

07-Coleções-Parte-1

Conteúdo: Coleções em Java. Introdução ao Generics.

Coleções em Java (1)

- Array x Coleções
 - Array: tamanho estático, manipulação elemento-a-elemento, por índice.
 - Coleções: tamanho dinâmico, diferentes maneiras de manipular os elementos
- Coleção: objeto que agrupa múltiplos objetos-elemento dentro de si.
- Usados para armazenar, recuperar e manipular dados, bem como transmití-los entre objetos
- Java suporta coleções de Object (pré-generics, pré-java 5) e de um tipo especializado (usando Generics).

Equals de Object (1)

- boolean equals(Object o): define se dois objetos são equivalentes em significado.

Exemplo:

```
Integer i1=new Integer(7); Integer i2=new Integer(7); String s="";  
if(i1==i2) s += "i1==i2 sao iguais! "; else s += "i1!=i2! ";  
if(i1.equals(i2)) s += "i1 equals i2! "; else s += "i1 not equals i2! ";
```

- Quando se cria uma classe, deve-se definir o que será determinante na equivalência (sobreposição do equals), caso contrário objetos diferentes serão sempre considerados diferentes. Exemplo: classe Pessoa.

p1:Pessoa	p2:Pessoa
-nome: String="Ana"	-nome:String="Ana"
-CPF : long=123456789	-CPF : long=123456789

Equals de Object (2)

- Quando se deve sobrepor o equals?
 - quando o objeto é usado como chave de busca numa das coleções (Sets)
 - Exemplo na classe Pessoa

