Orientação a Objetos II

ArrayList

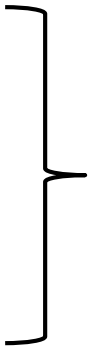
Prof. Felippe Scheidt

2015

Instituto Federal do Paraná

Introdução

- Na modelagem Orientada a Objetos existem cenários onde objetos possuem associações que variam de de 1 para 1 até n para n.
- Existem 3 tipos de relacionamentos entre objetos:
  - associação
  - agregação
  - composição



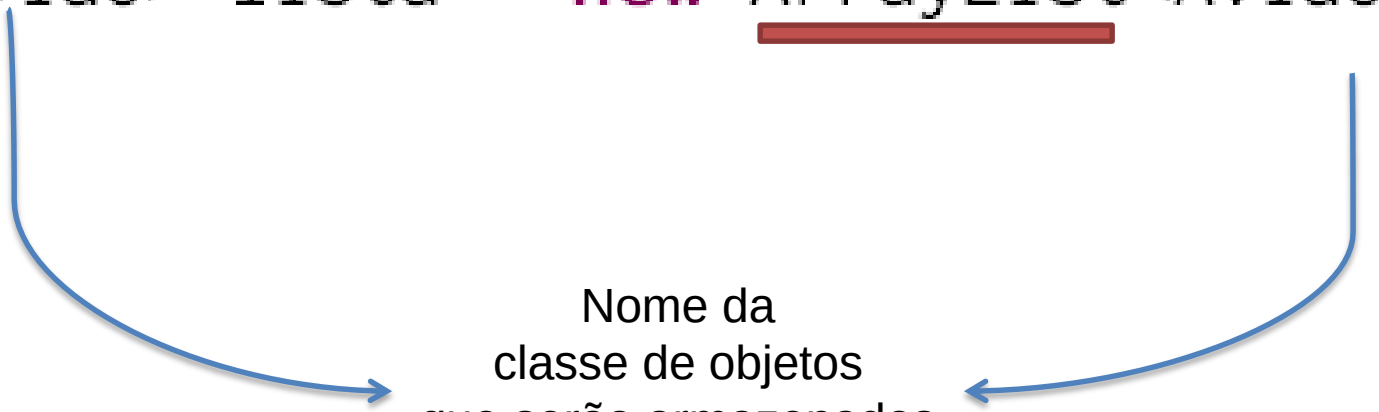
estes tipos de relacionamentos são representados por listas

O que é uma lista?

- Uma lista é uma coleção ordenada de elementos do mesmo tipo.
- Os elementos de uma lista podem ser acessados pela sua posição, isto é, seu índice.
- OBS.: o primeiro elemento de qualquer lista tem índice 0 (zero).

Criando uma lista

```
List<Aviao> lista = new ArrayList<Aviao>();
```



Principais métodos

```
Aviao jatinho = new Aviao("Jatinho","R2012");
lista.add(jatinho);
```

- Método add() permite adicionar um objeto a lista.

Principais métodos

```
int tamanho = lista.size();
```

- Método size() retorna o tamanho da lista.

Principais métodos

```
Aviao jato = lista.get(0);
```

- Método get() retorna um objeto da lista, sem removê-lo. No exemplo acima, retorna o objeto que está na posição `0`

Principais métodos

```
Aviao jumbo = new Aviao("Jumbo","J949");
lista.add(0,jumbo);
```

- Método add(i, obj) especifica que o objeto jumbo deve ser inserido na posição `0`.

Principais métodos

```
lista.remove(0);
```

- Método remove() retira o elemento que está na posição `0` da lista.

Principais métodos

```
lista.set(5, jumbo);
```

- Método set() insere um objeto numa determinada posição, substituindo o objeto que estiver nesta posição.

Principais métodos

```
lista.clear();
```

- Método clear() remove todos os elementos da lista. Limpa a lista.

Percorrendo uma lista (1)

- Percorrendo a lista e imprimindo cada um dos elementos.

```
for(int k=0;k<lista.size();k++){
    System.out.println(lista.get(k));
}
```

Percorrendo uma lista (2)

- Forma simplificada de percorrer uma lista e imprimir seus elementos.

```
for (Aviao obj : lista) {
    System.out.println(obj);
}
```

Principais métodos

- Usando o método addAll() para anexar uma lista a outra lista.

```
List<Aeronave> listaAngar = new ArrayList<Aeronave>();
listaAngar.add(new Aeronave("TAM","DX1981"));

lista.addAll(listaAngar);
```

Principais métodos

```
if(lista.contains(dx1))
    System.out.println("Objeto encontrado");
```

- O método contains() verifica se um objeto está contido na lista.

Exercício 1

- Crie as seguintes classes: Filme e Artista
- Crie a seguinte relação
  - 1 Filme possui vários Artistas
- Para todas as classes, definir:
  - 4 atributos (características)
  - Getters e Setters
  - Compile e execute o arquivo, instanciando 1 filme e 4 artistas
- Implemente um método na classe Filme que calcula o orçamento do filme ex.: `calculaCusto()`
- Dica: adicione o atributo **int estrela** na classe **Artista**. Cada estrela custa 100.000 mil reais de custo. Logo um artista 5 estrelas custa 500mil reais.
- Contabilize o número total de artistas e de estrelas para ter o custo total do filme.

Exercício 2

- Criar a classe **Aviao** (nome, qtdPassgeiro) e a classe **Passageiro** (nome, rg)
- Instancie 5 passageiros.
- Faça um método **embarcarPassageiro()** que adiciona os passageiros ao avião. Este método recebe **um** passageiro e adiciona a Lista de passageiros.
- Um método `desembarcarPassageiros()` quando chamado deve limpar a lista de passageiros (remover todos os passageiros).
- Faça um método que `verifica()` se uma pessoa já embarcou no avião ou não.
- Remove da aeronave qualquer passageiro menor que 10 anos (dica: adicione o atributo idade na classe passageiro).
- Adicione uma "**Regra de Negócio**" que não permite o embarque de passageiros caso o avião já tenha atingido a capacidade máxima.