Estrutura de Dados

Java Generics, Coleções, Árvores e Mapas

Prof. MSc. David Alain do Nascimento

IFPE Campus Garanhuns

Java Generics

```
32          ArrayList<String> listaInteiros = new ArrayList<>();
33          listaInteiros.add("teste");
34          Object o2 = listaInteiros.get(0);
35          String i2 = listaInteiros.get(0);
36
37          ArrayList lista = new ArrayList();
38          lista.add("teste");
39          Object o = lista.get(0);
40          String i = lista.get(0);
41
42          E e = new E();
43          E[] e2 = new E[10];
```

- Representação de objetos como um tipo específico
  - Não é mais tratado como sendo **Object**
  - Não precisa fazer ***cast*** para acessar os atributos e métodos
- O tipo genérico não é instanciável dentro da classe onde se está declarando o tipo

Java Generics

```
32  ArrayList<String> listaInteiros = new ArrayList<>();
33  listaInteiros.add("teste");
34  Object o2 = listaInteiros.get(0);
35  String i2 = listaInteiros.get(0);
36
37  ArrayList lista = new ArrayList();
38  lista.add("teste");
39  Object o = lista.get(0);
40  String i = lista.get(0);
41
42  E e = new E();
43  E[] e2 = new E[10];
```

Jeito apropriado!

Não deve ser utilizado assim

Erro

- Representação de objetos como um tipo específico
  - Não é mais tratado como sendo **Object**
  - Não precisa fazer ***cast*** para acessar os atributos e métodos
- O tipo genérico não é instanciável dentro da classe onde se está declarando o tipo

Java Generics

- Exemplo de uso
  - Um tipo genérico: <E>
    - Repositório
    - DAO (Data Access Object)
  - Mais de um tipo genérico: <E,T> ou <E,T,V> ou outros
    - Dupla, tripla, etc

- Exercício: Implementar um DAO de pessoas
  - Uma pessoa tem um nome, cpf, login e senha.
  - O DAO tem os métodos inserir, atualizar, remover e pegar.
  - As pessoas do DAO (de pessoas) são guardados em um ArrayList.

Comparação de objetos

- Como saber se duas instancias são iguais?

```
String a = "2";
String b = "2";
if(a == b) {
    System.out.println("SIM");
}else {
    System.out.println("NÃO");
}
```

Irá imprimir SIM!

Comparação de objetos

- Como saber se duas instancias são iguais?

```
String a = new String("2");
String b = new String("2");
if(a == b) {
    System.out.println("SIM");
}else {
    System.out.println("NÃO");
}
```

Irá imprimir NÃO!

Comparação de objetos
Método *equals*

- Como saber se duas instancias são iguais?

```
String a = new String("2");
String b = "2";
if(a.equals(b)) {
    System.out.println("SIM");
}else {
    System.out.println("NÃO");
}
```

Irá imprimir SIM!

Comparação de objetos
Método *equals*

- Como saber se duas instancias são iguais?
  - Usando o método equals

- **Todas as classes** devem ter o método equals implementado.
  - Quando não tem implementado na classe, então é utilizado método equals da classe Object.
    - Comparação de objetos com ==
    - Compara se são a mesma instância sem comparar o conteúdo

