Aula 9

C.E.S.A.R school

NExT

Nova Experiência de Trabalho

Novas oportunidades, novos desafios

Ao vivo | Nova edição

APOIO:

PortoDigital
Onde as conexões acontecem

SOFTEX RECIFE

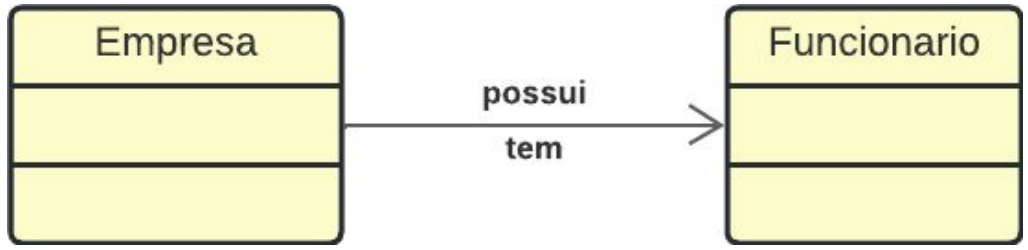
O QUE É ASSOCIAÇÃO EM JAVA

COLABORAÇÃO, COMPOSIÇÃO E AGREGAÇÃO

A Associação é a relação entre duas classes distintas que se estabelecem por meio de seus objetos. A associação pode ser:

- um para um
- um para muitos
- muitos para um
- muitos para muitos.

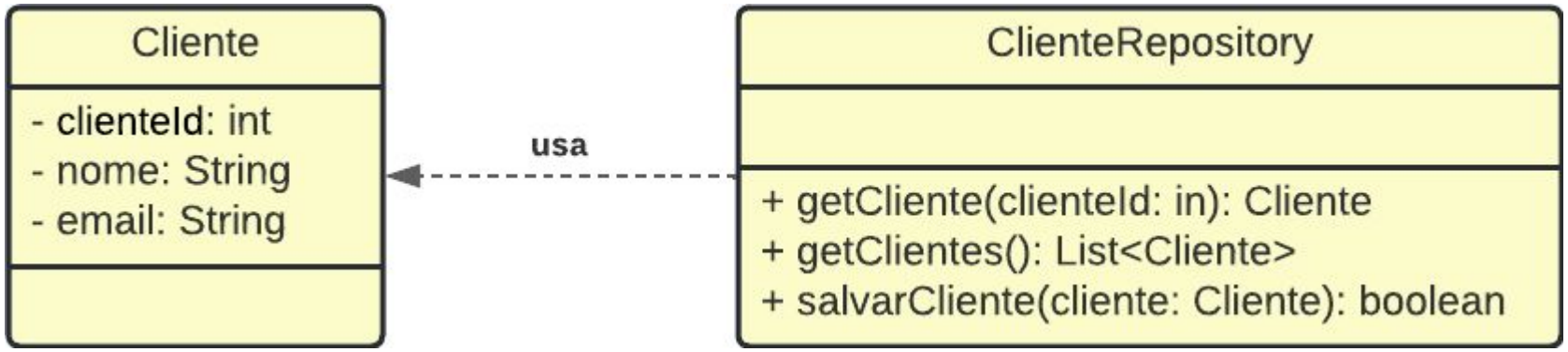
Na programação orientada a objetos, um objeto se comunica com outro objeto para usar a funcionalidade e os serviços fornecidos por esse objeto. Colaboração, Composição e agregação são as três formas de associação.



COLABORAÇÃO

COLABORAÇÃO

- O relacionamento de colaboração é muitas vezes conhecido como relacionamento 'usa um' ou 'uses a'. A colaboração declara que dois objetos estão colaborando quando um objeto faz uso de outro objeto para completar uma operação.
- Em nossa exemplo, para salvar e recuperar os detalhes do clientes, a classe ClienteRepository usa um objeto Cliente para salvar e recuperar os dados. Da mesma forma outras classes de repositório como ProdutoRepository e PedidoRepository usam objetos Produto e Pedido respectivamente, e, portanto, estão colaborando para executar uma operação.



COLABORAÇÃO

```java
public class Cliente {
    public int clienteId;
    public String nome;
    public String email;
    public Cliente(int clienteId) {
        this.clienteId = clienteId;
    }
}
```

usa

```java
import java.util.ArrayList;
import java.util.List;
public class ClienteRepository {
    public Cliente getCliente(int clienteId) {
        // cliente uma instância da classe Cliente
        Cliente cliente = new Cliente(clienteId);
        // retorna um cliente
        return cliente;
    }
    public List<Cliente> getClientes() {
        // retorna todos os clientes
        return new ArrayList<Cliente>();
    }
    public boolean salvarCliente(Cliente cliente) {
        // salva o cliente definido
        return true;
    }
}
```

AGREGAÇÃO

AGREGAÇÃO

- Busca demonstrar que as informações de um objeto precisam ser complementadas pelas informações contidas em objetos de outra classe.
- Representam uma relação todo-parte.
    - A agregação indica que uma das classes do relacionamento é uma parte, ou está contida em outra classe
- É representada por uma seta com um losango vazado na ponta que deve ficar ao lado do objeto-todo.

