



Universidade Estadual de Maringá
Departamento de Informática



Classes de projeto Implementando Herança e Polimorfismo

Conteúdo baseado nos materiais dos Professor
Marcos Aurélio Domingues (DIN/UEM)

Prof.^a Juliana Keiko Yamaguchi
abril de 2019

Objetivos

- Compreender como implementar o conceito de herança e polimorfismo.

Introdução

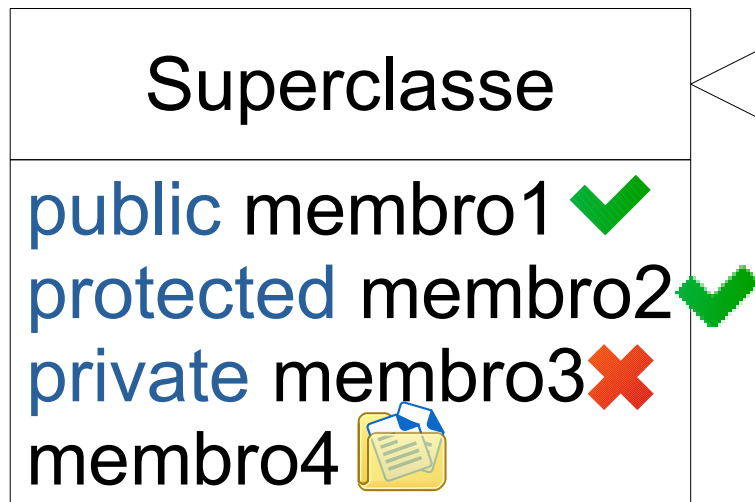
- No contexto da programação orientada a objetos, qual o conceito de herança?
- Qual a diferença entre uma superclasse e uma subclasse?

Herança

- Herança é o mecanismo que permite a uma classe (subclasse) herdar todos os atributos e métodos de outra classe (superclasse).
- Superclasses tendem a ser mais gerais enquanto que subclasses mais específicas.
- Qual a regra de acesso das subclasses para os atributos e métodos declarados com os modificadores public, protected, default e private?

Herança

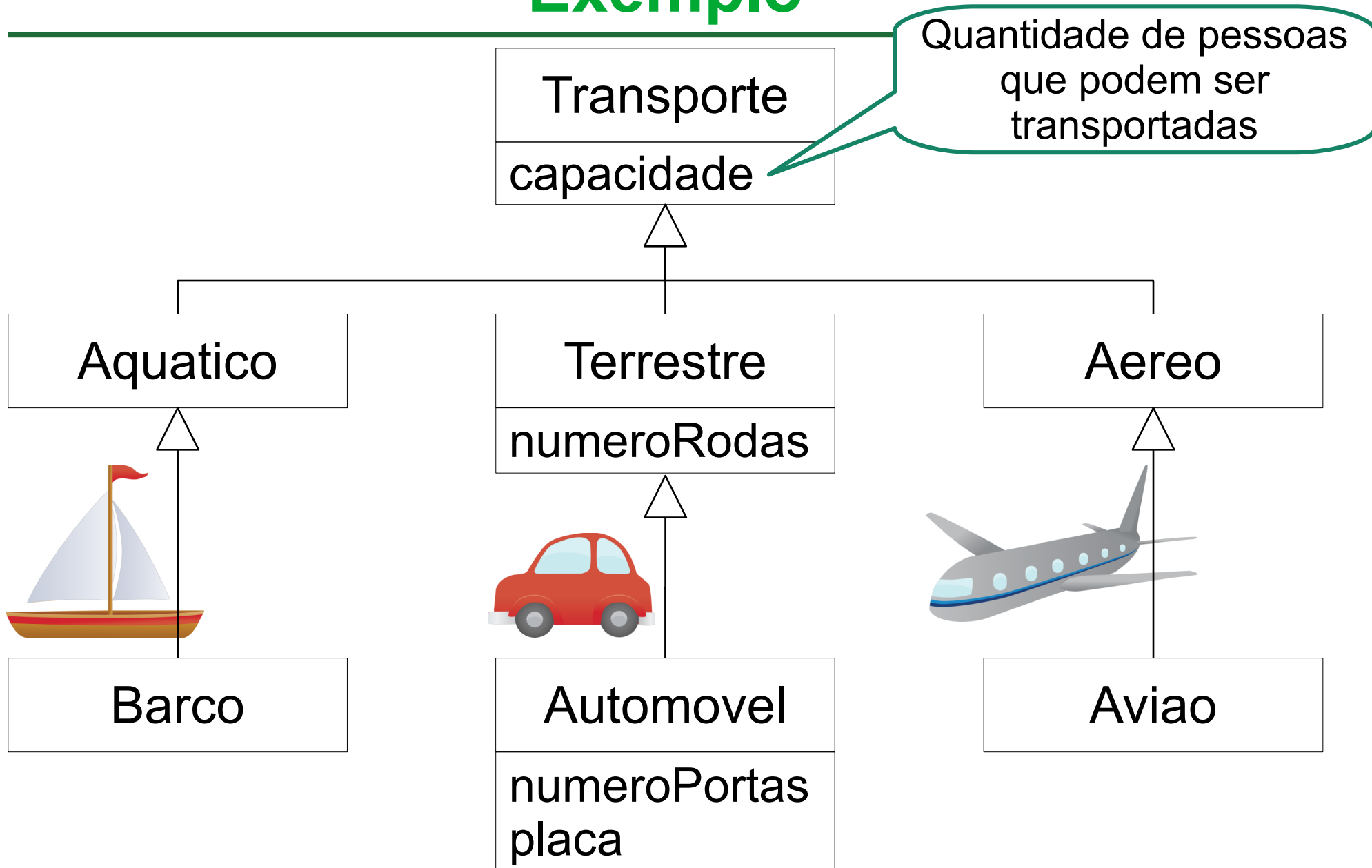
- O acesso de uma subclasse aos membros (atributos e métodos) de uma superclasse segue as seguintes restrições:



- A subclasse tem acesso aos membros public e protected.
- O acesso a membros privados só é permitido na superclasse.
- O acesso ao membro default somente se a subclasse estiver no mesmo pacote da superclasse.

Herança

Exemplo



Herança

Exemplo

- Toda classe que herda de uma outra, acaba herdando os seus atributos e métodos.
- Por exemplo:
 - A classe Transporte possui um atributo chamado capacidade;
 - A classe Aquático, como ela herda de Transporte, pode-se dizer que também possui o atributo capacidade;
 - E Barco, como herda de Aquático, também possui o atributo capacidade.

Herança

Exemplo

- Quanto mais alta a classe na hierarquia, mais ela tende a ser abstrata, ou menos definida, com menos atributos e métodos.
 - Isso garante que a classe tenha mais chance de ser reusada por outras classes.

Herança

Exemplo

- Entendeu? Não?
- Suponha que a classe Transporte, além do atributo capacidade, possuísse também o atributo número de rodas.
- Nesse caso, não seria interessante nem faria sentido para a classe Aquático herdar os atributos de Transporte.

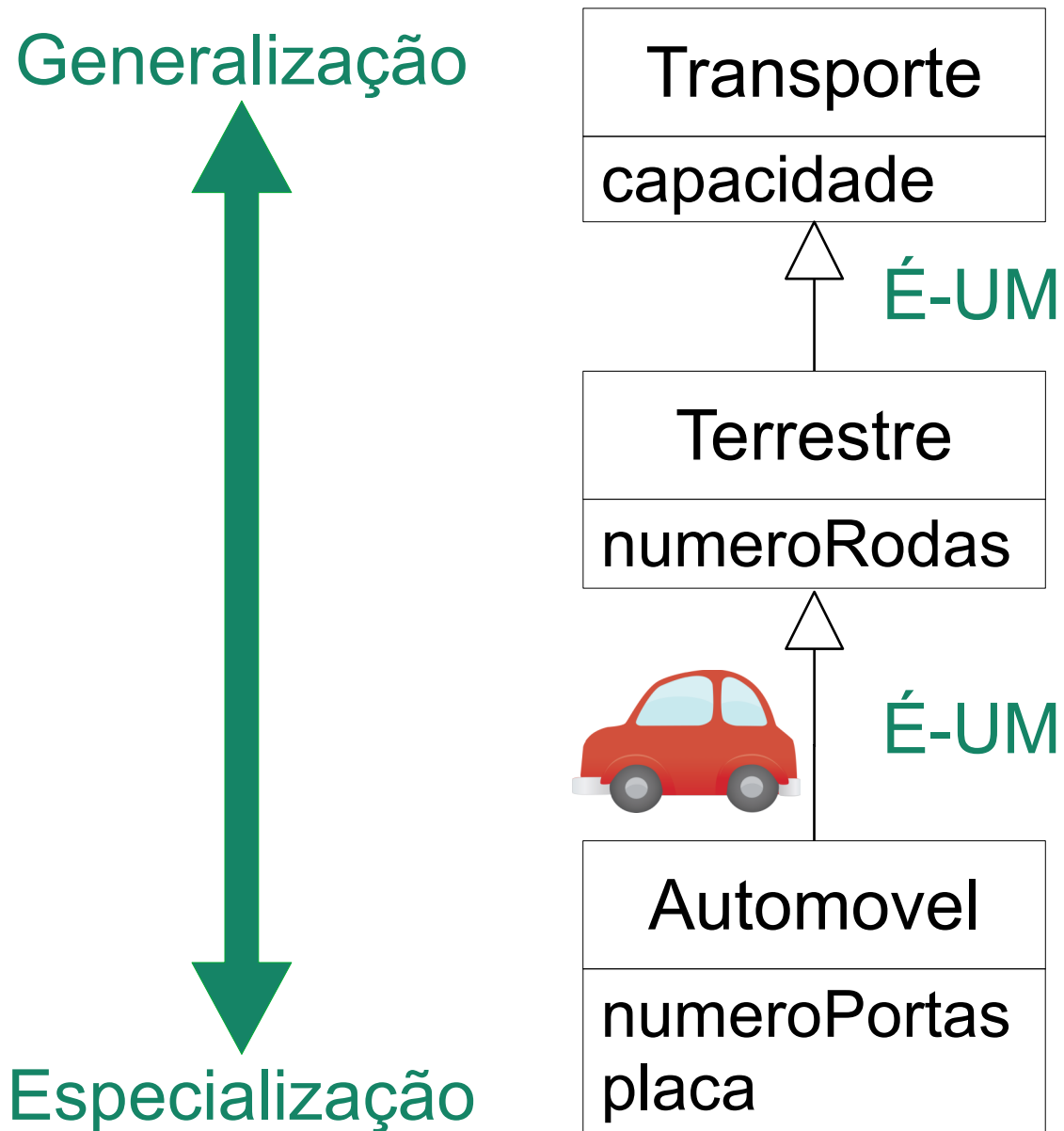
Herança

Generalização vs Especialização

- Generalização
 - classes mais genéricas e abstratas disponíveis, as quais podem ser usadas para outras descenderem delas.
- Especialização
 - classes que estão numa posição inferior na hierarquia e possuem características e comportamentos mais especializados.

Herança

Generalização vs Especialização



Herança

Em Java

- Na linguagem Java, todas as classes descendem da classe Object.
- Para implementar herança, usa-se a palavra reservada *extends*, para declarar que uma classe é herdeira de outra.
- Exemplo:

```
public class Subclasse extends Superclasse{  
  
    //código fonte da classe  
  
}
```

Herança

Simple vs Múltipla

- Herança Simples
 - Cada classe pode ter apenas uma superclasse.
- Herança Múltipla
 - É a capacidade de uma classe possuir mais de uma superclasse e herdar os atributos e métodos de todas as superclasses.

Herança

Simples vs Múltipla

- Para simular a herança múltipla em Java, usa-se **interfaces**.
- Portanto, uma classe java pode herdar **somente de uma classe**, mas pode implementar várias interfaces.

```
public class Subclasse extends Superclasse  