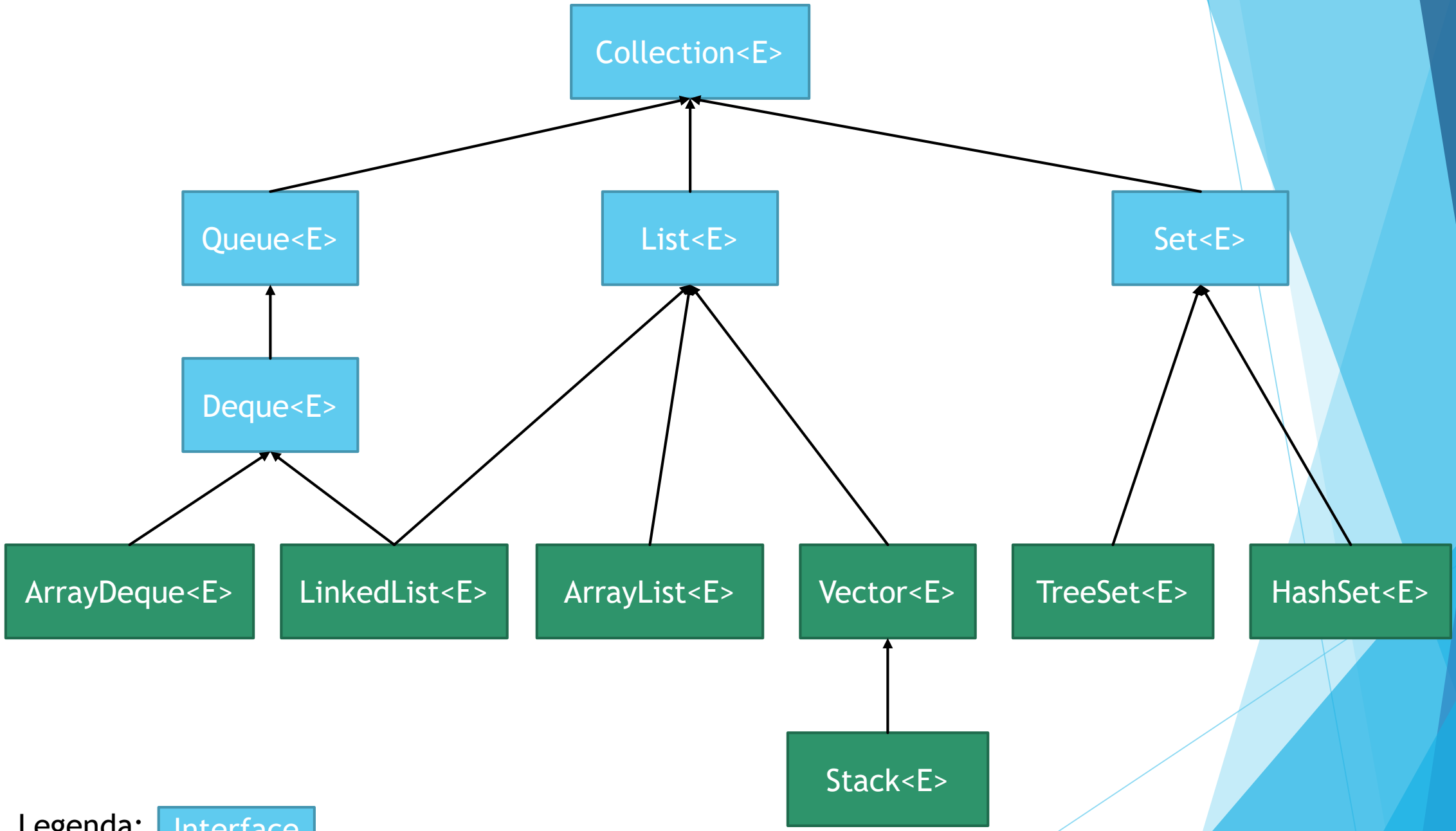
Comparação de objetos
Método *equals*

Exercício

- Implemente a classe Pessoa que contenha os atributos nome, cpf, login e senha e o método equals. Nesta classe, uma pessoa é igual a outra se tiver o mesmo valor de CPF.
- Crie um ArrayList de pessoas e adicione 3 pessoas e depois imprima a lista.
  - {“Zé”, “000.000.000-00”, “ze”, “1234”}
  - {“João”, “001.001.001-01”, “joao”, “0101”}
  - {“Maria”, “002.002.002-02”, “maria”, “2345”}
- Remova Maria da lista, **sem utilizar o índice** e imprima a lista novamente.

