Técnicas de Orientação a Objetos

Etec

Cubatão

Prof. Gunther Graf
gunthergrafpr@gmail.com

POLIMORFISMO

POLIMORFISMO

- ✓ É o último princípio que serve de base para a Programação Orientada a Objeto.
- ✓ Esse princípio também aposta na ideia da reutilização para facilitar o dia a dia da programação.
- ✓ Ele é também bastante importante tanto para o entendimento de programas OO em Java e outras linguagens, como também é um mecanismo bastante sofisticado para permitir a reutilização e flexibilidade durante o desenvolvimento.

POLIMORFISMO

- **Entender o princípio do Polimorfismo;**
- **Conhecer Polimorfismo de Sobreposição e de Inclusão;**

Prof. Gunther Graf
gunthergrafpr@gmail.com

Etec

GOVERNO DO ESTADO
SÃO PAULO

Qu4m 4 4554? 5er@ m4u cl0ne? 5le é 1gu@l @ m1n, mas sua F05M@ de f@l@r é d1f5r5nt5.

Quem é esse? Será meu clone? Ele é igual a min, mas sua FORMA de falar é diferente.

Isso é POLIMORFISMO!

Prof. Gunther Graf
gunthergrafpr@gmail.com

POLIMORFISMO

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

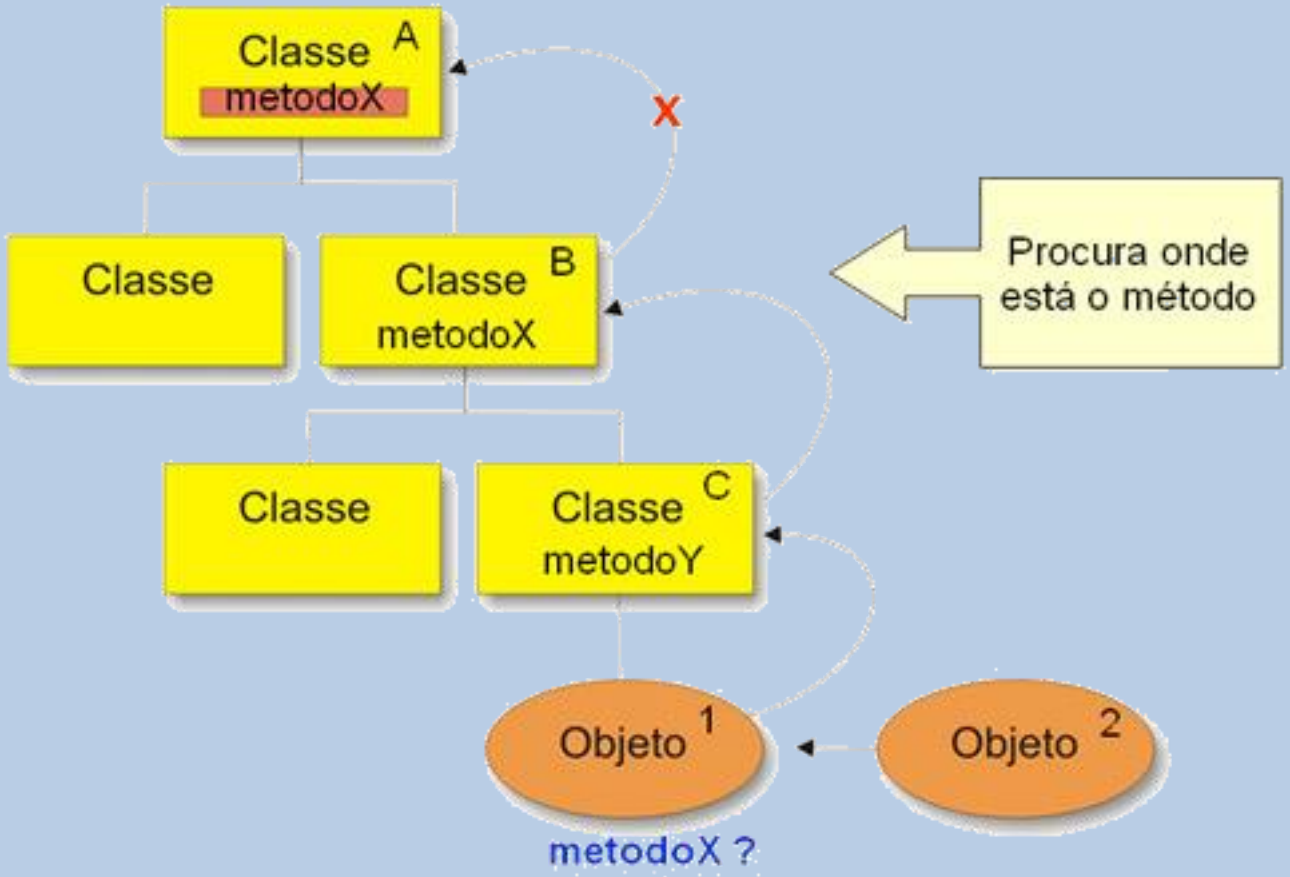
TIPOS DE POLIMORFISMO

- Polimorfismo de Sobrecarga;
- **Polimorfismo de Sobreposição;**
- Polimorfismo de Inclusão.