Cuidado com as regras para sobreposição.

```
public boolean equals(Object o) {  
    if ( (o instanceof Pessoa) &&  
        ((Pessoa)o).getCPF() == this.getCPF() ) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

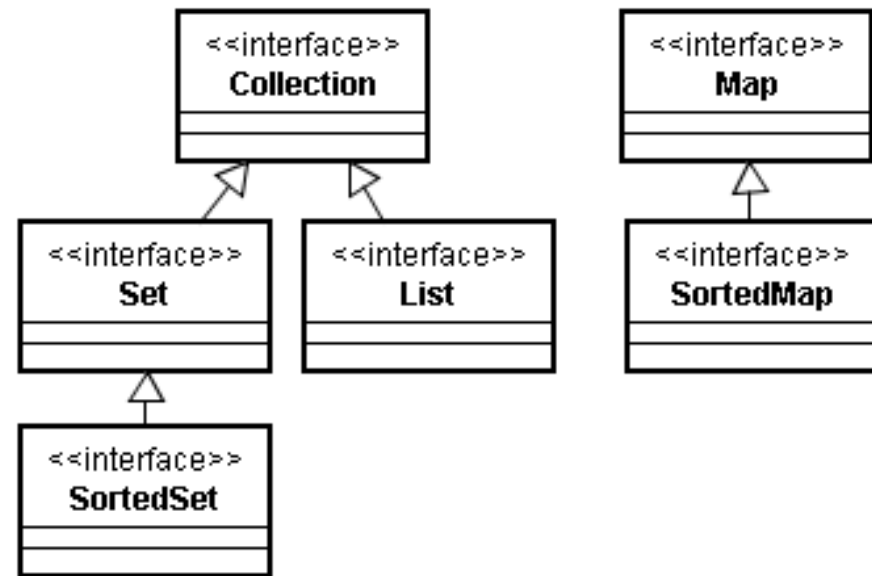
Este código está correto?

Coleções em Java (2)

- Framework de Coleções Java
 - Biblioteca Java de classes e interfaces reutilizáveis para manipulação de coleções (em `java.util`)
 - Reduz o esforço de programação, fornecendo qualidade, velocidade e interoperabilidade
- As coleções só armazenam tipos de referência, não os primitivos. Alternativa: os *wrappers*.
- Interfaces com duas hierarquias distintas: `Collection` e `Map`

Coleções em Java (3)

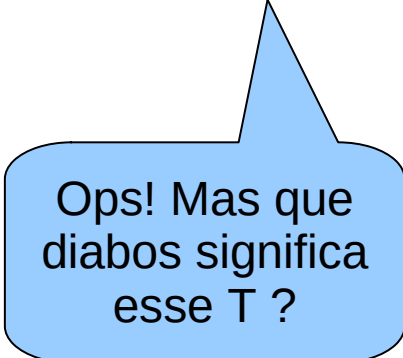
Interfaces do Framework:



- Baseado em hierarquia de interfaces
- Collection: guarda e manipulação de elementos
- Map: mapeia chaves para valores (pares “key=value”). A chave não pode ser duplicada e só mapeia para um único valor

Coleções em Java (4)

- Interface Collection (principais métodos)
 - Métodos: `int size()`, `boolean isEmpty()`, `boolean contains(T)`, `boolean add(T)`, `boolean remove(T)`, `Iterator iterator()`, `boolean containsAll(Collection<T>)`, `boolean addAll(Collection<T>)`, `boolean removeAll(Collection<T>)`, `clear()`, `T[] toArray()`,
 - Interface Iterator: `boolean hasNext()`, `T next()`, `remove()`



Ops! Mas que diabos significa esse T ?

Uma breve introdução ao Generics (1)

- Denominação em Java para uma classe que pode receber um ou mais tipos parametrizados
- Com o tipo específico, posso garantir que somente objetos que passem no teste do É-UM possam ser manipulados pela classe, evitando erros em tempo de execução.
 - Ou você quer que um Gato possa entrar na sua lista de Cachorros?
- 99% do tempo usamos Generics apenas nas classes de coleção do Java

Uma breve introdução ao Generics (2)

- Classe com tipo genérico (JavaDocs)

E é um marcador, indicando que aqui deve ser informado um tipo.

- `public class ArrayList<E>`
- métodos (get, add, etc.) usam esse mesmo E!

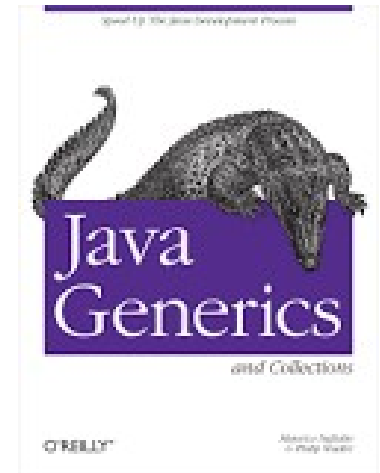
- Criando um objeto de uma classe Generics

- `List<Gato> lista = new ArrayList<Gato>();`
- `lista.add(new Gato()); // Ok.`
- `lista.add(new Cachorro()); // Erro`
- `Gato g=lista.get(0); // Ok.`
- `Cachorro c=lista.get(1); // Erro, retorna um Gato.`

ArrayList é a classe base (generics) e Gato é o tipo parametrizado.

Uma breve introdução ao Generics (3)

- Existe muito mais por trás do Generics
 - Não há polimorfismo para o tipo parametrizado, só para o tipo base
 - `List<Animal> lista=new ArrayList<Gato>(); // Erro`
 - Um método por usar wildcard
 - `public void metodo(List<E extends Animal>)`
 - pode receber qualquer subtipo de Animal, por exemplo um `List<Gato>` ou um `List<Cachorro>`, mas só para leitura.
 - Método genérico, etc...



Robin: "Santa Complexidade, Batman! esse livro tem 286 páginas!")

Coleções em Java (5)

- Coleção List

- objetos-elemento armazenados de forma ordenada

- Implementações

- ArrayList

- LinkedList

- Exemplo:

```
List<Integer> al=new ArrayList<Integer>();
```

```
al.add(10); al.add(30); al.add(20);
```

```
for(Integer item:al) System.out.print(item+"-");
```

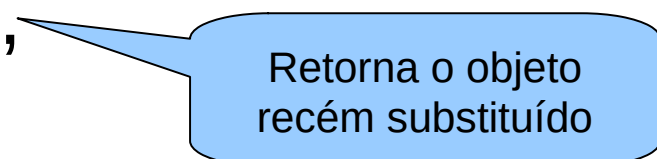
```
System.out.print(al.get(1));
```



O que produz?

Coleções em Java (6)

- Interface List: Collection indexada, que permite elementos duplicados
- Permite acessar os elementos de forma indexada: `T get(int)`, `T set(int,T)`,
`void add(int,T)`, `T remove(int)`,
`boolean addAll(int,Collection<T>)`,
`int indexOf(T)`, `int lastIndexOf(T)`,
`ListIterator listIterator()`,
`ListIterator listIterator(int)`, `List subList(int,int)`



Retorna o objeto recém substituído

Exercícios de Fixação

- 07-Coleções-Parte-1

