é uma estrutura de dados que permite armazenar vários objetos.

Em Java, a coleção também é um objeto.

As operações que podem ser feitas em coleções variam, mas normalmente incluem:

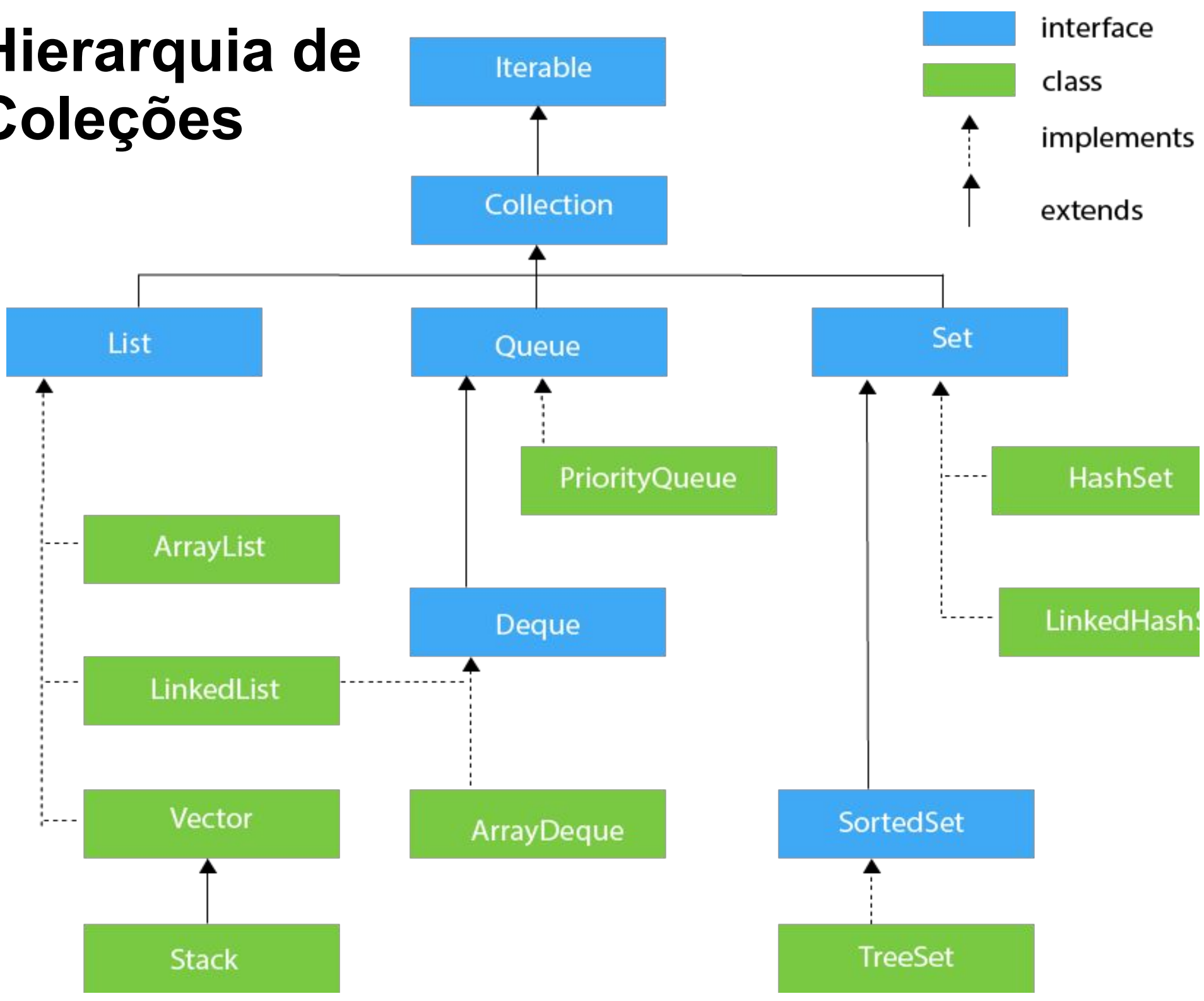
Adição de elementos;

Remoção de elementos;

