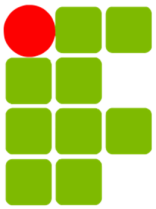


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARAÍBA
Campus João Pessoa

Programação Orientada a Objetos

Fausto Maranhão Ayres

9
Relacionamento entre Objetos (Unidirecional)

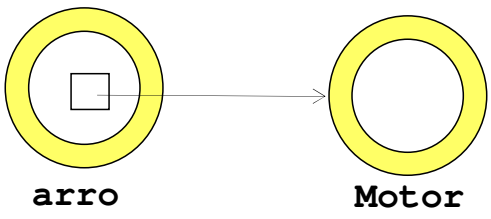
O que é relacionamento entre objetos?

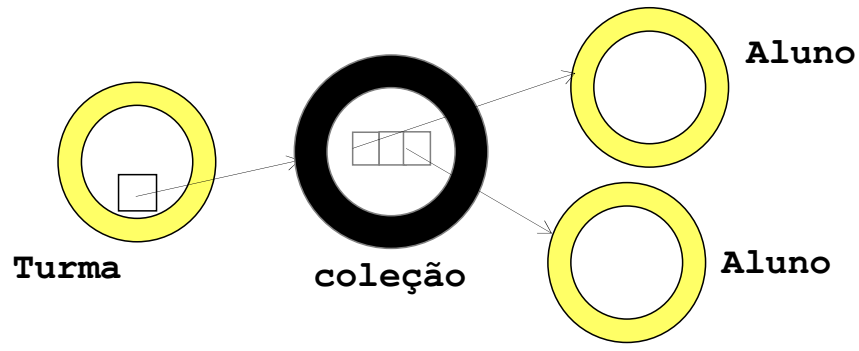
Relacionamentos

O que é relacionamento entre objetos

- Quando um objeto contém uma (ou mais) referência(s) para outro(s) objeto(s)

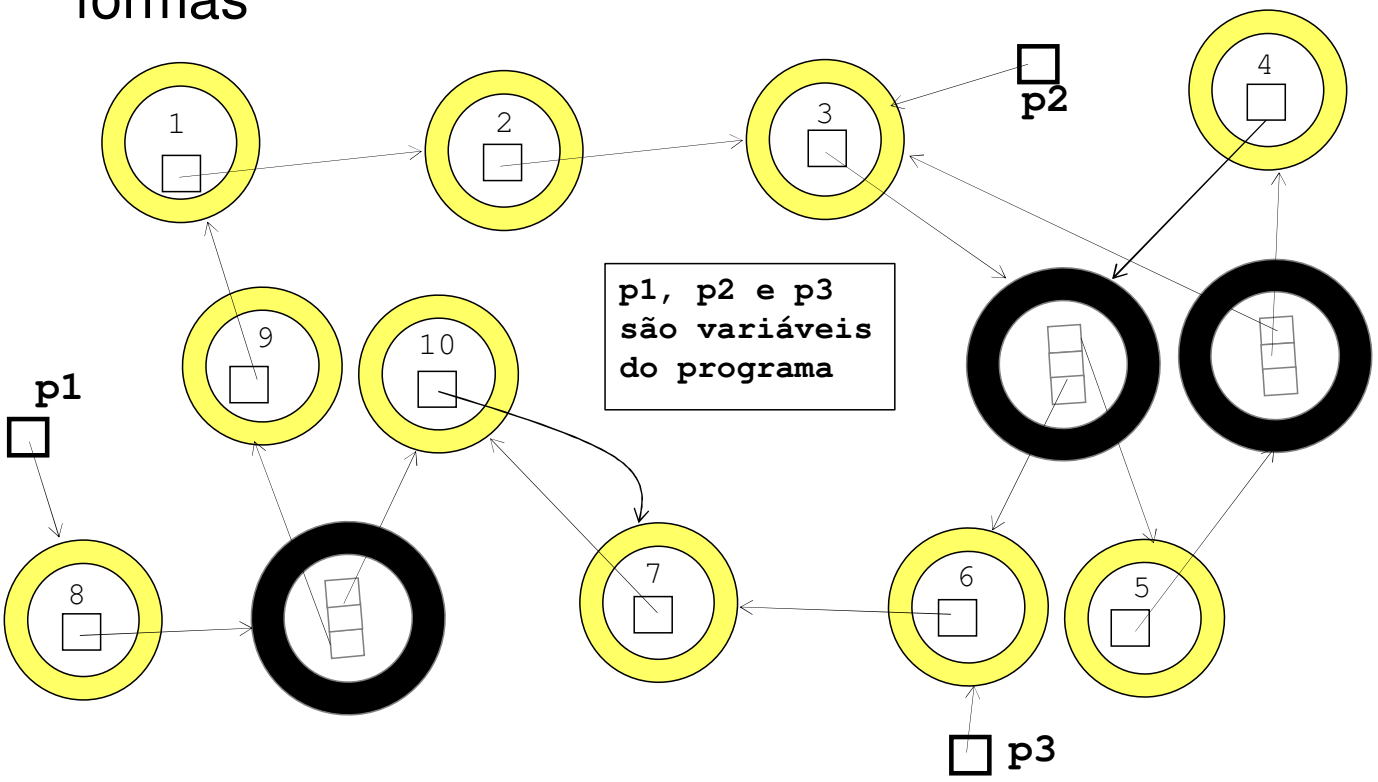
  Exemplos:





O que é um grafo de objetos

- É uma “rede” de objetos relacionados de várias formas



Tipos de relacionamento

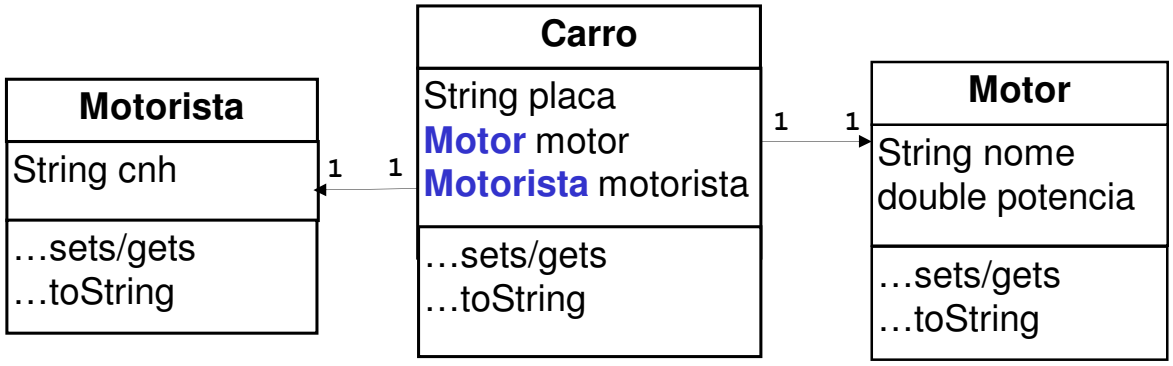
- Quanto à Cardinalidade:
  - um para um (1:1)
  - um para muitos (1:*)
  - muitos para muitos (*:*)
- Quanto à Direção:
  - Unidirecional
  - Bidirecional

Como implementar estes tipos de relacionamentos em **Java**?

Implementação do relacionamento Unidirecional um-para-um (1:1)

Exemplo

Um carro tem um motor
Um carro tem um motorista



- Logo, a classe Carro possui dois relacionamentos unidirecional 1:1

Implementação dos relacionamentos

```
public class Carro {
    private String placa;
    private Motor motor;
    private Motorista motorista;

    public Carro(String placa, Motor motor, Motorista motorista)
        this.placa = placa;
        this.motor = motor;
        this.motorista = motorista;
    }
```

Referência para objeto Motor
Referência para objeto Motorista

Armazenar referência

```
public class Motor {
    private String nome;
    private double potencia;
```

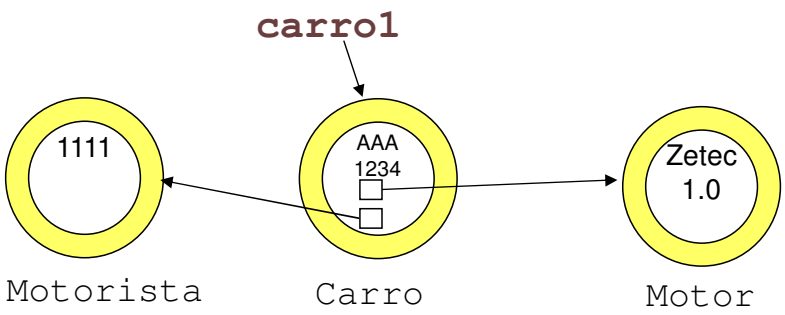
```
public class Motorista {
    private String cnh;
```