AGREGAÇÃO

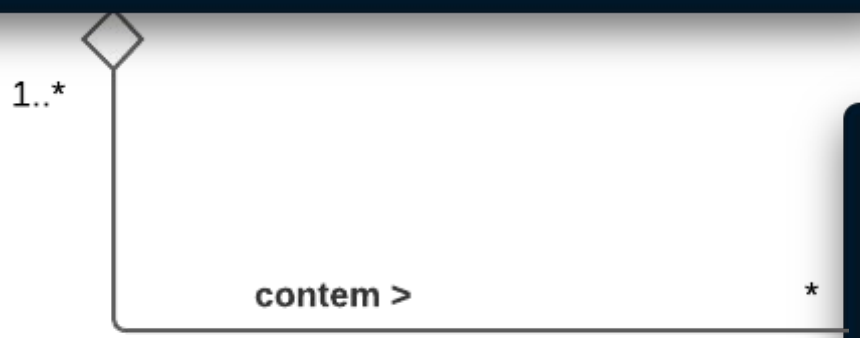
Objetos da classe Time são objetos-todo que precisam ter suas informações complementadas pelos objetos da classe Pessoa que são objetos-parte

AGREGAÇÃO

Dessa forma, sempre que um Time for consultado, além de suas informações pessoais, serão apresentadas todas as Pessoas que o constitui.

AGREGAÇÃO

```java
import java.util.List;
public class Time {
    public String nome;
    public List<Jogador> jogadores;
    public Time(String nome, List<Jogador> jogadores) {
        this.nome = nome;
        this.jogadores = jogadores;
    }
}
```




```java
public class Jogador {
    public String nome;
}
```

AGREGAÇÃO

- Sua função principal é identificar a obrigatoriedade de uma complementação das informações de um objeto-todo por seus objetos-parte, quando este for consultado.

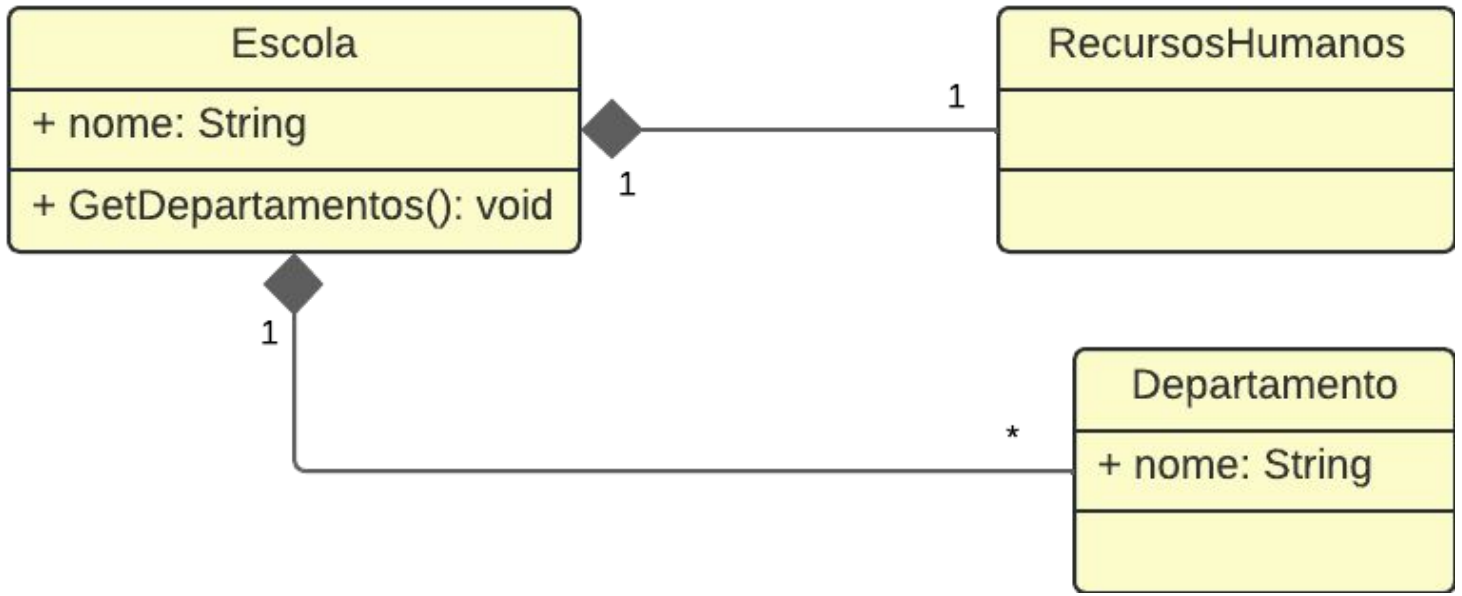
- É uma associação unidirecional, ou seja, um relacionamento de mão única. Por exemplo, o time pode ter pessoas, mas vice-versa não é possível e, portanto, de natureza unidirecional.