Acesso aos elementos;

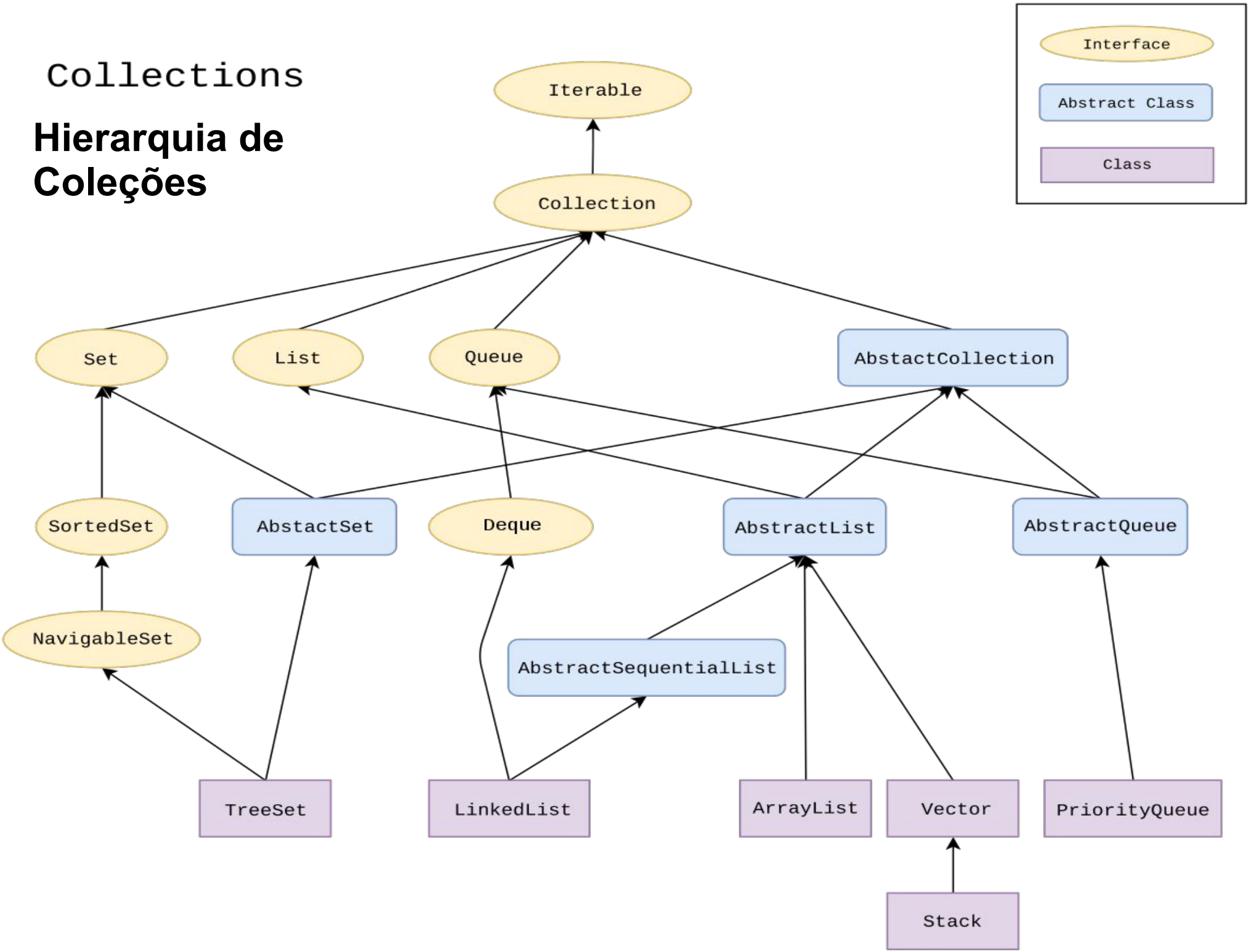
Pesquisa de elementos;