implements Interface1, Interface2{  
  
    //código fonte da classe  
  
}
```

Classe Interface

- Uma classe Interface lista um conjunto de assinaturas de métodos que determina um comportamento ou característica.
- As implementações dos métodos devem ser realizadas pela classe que a implementa.

```
public interface Interface1{  
    public abstract tipoRetorno nomeMetodo1();  
    public abstract tipoRetorno nomeMetodoN();  
}
```

Classe Interface

- Assim como uma classe, a interface não pode ser `private` nem `protected`. Ela somente pode ser `public` ou `default`.
- Métodos de interface são implicitamente `public` e `abstract`. Os modificadores `final` e `static` não são aceitos por métodos de interface.
- Atributos de interface são implicitamente `public`, `static` e `final` (isto é, não são atributos e sim variáveis de classe constantes).

Classe Interface

Exemplo

```
public interface Guiavel{  
  
    public abstract void irFrente();  
    public abstract void irRe();  
    public abstract void irEsquerda();  
    public abstract void irDireita();  
}
```

```
public class Barco implements Guiavel{  
  
    public void irFrente(){//implementação}  
    public void irRe(){//implementação}  
    public void irEsquerda(){//implementação}  
    public void irDireita(){//implementação}  
}
```

Exercício 01

- Implemente as classes ilustradas no diagrama anterior.
 - Crie um projeto chamado “Heranca”.
 - No pacote padrão, crie as classes e respectivos atributos.
 - Acrescente atributos e comportamentos para as classes.

Polimorfismo

- O polimorfismo deriva da palavra polimorfo, que significa multiforme, ou que pode variar a forma.
- Para a POO, polimorfismo é a habilidade de objetos de classes diferentes responderem a uma mesma mensagem de diferentes maneiras.
 - Ou seja, várias formas de responder à mesma mensagem.

Polimorfismo

- De forma prática, polimorfismo é alterar a implementação de um método, mantendo o mesmo nome.
- Para aplicar o polimorfismo, uma classe deve ter acesso aos métodos de outra.
 - Quando é possível aplicar o polimorfismo?

Polimorfismo

- Uma forma de empregar o polimorfismo é por meio da herança.
- Outra forma é aplicar o polimorfismo quando existem métodos abstratos que devem ser sobrescritos.

