



SÃO
PAULO
TECH
SCHOOL

Linguagem de Programação

List e ArrayList Relacionamento

Profa. Célia Taniwaki

Prof. Diego Brito

Profa. Giuliana Miniguiti

Coleção ArrayList

- A Java API fornece várias estruturas de dados predefinidas, chamadas **coleções**, utilizadas para armazenar grupos de objetos relacionados.
- Fornecem métodos eficientes que organizam, armazenam e recuperam seus dados sem que seja necessário conhecer como os dados são armazenados.
- Reduz o tempo de desenvolvimento de aplicativos.

Coleção ArrayList

- Os arrays (vetores) não podem ter seu tamanho alterado dinamicamente para acomodar mais elementos.
- **ArrayList <T>** (pacote java.util) é uma das coleções do Java, que equivale a um vetor, mas pode ter seu tamanho alterado dinamicamente para acomodar mais elementos.
 - **T** é um espaço reservador para o tipo de elemento armazenado na coleção.
 - Isso é semelhante a especificar o tipo ao declarar um vetor, exceto que apenas tipos **não primitivos** podem ser utilizados com essas classes de coleção.
 - Classes com essa espécie de marcador que podem ser utilizadas com qualquer tipo são chamadas **classes genéricas**.

ArrayList - Métodos

Método	Descrição
<code>add</code>	Adiciona um elemento ao fim de <code>ArrayList</code> .
<code>clear</code>	Remove todos os elementos de <code>ArrayList</code> .
<code>contains</code>	Retorna <code>true</code> se <code>ArrayList</code> contiver o elemento especificado; do contrário, retorna <code>false</code> .
<code>get</code>	Retorna o elemento no índice especificado.
<code>indexOf</code>	Retorna o índice da primeira ocorrência do elemento especificado em <code>ArrayList</code> .
<code>remove</code>	Remove a primeira ocorrência do valor especificado.
<code>remove</code>	Remove o elemento no índice especificado.
<code>size</code>	Retorna o número de elementos armazenados no <code>ArrayList</code> .
<code>trimToSize</code>	Reduz a capacidade de <code>ArrayList</code> de acordo com o número de elementos atual.

Figura 7.23 | Alguns métodos e propriedades da classe `ArrayList<T>`.

Instrução for aprimorada (for enhanced)

- Acessa os elementos de um vetor ou List sem utilizar um contador
- Evita a possibilidade de ultrapassar o limite do vetor ou List
- Funciona para vetores e coleções predefinidas do Java

A instrução **for aprimorado** simplifica o código para percorrer um vetor ou uma coleção


Sintaxe:

```
for ( parâmetro : nomeDoArray ) {  
    // instruções  
}
```

- **parâmetro**: tem um tipo e uma variável
- **nomeDoArray**: é o array pelo qual iterar.
- O tipo do parâmetro deve ser consistente com o tipo de elemento no array.

Exemplo de uso ArrayList

```
public class ArrayListExemplo {  
    public static void main(String args[])  
    {  
        // cria um novo ArrayList de strings  
        List<String> cores= new ArrayList<String>();  
  
        cores.add("vermelho"); //acrescenta "vermelho" à lista  
        cores.add(0,"amarelo"); //insere "amarelo" no índice 0  
  
        for (int i=0; i < cores.size(); i++){ //exibe cores usando o for tradicional  
            System.out.printf(" %s", cores.get(i));  
        }  
  
        for (String cor: cores){ //exibe cores usando o for enhanced  
            System.out.printf(" %s", cor);  
        }  
    }  
}
```



Cria um ArrayList de Strings

Relacionamento entre classes

Sistema Orientado a Objetos é formado por vários objetos que se **relacionam entre si**:

Relacionamento entre classes – há 3 tipos:

- **Associação**
 - Exemplo: classe Funcionário – classe Empresa
- **Composição e Agregação**
 - Um objeto é parte de outro objeto
 - Exemplo: botão é um objeto que faz parte do objeto Tela da classe JFrame
- **Herança**
 - Cria-se uma classe como herdeira de outra já existente

Associação

A Associação indica que as classes se relacionam de alguma forma (seus objetos se relacionam em tempo de execução)

No Diagrama de Classes da UML, a Associação é indicada por uma linha que une as 2 as classes

Exemplo: Classe Cliente e Classe Pedido – Um cliente faz 0 ou mais pedidos. Os números nos extremos da linha indicam a multiplicidade do relacionamento



Composição / Agregação

Indicam uma relação objeto todo e objeto parte.