Hierarquia de Coleções

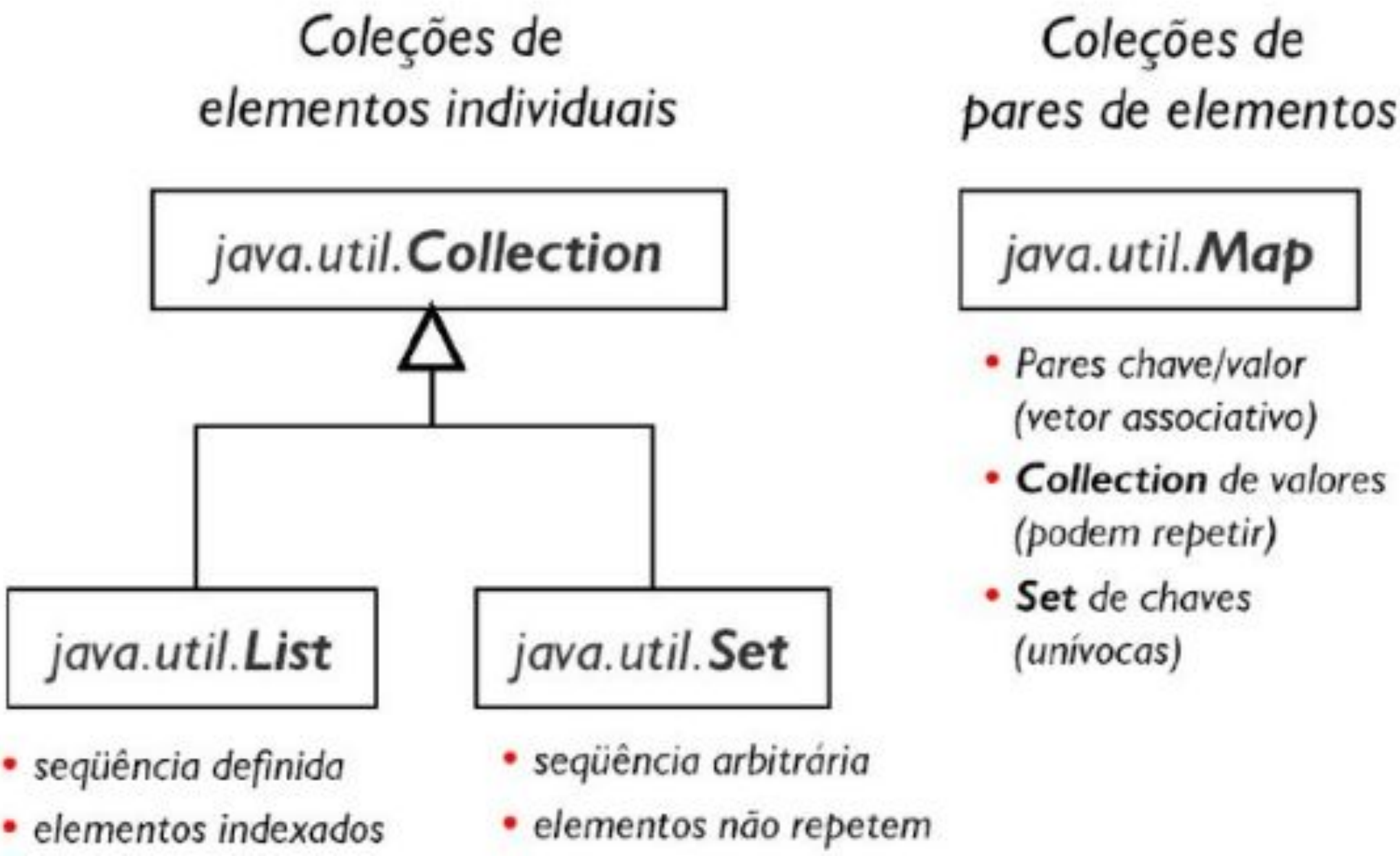


Collections

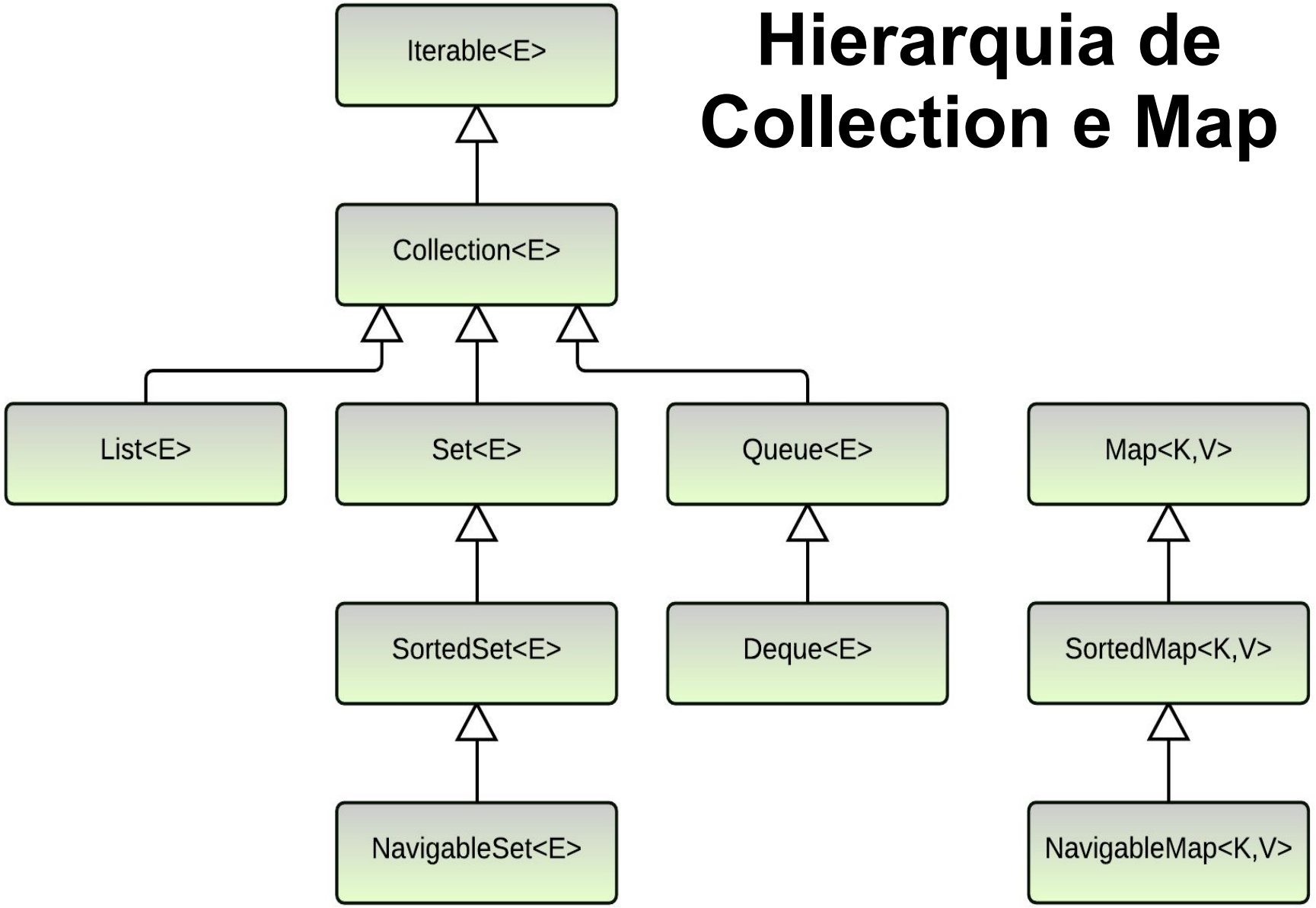
Hierarquia de Coleções



Collection x Map



Hierarquia de Collection e Map



Interfaces:

Classes:

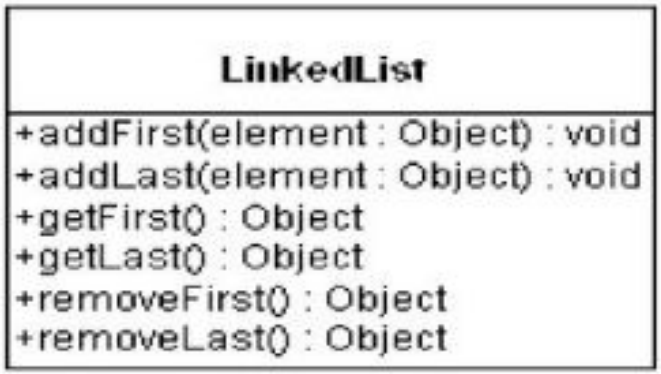
ArrayList LinkedList Vector Stack	HashSet LinkedHashSet TreeSet	PriorityQueue ArrayDeque LinkedList	HashMap HashLinkedMap HashTable TreeMap

Visão Geral das Coleções

- **Collection:**
  - representa um grupo de objetos denominados elementos
  - é apenas uma interface que define métodos para se adicionar, remover e pesquisar em uma estrutura de dados
  - o JDK não provê implementação direta desta Interface.
- **Interface Set:**
  - coleção que não permite elementos duplicados
  - um set não pode conter um par e1, e2, tal que e1.equals(e2)
- **Hashset extends AbstractSet implements Set:**
  - mantém uma coleção de objetos nos quais se pode aplicar operações de intersecção, diferença e iteração
- **TreeSet:**
  - apresenta a característica de armazenar os elementos em ordem crescente

Visão Geral das Coleções

- **List**
  - é uma coleção que traz uma ordem associada aos seus elementos
  - os elementos podem ser acessados pelos seus respectivos índices dentro da lista
  - permitem elementos duplicados
- **ArrayList extends AsbtractList implements List**
  - provê métodos para se manipular o tamanho do array utilizado para armazenar a lista
  - cada instância de ArrayList possui uma capacidade (10 elementos por default)
  - permite todos os elementos (inclusive elementos *null*)
- **LinkedList implements List**
  - provê métodos para se manipular elementos nas extremidades da lista

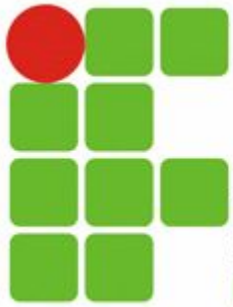


Visão Geral das Coleções

- **Map**
  - permite armazenar pares chave/valor (similar a uma tabela de 2 colunas)
  - dada a chave, permite recuperar o valor associado
  - não permite chaves duplicadas
- **HashMap extends AbstractMap implements Map**
  - permite o armazenamento de *null* para valores e chaves

Visão Geral das Coleções

Interfaces / Classes Abstratas	Classes Concretas
`Collection`	acesso básico e funções de atualização
`Set`	• `HashSet` (conjunto de valores armazenados em uma Hash Table) • `TreeSet` (conjunto ordenado de valores)
`List`	• `ArrayList` (pode substituir a classe `Vector` a partir do JDK 1.2) • `LinkedList`
`Map`	• `HashMap` (pode substituir a classe `Hashtable` a partir do JDK 1.2) • `TreeMap` (um *map* ordenado) • `WeakHashMap` (tabela cujas entradas são eliminadas sempre que a coisa a qual ela se refere é eliminada da memória (através da coleta automática de lixo)
Outros	• `Stack` • `Array` • `BitSet` • `Iterator` (substitui a classe `Enumerator`)



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
FLUMINENSE

Interface Collection (java.util.Collection)

Interface Collection

As coleções têm como base a interface Collection, que define métodos para adicionar e remover um elemento, e verificar se ele está na coleção, entre outras operações, como mostra a tabela a seguir:

`boolean add(Object)`	Adiciona um elemento na coleção. Como algumas coleções não suportam elementos duplicados, este método retorna `true` ou `false` indicando se a adição foi efetuada com sucesso.
`boolean remove(Object)`	Remove determinado elemento da coleção. Se ele não existia, retorna `false`.
`int size()`	Retorna a quantidade de elementos existentes na coleção.
`boolean contains(Object)`	Procura por determinado elemento na coleção, e retorna verdadeiro caso ele exista. Esta comparação é feita baseando-se no método `equals()` do objeto, e não através do operador ==.
`Iterator iterator()`	Retorna um objeto que possibilita percorrer os elementos daquela coleção.