Polimorfismo

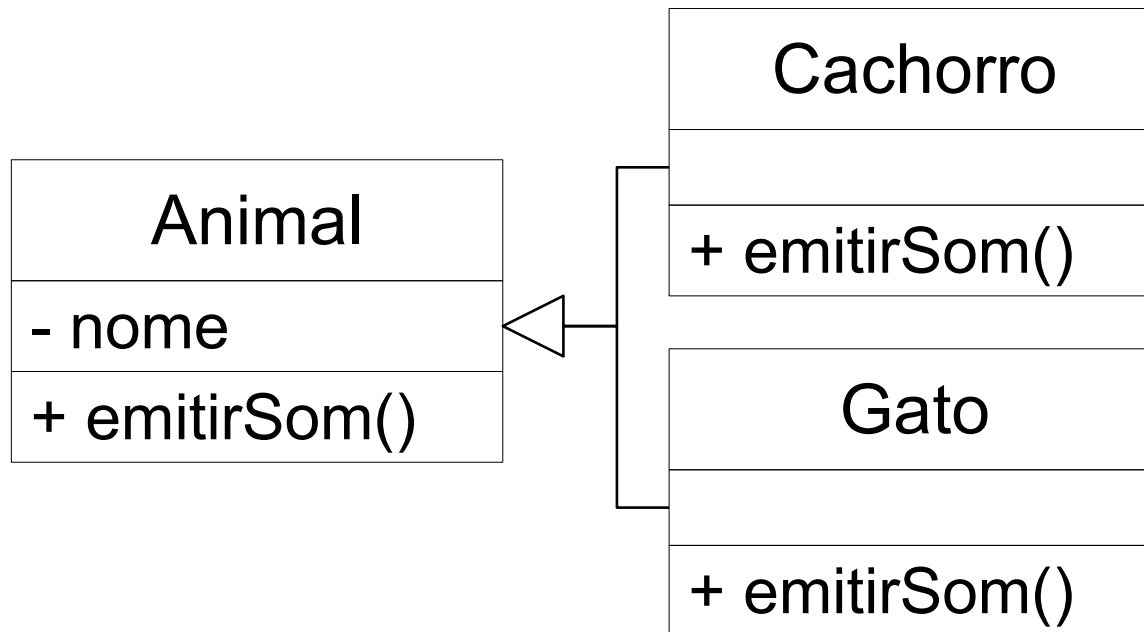
Por herança

- A subclasse pode sobrescrever o método herdado da superclasse.
- Dessa forma, a subclasse pode se comportar de maneira mais específica.

Polimorfismo

Por herança

- Exemplo:



- Como implementar as várias formas de `emitirSom()`?

Polimorfismo

Por herança

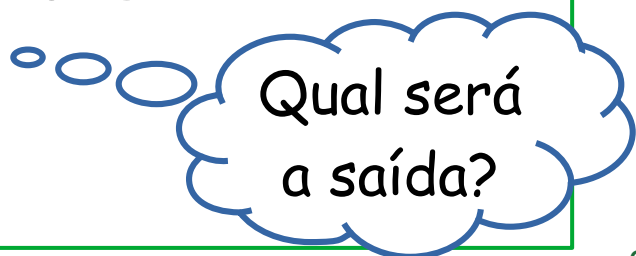
```
public class Animal {  
    protected String nome;  
  
    public String emitirSom() {  
        return "Animal emitindo som...";  
    }  
}
```

```
public class Cachorro extends Animal{  
  
    @Override  
    public String emitirSom() {  
        return "Au-Au";  
    }  
}
```


Polimorfismo

Por herança

```
public class ProjetoAnimal {  
  
    public static void main(String[] args) {  
        Cachorro cachorro1 = new Cachorro();  
        System.out.println("cachorro 1 " +  
cachorro1.emitirSom());  
  
        Animal cachorro2 = new Cachorro();  
        System.out.println("cachorro 2 " +  
cachorro2.emitirSom());  
  
        Animal cachorro3 = new Animal();  
        System.out.println("cachorro 3 " +  
cachorro3.emitirSom());  
    }  
}
```



Qual será
a saída?

Polimorfismo

Por herança

- A saída será:

```
cachorro 1 Au-Au  
cachorro 2 Au-Au  
cachorro 3 Animal emitindo som...
```

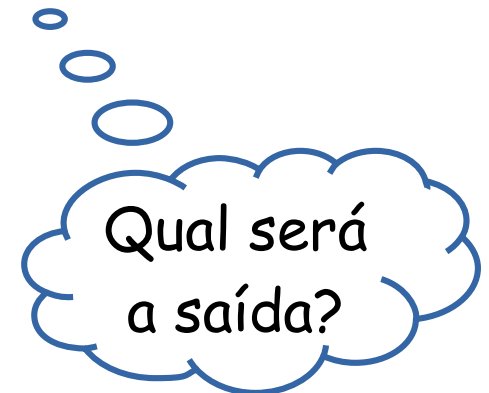
- Um objeto do tipo Cachorro nunca poderá se comportar como um objeto do tipo Animal?

Polimorfismo

Por herança

- A subclasse pode realizar uma chamada para o método da superclasse:

```
public class Cachorro extends Animal{  
  
    @Override  
    public String emitirSom() {  
        return super.emitirSom() + "Au-Au";  
    }  
}
```