Adicione toString() nas 3 classes

Criar relacionamento

Teste

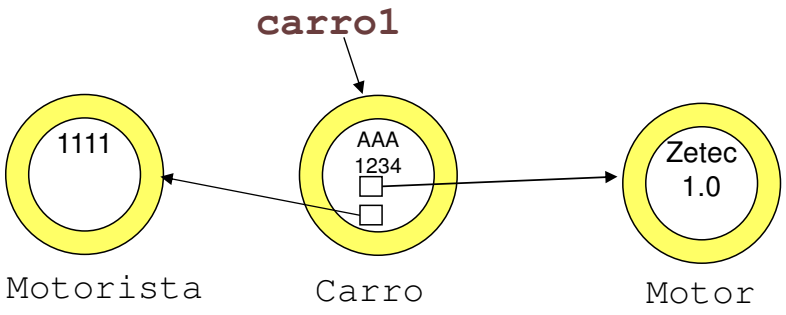
```
Motor motor = new Motor("Zetec", 1.0);
Motorista motorista = new Motorista("1111");
Carro carro1 = new Carro("AAA1234", motor, motorista);
```



Criar relacionamento

De outra maneira...

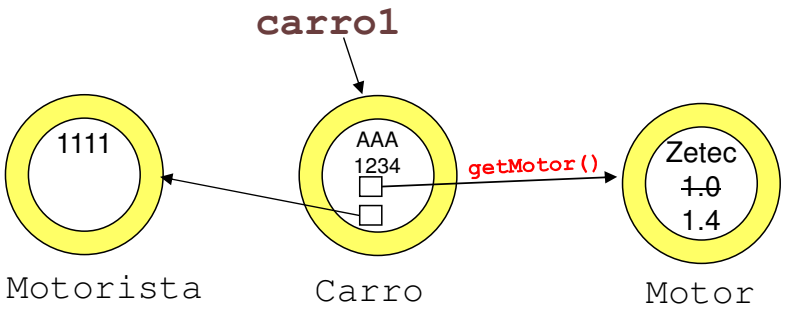
```
Carro carro1 = new Carro("AAA1234",
                         new Motor("Zetec",1.0),
                         new Motorista("1111"));
```



Navegar pelo relacionamento

- Navegar pelo relacionamento entre carro e motor
  - Alterar a potencia do motor para 1.4
  - Imprimir a potencia do motor

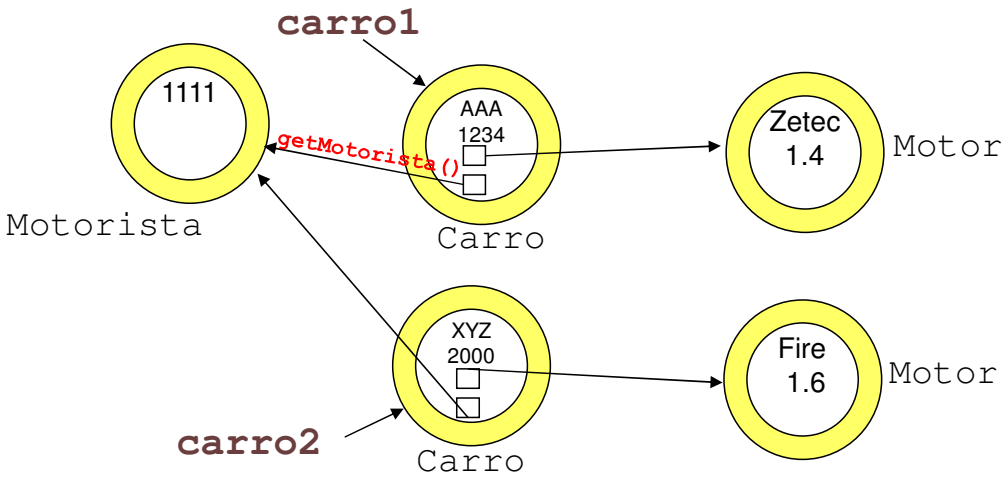
```
carro1.getMotor().setPotencia(1.4);
System.out.println(carro1.getMotor().getPotencia());
```



Navegar pelo relacionamento

- Criar um segundo carro com a placa “XYZ2000”, com o novo motor Fire 1.6 e com o motorista do primeiro carro.