Coleções

- **Collection**
  - **List** (lista)
    - **ArrayList** (lista implementada com array)
    - **LinkedList** (lista encadeada)
  - **Queue** (fila)
    - **Deque** [**D**ouble **E**nded **Que**ue] (fila com duas caudas)
      - **ArrayDeque** (fila com duas caudas implementada com array)
      - **LinkedList** (fila com duas caudas implementada com lista)
  - **Set** (conjunto, **não pode ter elementos repetidos**)
    - **TreeSet** (árvore de conjuntos, **os elementos precisam ser comparáveis**)
    - **HashSet** (tabela hash de conjuntos, **os elementos precisam ter o método hashCode**)


Collection<E>
Queue<E>
List<E>
Set<E>
Deque<E>
ArrayDeque<E>
LinkedList<E>
ArrayList<E>
Vector<E>
TreeSet<E>
HashSet<E>
Stack<E>
Legenda:
Interface
Classe

Coleções
Principais métodos

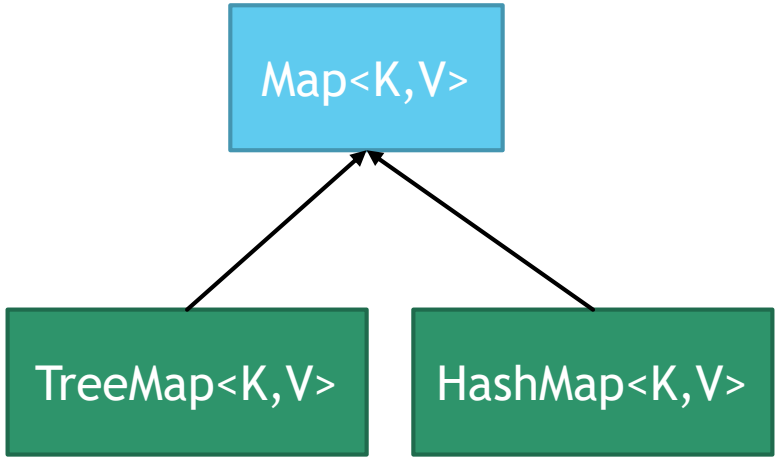
- **Collection<E>**
  - int size(); //quantidade de elementos
  - boolean isEmpty(); //verifica se está vazia
  - boolean contains(Object o); //verifica se o objeto ***o*** está contido
  - boolean add(E e); //adiciona um elemento ***e***
  - boolean remove(Object o); //remove o objeto ***o***

Coleções
Principais métodos

- **List<E>**
  - int size(); //quantidade de elementos
  - boolean isEmpty(); //verifica se está vazia
  - boolean contains(Object o); //verifica se o objeto ***o*** está contido
  - boolean add(E e); //adiciona um elemento ***e***
  - boolean remove(Object o); //remove o objeto ***o***
  - void add(int index, E element); //adiciona ***element*** no índice ***index***
  - E get(int index); //pegar o elemento que está no índice ***index***
  - E set(int index, E element); //trocar o elemento do índice ***index*** por ***element***
  - E remove(int index); //remove o elemento que está no índice ***index***

Mapas

- Map (mapa, relaciona um objeto **chave** com um objeto **valor**)
  - HashMap (cria o mapa através da hash da chave)
  - TreeMap (cria o mapa através de uma árvore)



Legenda:

Interface

Classe

Mapas
Principais métodos

- **Map<K,V>**
  - int size(); //retorna a quantidade de pares <chave,valor> estão armazenados
  - boolean isEmpty(); //verifica se está vazio
  - boolean containsKey(Object key); //verifica se tem uma determinada chave ***key***
  - boolean containsValue(Object value); //verifica se tem tem um determinado valor ***value***
  - V get(Object key); //pega o valor dada a chave ***key*** associada àquele valor
  - V put(K key, V value); //associa a chave ***key*** com o valor ***value*** ou substitui o valor antigo
  - V remove(Object key); //remove o par <chave,valor> dada a chave ***key***
  - Set<K> keySet(); //retorna o conjunto de chaves do mapa
  - Collection<V> values(); //retorna a coleção de valores do mapa
  - Set<Map.Entry<K, V>> entrySet(); //retorna o conjunto com todos os pares <chave,valor> armazenados