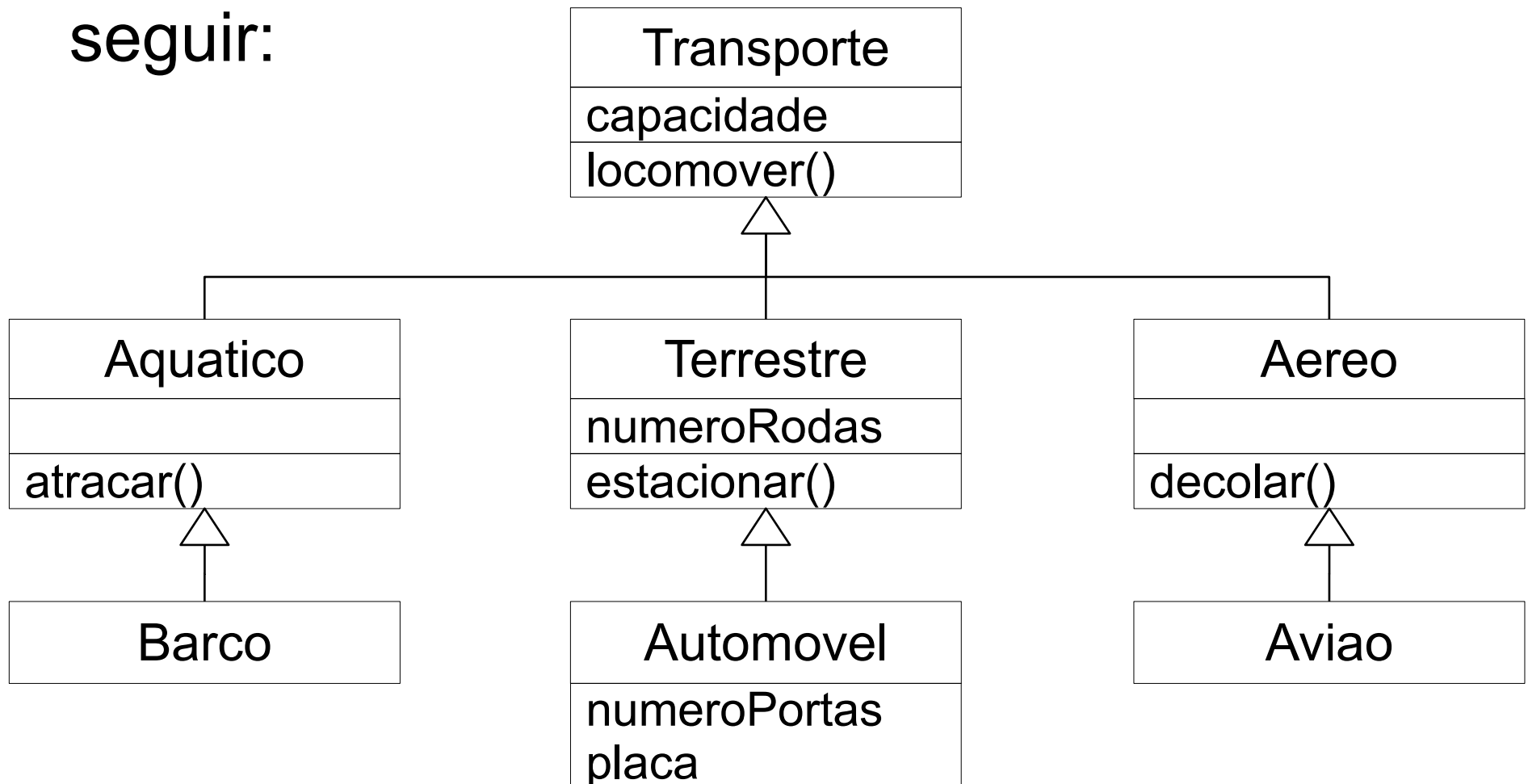
Polimorfismo

Por herança

- Suponha que no diagrama do exercício 01, a classe Transporte possua o método locomover().
- Desse modo, todas as classes filhas também possuem o comportamento de locomover().
- No entanto, o método locomover:
 - Para um barco, significa navegar;
 - Para um automóvel, significa correr;
 - Para um avião, significa voar;

Exercício 02

- Continuando o exercício anterior, realize as modificações de acordo com o diagrama a seguir:



Polimorfismo

Sobreposição (*Overriding*)

- O polimorfismo por herança é também conhecido como polimorfismo de sobreposição ou *overriding*.
- Exemplo:

```
public class A{  
    public void metodo1() {  
        System.out.println("Classe A");  
    }  
}
```

```
public class B extends A{  
    public void metodo1() {  
        System.out.println("Classe B");  
    }  
}
```