Segue algumas das operações fornecidas pela interface *Collection* e consequentemente presentes nas classes concretas:

- **add**: Adiciona o objeto passado por parâmetro a coleção.
- **addAll**: Adiciona todos os elementos da coleção passada por parâmetro na coleção que chamou o método.
- **remove**: Remove o objeto passado por parâmetro da coleção.
- **removeAll**: Remove todos os elemento da coleção passada por parâmetro da coleção que chamou o método.
- **contains**: Retorna true se o objeto passado por parâmetro estiver dentro da coleção, senão retorna false.
- **isEmpty()**: Retorna true se a coleção não tiver nenhum objeto dentro dela.
- **size()**: Retorna o número de objetos da coleção.
- **clear()**: Remove todos os objetos de dentro da coleção.
- **toArray()**: Retorna um array contendo todos os objetos da coleção.
- **Iterator()**: Retorna um objeto Iterator utilizado para percorrer os objetos da coleção. Este método é especificado na interface **Iterable** a qual **Collection** estende.

Interface Collection

Coleção de Objetos Produto

```
Collection<Produto> colecaox = new HashSet<Produto>();

Produto p1 = new Produto(1012, “Cerveja em lata”);
colecaox.add(p1);

Produto p2 = new Produto(1050, “Biscoito recheado”);
colecaox.add(p2);

colecaox.add(new Produto(2034, “Sabão em pó”));

System.out.println(“Quantidade: ” + colecaox.size() + “ itens”);
```

Interface Collection

Coleção de Objetos Diversos

```
Collection<Object> colecaoy = new ArrayList<Object>();

colecaoy.add("Item 1");
colecaoy.add(56);
colecaoy.add(false);
colecaoy.add(new Date());
colecaoy.add(new Produto(2034, "Sabão em pó"));
colecaoy.add(new Cliente("Manuel", "Rua 15", "4532-7125"));

System.out.println("Quantidade: " + colecaoy.size() + " itens");
```

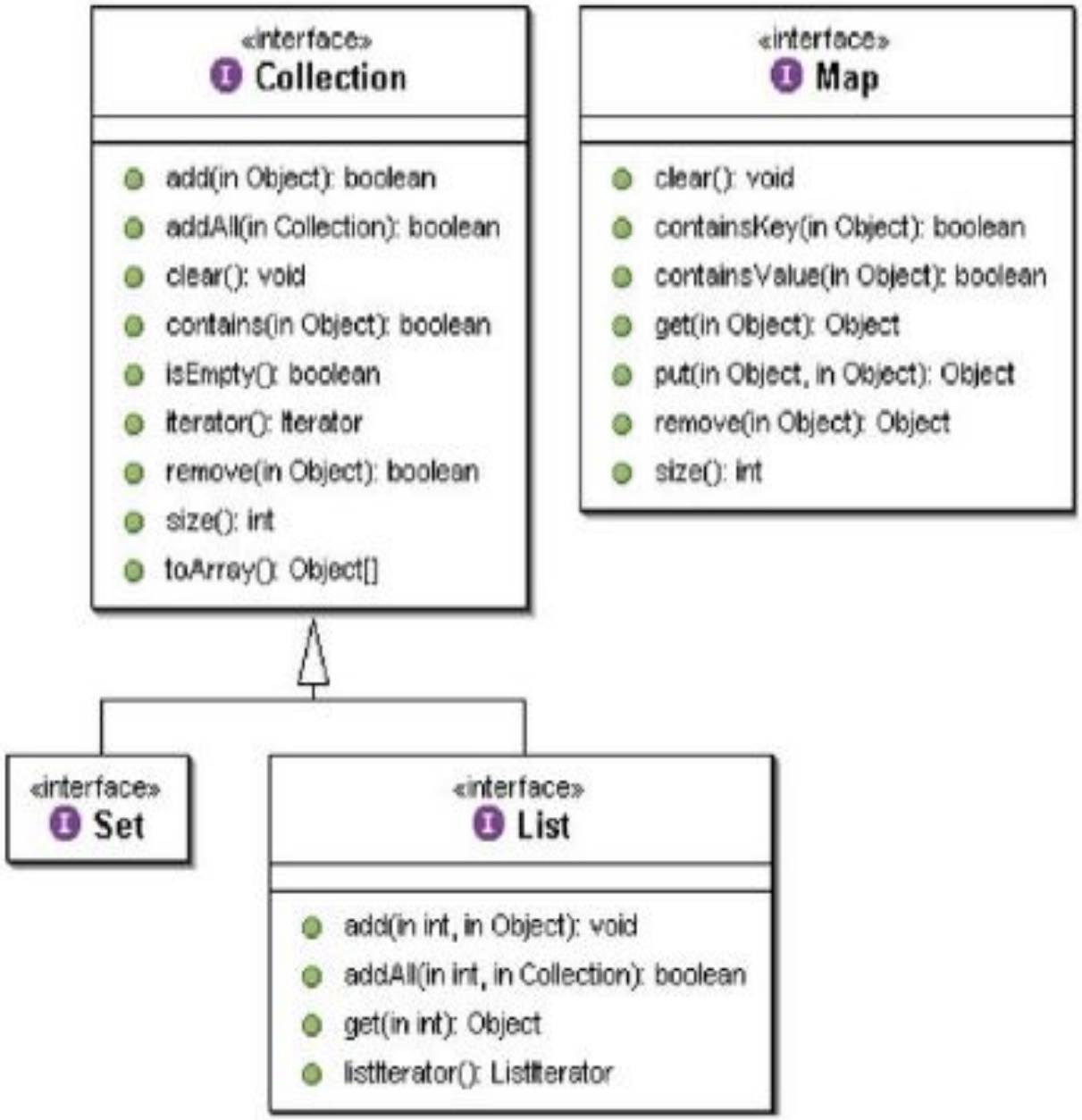
- Percorrendo uma coleção com a interface Iterator

```
Iterator<Funcionario> it = colecao.iterator();

while (it.hasNext()) {
        Funcionario func = it.next();
        System.out.println("Funcionario: " + func.getNome());
}
```