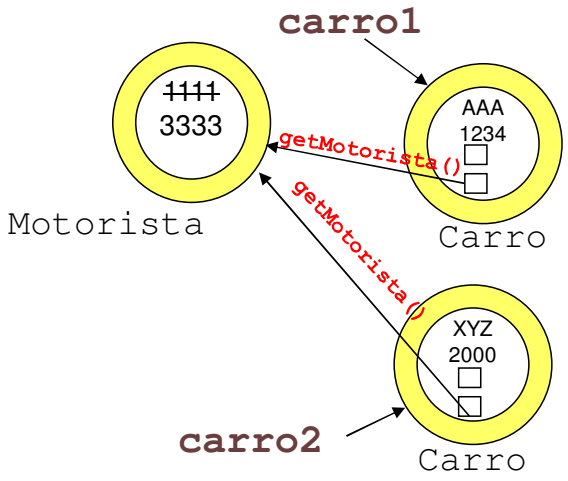
```
Carro carro2 = new Carro("XYZ2000",
       new Motor(“Fire",1.6), carro1.getMotorista());
```



Navegar pelo relacionamento

- Alterar a CNH do motorista do carro1:

```
carro1.getMotorista().setCnh(“3333");
System.out.println(carro2.getMotorista().getCnh()); 3333
```