Polimorfismo

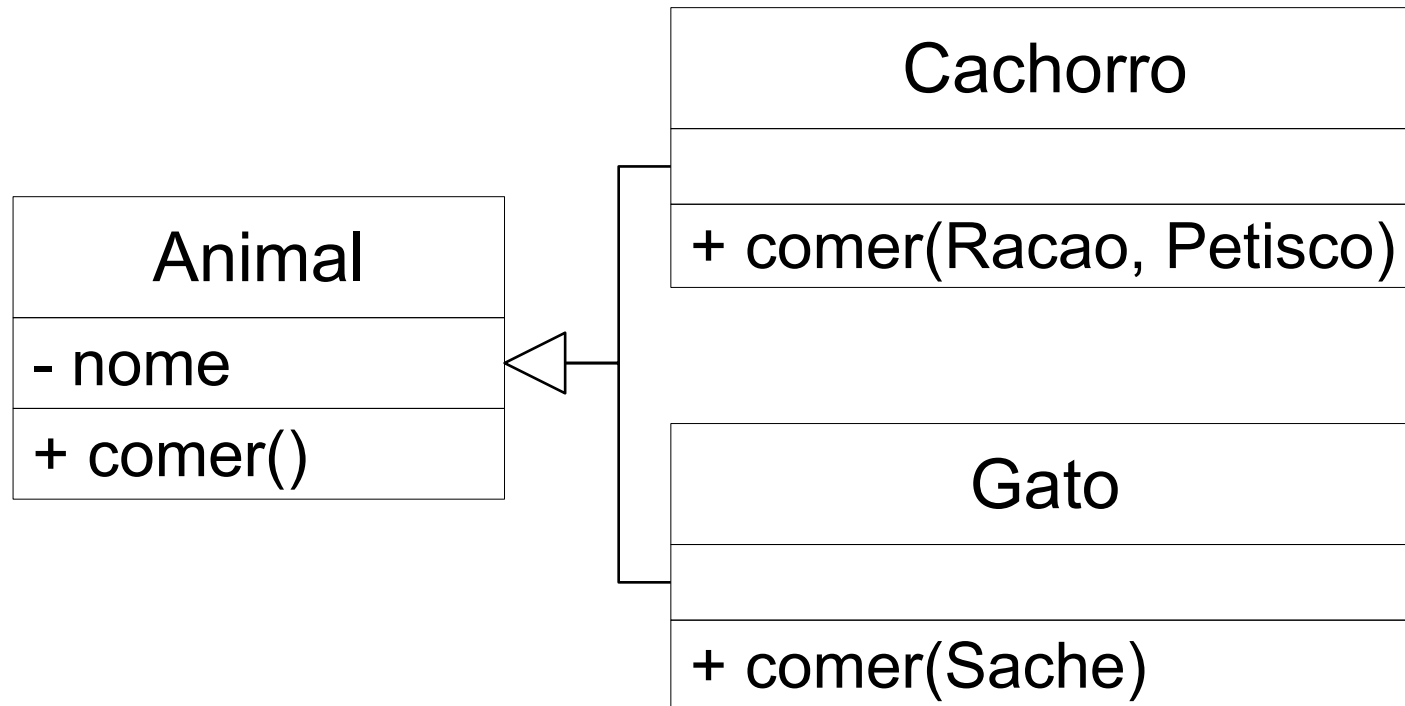
Sobrecarga (*Overloading*)

- Permite que um método de determinado nome tenha comportamentos distintos, em função de diferentes parâmetros que ele recebe.
- Cada método difere no número e no tipo de parâmetros.

Polimorfismo

Sobrecarga (*Overloading*)

- Exemplo:



Polimorfismo

Sobrecarga (*Overloading*)

```
public class Animal{  
    public String comer() {  
        return "Animal comendo...";  
    }  
}
```

```
public class Cachorro extends Animal{  
  
    @Override  
    public String comer(Racao r, Petisco p) {  
        return "Animal comendo " + r.getNome() +  
" e " + p.getNome();  
    }  
}
```

Polimorfismo

Sobrecarga (*Overloading*)

- O polimorfismo de sobrecarga normalmente acontece quando sobrecarregamos os métodos **construtores**.
- É comum para uma classe ter várias maneiras de instanciá-la.

Polimorfismo

Sobrecarga (*Overloading*)

- Exemplo:

```
public class Automovel{  
  
    private int numeroPortas;  
    private String placa;  
  
    public Automovel(int capacidade,String placa) {  
        super.capacidade = capacidade;  
        this.placa = placa;  
    }  
  
    public Automovel(int numeroPortas) {  
        this.numeroPortas = numeroPortas;  
    }  
}
```