- Na agregação, ambas as entradas podem sobreviver individualmente, o que significa que terminar uma entidade não afetará a outra entidade

COMPOSIÇÃO

COMPOSIÇÃO

- A Composição, representa um vínculo forte entre duas classes , e , é também um relacionamento caracterizado como parte/todo, mas, neste caso, o todo é responsável pelo ciclo de vida da parte. Assim a existência do Objeto-Parte NÃO faz sentido se o Objeto-Todo não existir.



COMPOSIÇÃO

```java
import java.util.ArrayList;
import java.util.List;
public class Escola {
    public String nome;
    public RecursosHumanos rh;
    public List<Departamento> listaDepartamentos;
    public Escola(String nome, String nomeRH, List<String> departamentos) {
        this.nome = nome;
        this.listaDepartamentos = new ArrayList<Departamento>();
        this.rh = new RecursosHumanos(nomeRH);
        for (String nomeDepartamento : departamentos) {
            Departamento dep = new Departamento(nomeDepartamento);
            listaDepartamentos.add(dep);
        }
    }
}
```

```java
public class RecursosHumanos {

}
```

1

*

```java
public class Departamento {
    public String nome;
    public Departamento(String nome) {
        this.nome = nome;
    }
}
```

COMPOSIÇÃO

- Primeiro, cada instância de departamento, "dep", pode pertencer apenas a uma única instância de Escola por vez. Mas uma instância de escola pode ter várias instâncias de departamento anexadas a ela.

- Portanto, isso torna a instância de escola a proprietária das instâncias "dep", e esse é o recurso herdado da agregação na composição.

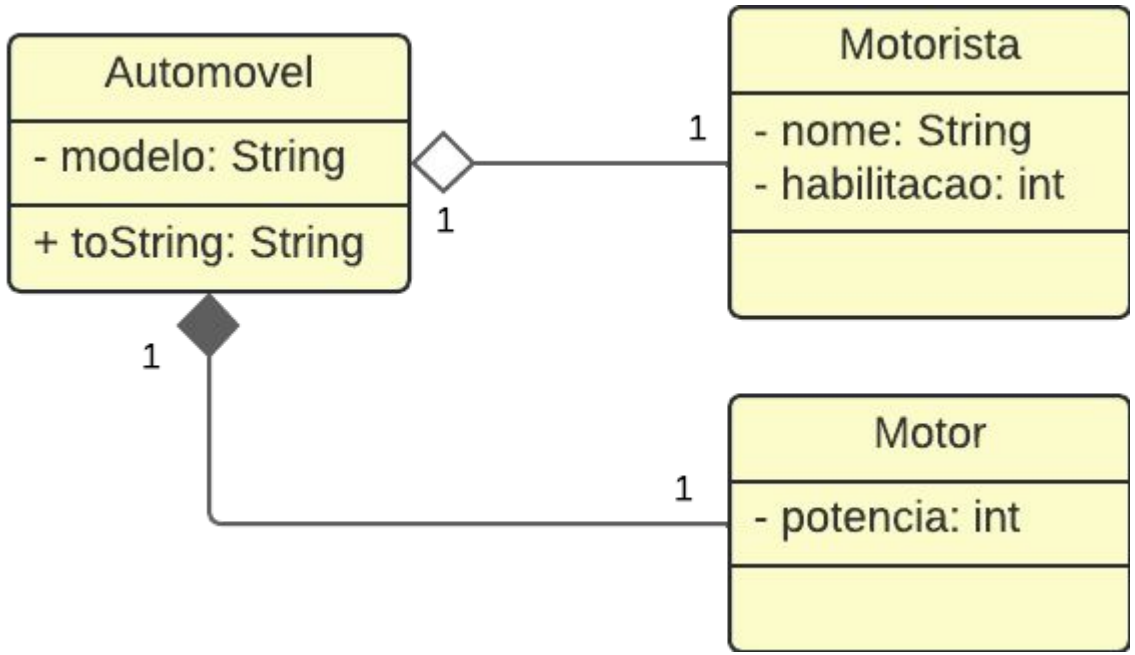
- O tempo de vida das instâncias de departamento depende das instâncias de Escola, uma vez que elas são criadas dentro da classe da Escola. Portanto, quando a instância de escola for descartada, as instâncias "dep" também serão eliminadas.

- Os departamentos não existem sem a escola.

Exercícios

Exercício 1

- **3 -** Implemente as classes conforme o diagrama:
  - Lembre-se de implementar os métodos de acesso e definição.
  - Os construtores das classes
- Na classe principal crie instâncias das classes implementadas para teste.



Exercício 2

- a

C . E . S . A . R

school

Dúvidas? Entre em contato com a gente:

grs@cesar.school