

POO - PROGRAMAÇÃO ORIENTADA A OBJETOS

Coleções de Objetos

Rodrigo R Silva

Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense
Campus Bagé

Nesta Aula Veremos...

1 Coleções de Objetos

Coleções de Objetos

Coleções de Objetos

- Na interface Collection estão definidos métodos básicos que todas as coleções terão.
 - boolean add(Object o)
 - boolean addAll(Collection c)
 - void clear()
 - boolean contains(Object o)
 - boolean containsAll(Collection c)
 - boolean equals(Object o)
 - boolean isEmpty()
 - boolean remove(Object o)
 - boolean removeAll(Collection c)
 - boolean retainAll(Collection c)
 - int size()
 - Object[] toArray()

A Interface List

- A interface List é uma especialização de Collection.
- Uma List declara o comportamento de um tipo de coleção que armazena uma sequência de objetos.
- Os elementos podem ser inseridos ou acessado através de uma posição (índice) na lista.
- Os índices permite acesso aleatório aos elementos da lista.
- Listas podem conter elementos duplicados.

Métodos de List

■ List herda os métodos de Collection e adiciona outros:

- void add(int indice, Object o)
- boolean addAll(int indice, Collection c)
- Object get(int indice)
- int indexOf(Object o)
- int lastIndexOf(Object o)
- Object remove(int indice)
- Object set(int indice, Object o)
- List subList(int inicio, int fim)

Classes de coleções

- As interfaces de coleções são implementadas por várias classes.
- ArrayList é uma classe para implementação de listas a qual dá suporte a criação de Arrays dinâmicos.
- ArrayList é uma especialização de AbstractList e implementa a interface List.
- Diferente de Arrays, estruturas do tipo ArrayList não tem um número de elementos fixos.
- O número de elementos aumenta e diminui conforme a execução de instruções que adicionam ou removem elementos.

Construtores de ArrayList

- A classe ArrayList possui três sobrecargas de construtores:
- ArrayList(): cria uma lista vazia;
- ArrayList(Collection c): cria uma lista já inicializada;
- ArrayList(int capacidadeInicial): cria uma lista já dimensionada.
- Em todas as formas serão criadas estruturas de listas dinâmicas, cujo número de elementos é variável.

ArrayList primeiro exemplo

```
package classe.executavel;

import java.util.ArrayList;
import java.util.List;

public class ClasseExecutavelLista {

    public static void main(String[] args) {

        List lista = new ArrayList();

        for(int i = 1; i <= 10; i++) {
            lista.add(i);
        }

        for(Object j: lista) {
            System.out.println(j);
        }

    }
}
```

ArrayList segundo exemplo

```
package classe.executavel;

import java.util.ArrayList;
import java.util.List;

public class ClasseExecutavelLista {

    public static void main(String[] args) {

        List lista = new ArrayList();

        List lista2;

        for(int i = 1; i <= 10; i++) {
            lista.add(i);
        }

        lista2 = new ArrayList(lista);

        for(Object j: lista2) {
            System.out.println(j);
        }
    }
}
```

ArrayList terceiro exemplo

```
public class ClasseExecutavelLista {  
  
    public static void main(String[] args) {  
  
        List lista = new ArrayList();  
        List lista2, lista3 = new ArrayList(5);  
  
        for(int i = 1; i <= 10; i++) {  
            lista.add(i);  
        }  
  
        lista2 = new ArrayList(lista);  
        lista3.add(lista);  
        lista3.addAll(lista2);  
  
        for(Object j: lista3) {  
            System.out.println(j);  
        }  
  
        System.out.println("-----");  
        System.out.println(lista3.size());  
    }  
}
```

Lista tipadas

- No momento da criação de uma lista podemos identificar o tipo de dados que será inserido.
- Esta listas são denominadas tipadas.
- Ex:
`List<ContaComum> contas;`
`contas = new ArrayList<>();`
- Note que o tipo dos objetos que serão armazenados na lista está identificado entre `<>`.
- No momento da instanciação da lista não precisamos identificar o tipo.

Exemplo

```
public class ClasseExecutavelLista {  
    public static void main(String[] args) {  
        Pessoa correntista = new Pessoa("João", 22233344455l, 22, 999887766, "joao@gmail.com");  
  
        List<ContaComum> contas = new ArrayList<ContaComum>();  
  
        for(int i = 1; i <= 10; i++) {  
            ContaComum contaComum = new ContaComum();  
            contaComum.setCodigo(100+i);  
            contaComum.setCorrentista(correntista);  
            contaComum.setSaldo(500 * i);  
            contas.add(contaComum);  
  
            //contas.add(new ContaComum(100+i, correntista, 500*i));  
        }  
  
        for(ContaComum conta : contas) {  
            System.out.println(conta.toString());  
        }  
    }  
}
```

OBRIGADO!