Um objeto contém um ou mais objetos de outra classe

Diferença entre **Composição** e **Agregação**:

- **Composição**
 - Objeto parte só existe se o objeto todo existe
 - Objeto parte não pode ser parte de outro objeto
 - Ex: Pedido e ItemPedido
- **Agregação**
 - Objeto parte existe independente do objeto todo
 - Objeto parte pode eventualmente ser parte de outro objeto
 - Ex: Equipe e Jogador

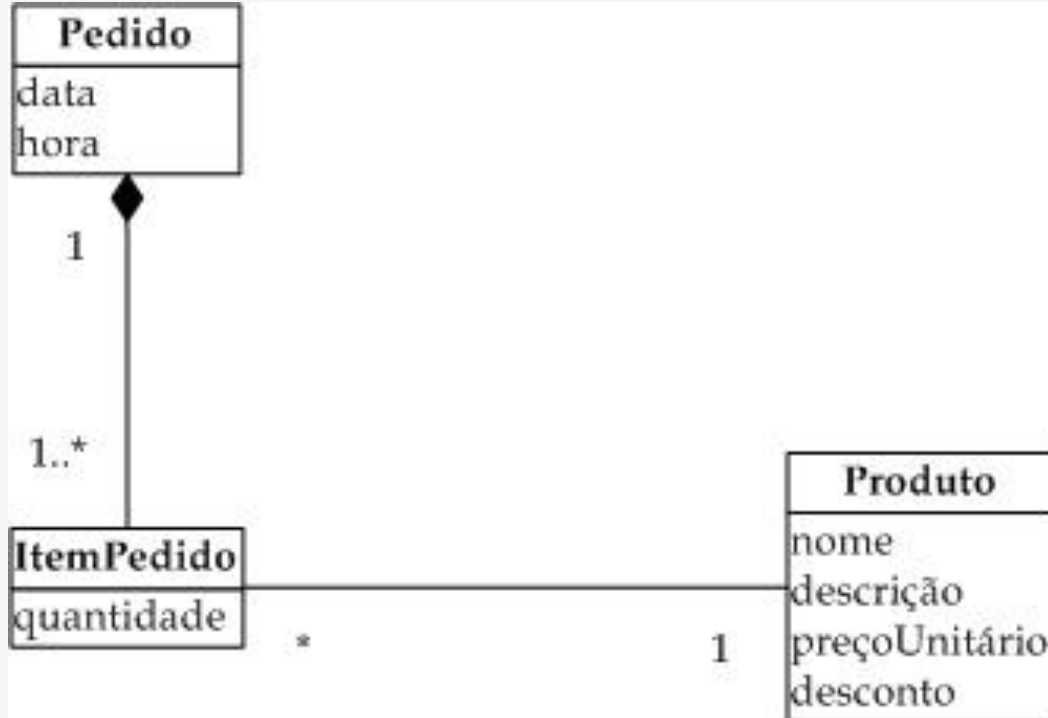
Relacionamento do tipo **"TEM-UM"**

- Ex: Pedido **"tem um"** ItemPedido
Equipe **"tem um"** Jogador

Composição / Agregação

No Diagrama de Classes da UML:

- Ambos são indicados por **um losango ao lado da classe todo**, com uma linha ligando a classe todo à classe parte, porém:
 - Composição – losango escuro
 - Agregação – losango claro



Objeto como atributo

Para implementar a composição ou agregação de classes, um objeto (todo) contém objetos de outras classes (partes).