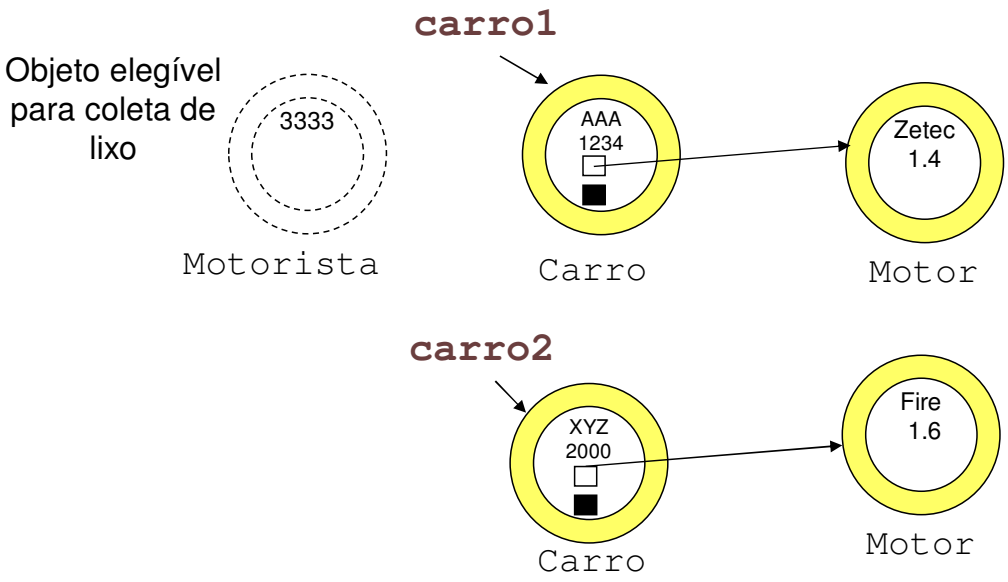


Os dois carros estão referenciando o mesmo motorista

Remover relacionamento

- Remover o motorista dos dois carros

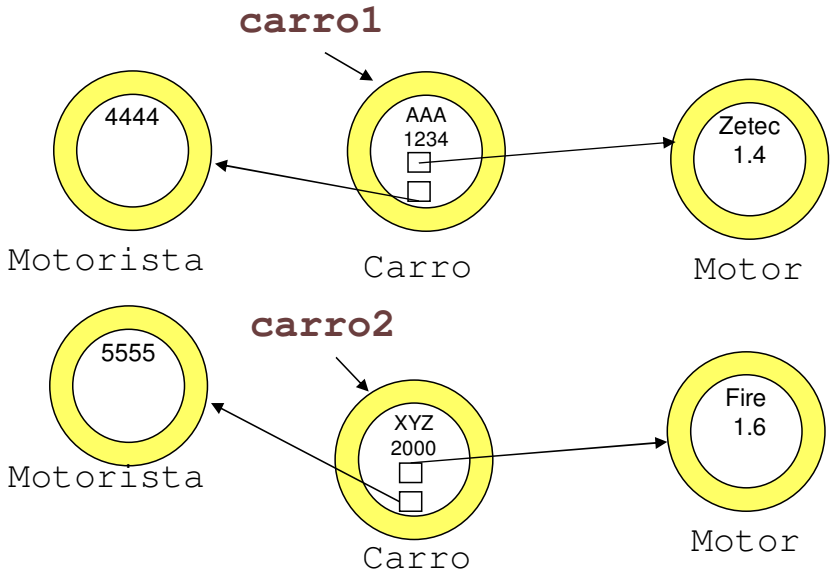
```
carro1.setMotorista(null);
carro2.setMotorista(null);
```



Criar novo relacionamento

- Criando novo motorista para os dois carros

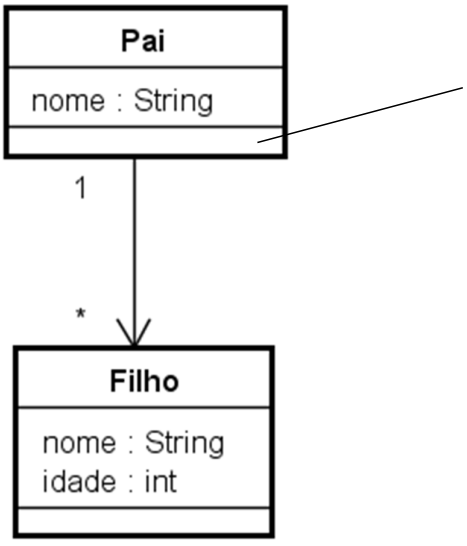
```
carro1.setMotorista(new Motorista("4444"));
carro2.setMotorista(new Motorista("5555"));
```



Implementação do relacionamento Unidirecional um-para-muitos (1:*)

Unidirecional 1:*

- Exemplo genérico: “um pai tem vários filhos”

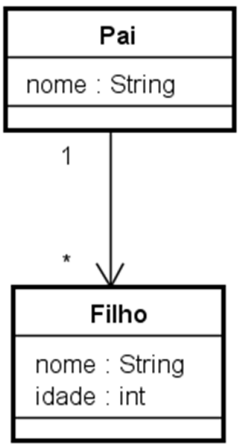


A classe Pai possui a **lista filhos** e métodos para:

- **Adicionar** um objeto Filho na lista
- **Remover** um objeto Filho da lista
- **Localizar** um objeto Filho na lista

Classe Pai

- Implementação do relacionamento 1:*




```
public class Pai {
  private String nome;
  private ArrayList<Filho> filhos =new ArrayList<>();

  public void adicionar(Filho f){
      filhos.add(f);
  }
  public void remover(Filho f){
      filhos.remove(f);
  }
  public Filho localizar(String nome) {
      for(Filho f: filhos) {
            if(f.getNome().equals(nome))
                  return f;
      }
      return null;
  }
```