Polimorfismo

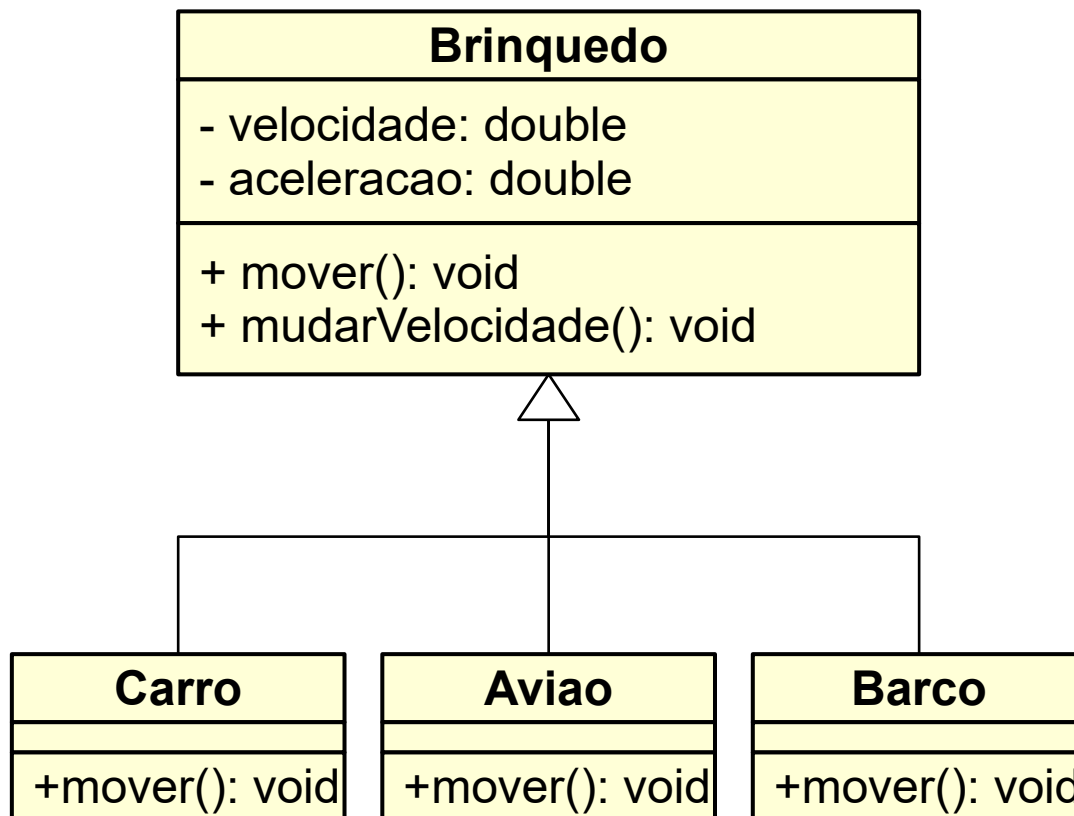
De inclusão

- As classes herdeiras realizam a sobrescrita dos métodos da superclasse.
- No polimorfismo de inclusão, a chamada dos métodos especializados das subclasses é abstraída.

Polimorfismo

De inclusão – Exemplo

- Considere que a classe Brinquedo possui como descendentes as classes Carro, Avião e Barco, conforme ilustra a Figura.



Observe que as classes filhas sobrepõem o método `mover()` da classe **Brinquedo**.

Polimorfismo

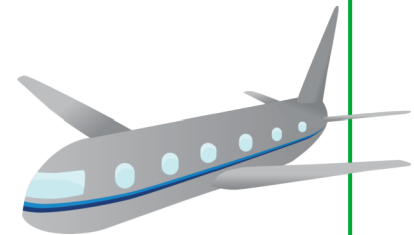
De inclusão – Exemplo

```
public class Brinquedo{  
    public String mover() {  
        return "Brinquedo movendo...";  
    }  
}
```

```
public class Carro extends Brinquedo{  
    public String mover() {  
        return "Correndo";  
    }  
}
```



```
public class Aviao extends Brinquedo{  
    public String mover() {  
        return "Voando";  
    }  
}
```



```
public class Barco extends Brinquedo{  
    public String mover() {  
        return "Navegando";  
    }  
}
```



Polimorfismo

De inclusão – Exemplo

```
public class ProjetoBrinquedo{  
    public static void main(String args[]){  
        Carro carro = new Carro();  
        carro.mover();  
        Aviao aviao = new Aviao();  
        aviao.mover();  
        Barco barco = new Barco();  
        barco.mover();  
    }  
}
```

- A decisão sobre qual método sobrescrito deve ser selecionado é tomada em tempo de execução, considerando a classe da instância (objeto) que o está chamando.

Polimorfismo

De inclusão – Exemplo

- Suponha que deva ser implementado uma classe ControleRemoto que deverá invocar o método mover() de um objeto Brinquedo.
- Como o controle remoto saberá qual método mover() ele deve chamar, se ele tem disponível três tipos de mover diferente (um para cada brinquedo)?

Polimorfismo

De inclusão – Exemplo

- Vamos ver inicialmente como fica a implementação do Controle Remoto.

```
public class ControleRemoto{  
    private Brinquedo brinquedo;  
  
    public void mover(Brinquedo b) {  
        this.brinquedo = b;  
        this.brinquedo.mover();  
    }  
}
```

Polimorfismo

De inclusão – Exemplo

- O programa abaixo mostra como fica a classe que possui o método principal (main).

```
public class ProjetoBrinquedo{  
    public static void main(String args[]){  
        Carro carro = new Carro();  
        ControleRemoto cr = new ControleRemoto();  
        cr.mover(carro);  
    }  
}
```

- Foi criado um brinquedo do tipo Carro, e o objeto **cr** do tipo ControleRemoto.
- Foi enviado o objeto carro para o objeto **cr** através da chamada mover() do ControleRemoto.
- Como o objeto carro é um brinquedo, o método mover() do controle remoto poderá ser executado.