- Percorrendo uma coleção com for-each

```
for (Funcionario func : colecao) {
        System.out.println("Funcionario: " + func.getNome());
}
```



Uma coleção pode implementar diretamente a interface Collection, porém normalmente se usa uma das duas sub interfaces mais famosas: justamente Set e List.

Qual Collection usar?

- A aplicação é quem indica o tipo de coleção que deve ser usada:
  - Se a aplicação requer **duplicatas** use **lista**;
  - Se requer **muita pesquisa** de dados **não use lista**;
  - Se **não requer duplicatas** e **não utiliza chaves**, use **conjunto**;
  - Se **não requer duplicatas e utiliza chaves**, use **mapa**.

Qual Collection usar?

- Se a coleção não apresenta elementos duplicados e se deseja que os elementos estejam em ordem: `TreeSet`
- Se existem entradas duplicadas: qualquer implementação de `List`
- Se a coleção é composta por pares chave/valor: qualquer implementação de `Map`

Leitura Obrigatória

https://www.caelum.com.br/apostila-java-orientacao-objetos/collections-framework/

Material Complementar

http://www.botecodigital.info/java/uma-visao-sobre-o-framework-collections-do-java/

https://slideplayer.com.br/slide/10074022/ - SET e MAP

Material Consultado

- https://www.caelum.com.br/apostila-java-orientacao-objetos/collections-framework/#arrays-so-trabalhosos-utilizar-estrutura-de-dados
- http://www.botecodigital.info/java/uma-visao-sobre-o-framework-collections-do-java/ - MUITO BOM
- https://slideplayer.com.br/slide/16805430/
- https://www.devmedia.com.br/java-collections-como-utilizar-collections/18450
- https://www.devmedia.com.br/classes-stack-queue-e-hashtable-colecoes-estrutura-da-linguagem-parte-3/19256
- https://www.devmedia.com.br/overview-of-java-arraylist-hashtable-hashmap-hashet-linkedlist/30383
- https://slideplayer.com.br/slide/10074022/
- https://www.caelum.com.br/apostila-java-estrutura-dados/vetores/