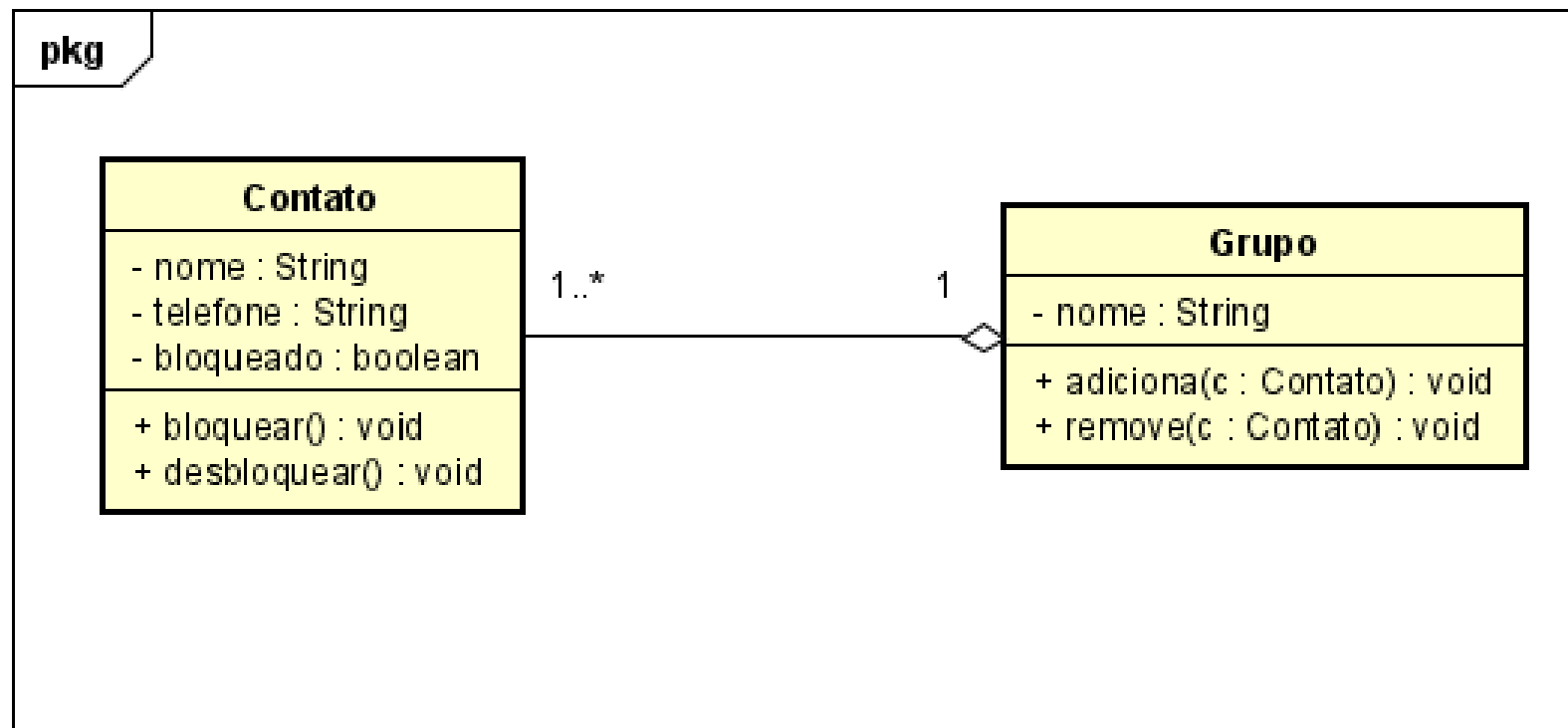
Por exemplo:

- Classe Disciplina:
 - Atributos:
 - String sigla;
 - int codigo;
 - Professor prof;
 - String descricao
 - Nesse exemplo, prof é um objeto da classe Professor
 - Na criação de um objeto Disciplina, pode-se ter:

Exemplo de agregação de classes

Classe **Grupo** :

- Representa uma agregação de **Contato**.
- Um objeto **Grupo** contém **1 ou mais** objetos **Contato**.
- Por isso a classe Grupo tem um atributo que é um **List** de Contatos (mas não aparece no diagrama).



Agradeço
a sua atenção!

Obrigada!

giuliana.franca@sptech.school
diego.lima@sptech.school

SÃO
PAULO
TECH
SCHOOL