Cria a lista vazia

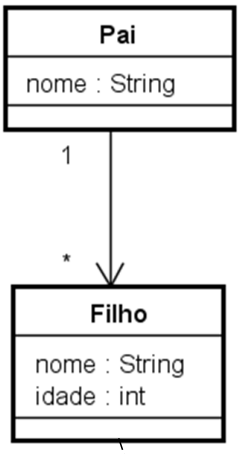
Busca pelo nome

Retorna o filho encontrado

Em caso de não encontrar

Classe Filho

- Não possui relacionamento com a classe Pai, pois é unidirecional

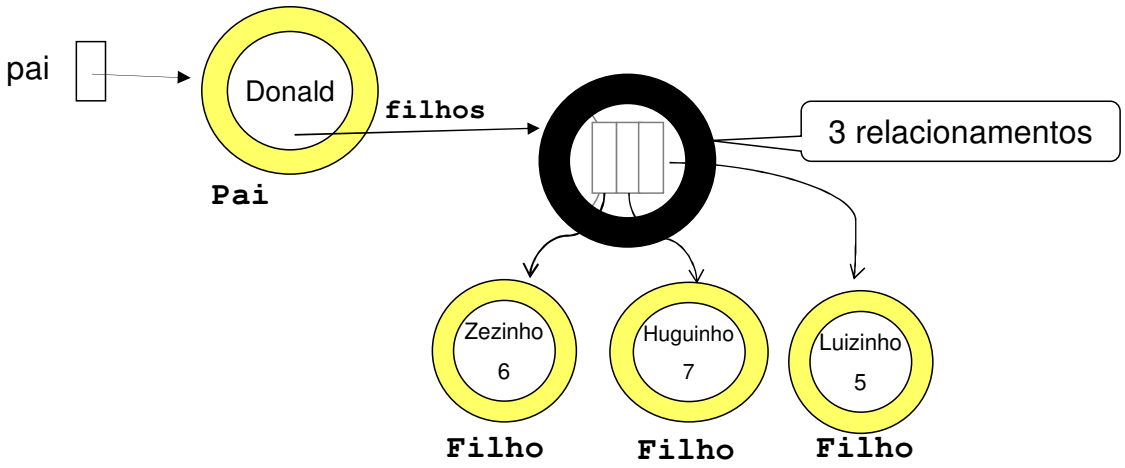



```
public class Filho{
  private String nome;
  private int idade;

  ...
}
```

Criar relacionamentos

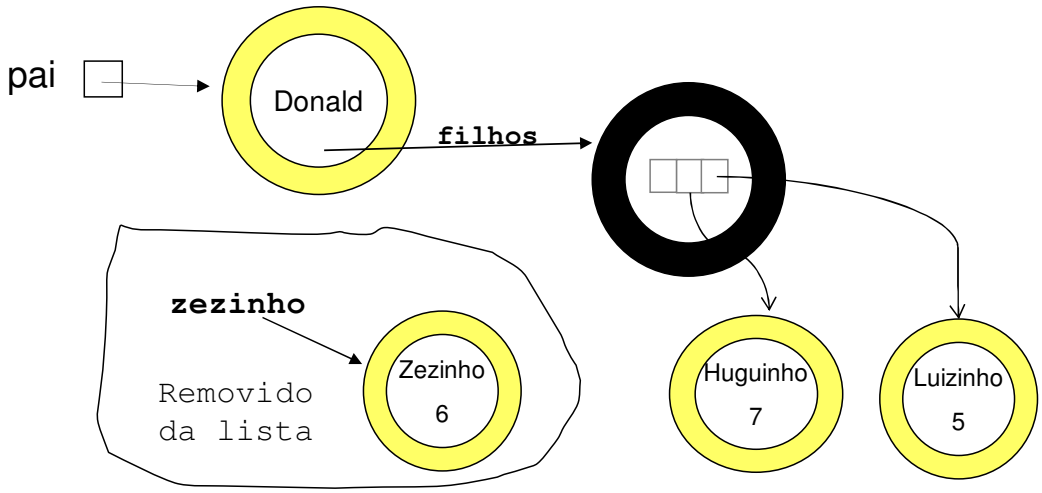
```
Pai pai = new Pai("Donald");
pai.adicionar(new Filho("Zezinho", 6));
pai.adicionar(new Filho("Huguinho", 7));
pai.adicionar(new Filho("Luisinho", 5));
System.out.println(pai);
```



Localizar e remover Filho

```
Filho zezinho;
zezinho = pai.localizar("Zezinho");
if(zezinho != null)
   pai.remover(zezinho);    //remover relacionamento


zezinho = pai.localizar("Zezinho");
if(zezinho == null)
   System.out.println("Zezinho inexistente");
```



Outros métodos de Pai

```
// localizar o filho com menor idade
System.out.println(pai.obterCacula());
```

```
   [nome="Luizinho", idade=5]
```

```
// calcular a idade media dos filhos
System.out.println(pai.obterIdadeMedia());
```

```
   6.0
```