Polimorfismo

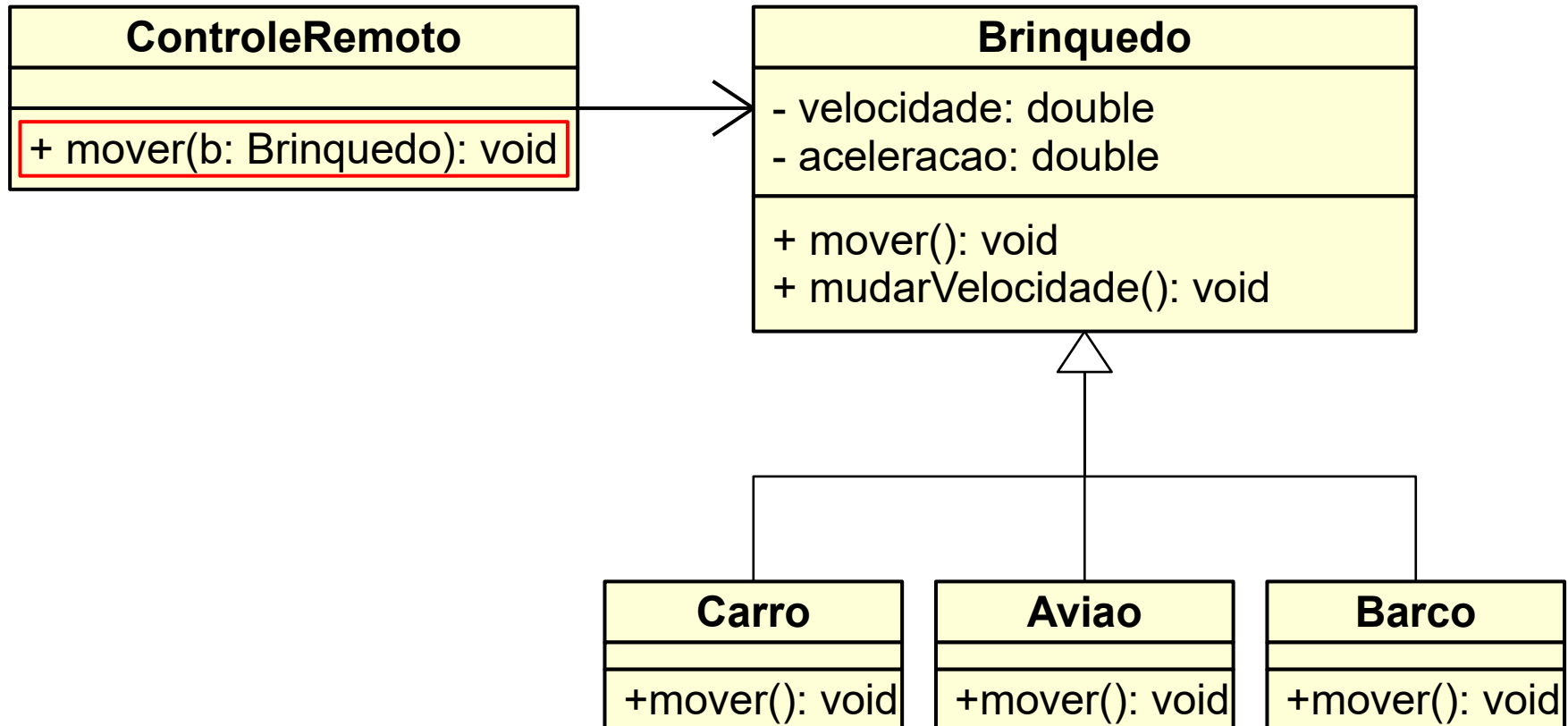
De inclusão – Exemplo

- No exemplo visto anteriormente, em que na classe ProjetoBrinquedo, foi criado um objeto do tipo Carro e outro do tipo ControleRemoto, utilizou-se o polimorfismo de inclusão.
- Isso foi feito substituindo a classe Brinquedo pela classe Carro dentro da classe ControleRemoto.

Polimorfismo

De inclusão – Exemplo

- Observe o diagrama:



Polimorfismo

De inclusão – Exemplo

- No diagrama anterior, pode-se notar que a classe ControleRemoto está relacionada com a classe Brinquedo, pois possui um atributo do tipo Brinquedo.
- Mas, como as classes Carro, Avião e Barco são do tipo Brinquedo, elas podem substituir a classe Brinquedo em qualquer método que a utilize.

Polimorfismo

De inclusão – Exemplo

- A capacidade do objeto (brinquedo) do tipo Brinquedo na classe ControleRemoto de receber qualquer objeto de subclasses da classe Brinquedo é que caracteriza o **polimorfismo de inclusão**.

Considerações finais

- A herança é um recurso do paradigma orientada a objetos utilizado para promover o reuso.
 - Uma subclasse pode herdar atributos e métodos da superclasse.
- Os métodos herdados podem ter sua implementação alterada.
 - A operação é a mesma, mas o comportamento é especializado.

Considerações finais

- Relacione as definições a seguir com o tipo de polimorfismo:

A)

Quando uma subclasse altera a implementação do corpo do método da superclasse, mantendo a sua assinatura.

☐

Polimorfismo por inclusão

B)

Quando uma subclasse altera a implementação do método da superclasse, alterando os de parâmetros de entrada.

☐

Sobrescrita

C)

Quando uma subclasse altera o método da superclasse, mas esse método é invocado no contexto da superclasse.

☐

Sobrecarga

Considerações finais

- Relacione as definições a seguir com o tipo de polimorfismo:

A)

Quando uma subclasse altera a implementação do corpo do método da superclasse, mantendo a sua assinatura.

B)

Quando uma subclasse altera a implementação do método da superclasse, alterando os de parâmetros de entrada.

C)

Quando uma subclasse altera o método da superclasse, mas esse método é invocado no contexto da superclasse.

C

Polimorfismo por inclusão

A

Sobrescrita

B

Sobrecarga