POLIMORFISMO DE SOBREPOSIÇÃO

- Polimorfismo de sobreposição é a redefinição de métodos em classes descendentes.
- Ou seja, um método de uma classe filha com o mesmo nome de um método de uma classe mãe irá sobrepor esse último.
- Vejamos o exemplo no slide a seguir.

POLIMORFISMO DE SOBREPOSIÇÃO



POLIMORFISMO DE SOBREPOSIÇÃO

- Como pode ser observado na hierarquia de classes apresentada, existe:
  1. Uma **classe A,** que implementa um **metodoX ()**;
  2. Uma **classe B,** que implementa um método com o mesmo nome;
  3. Uma **classe C,** que implementa um **métodoY ()**.

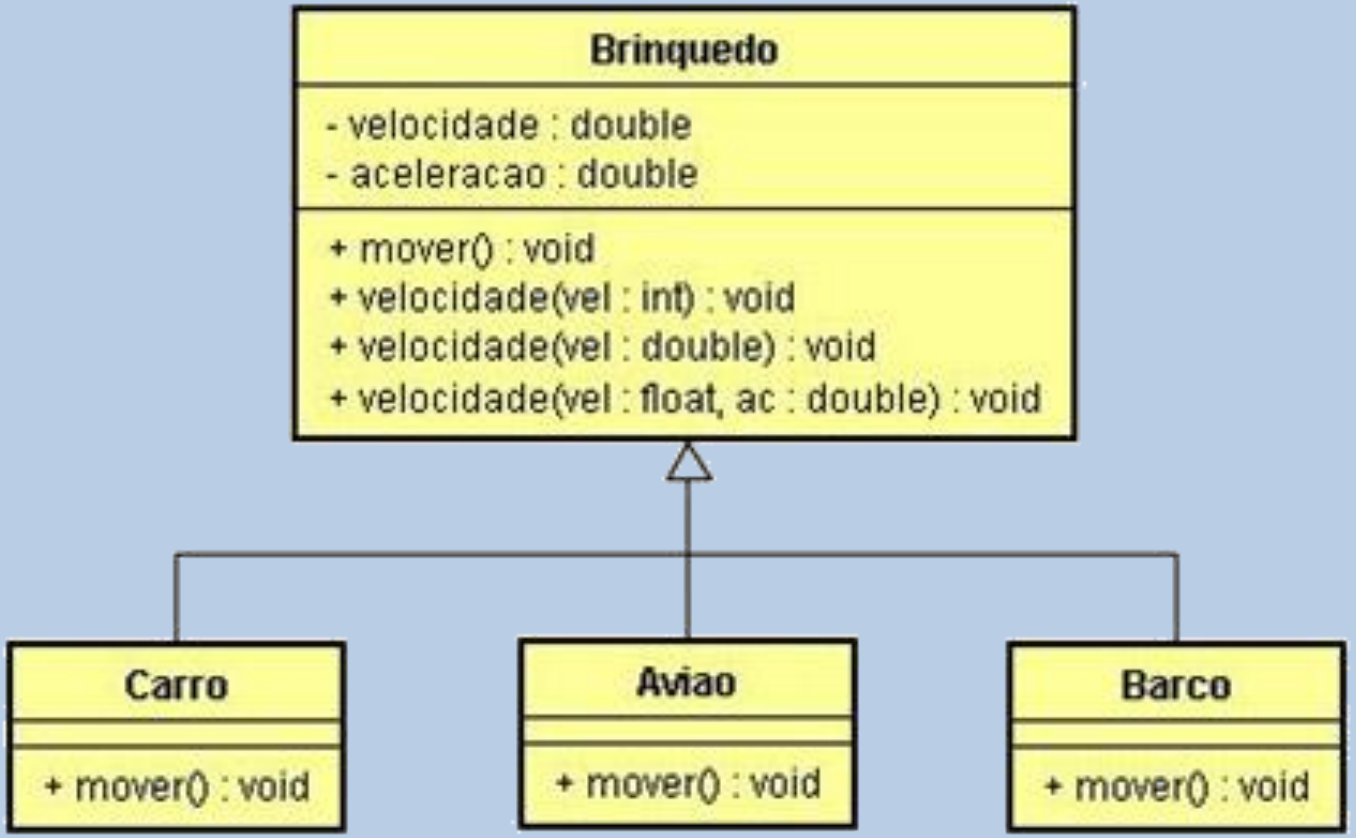
POLIMORFISMO DE SOBREPOSIÇÃO

- O que aconteceria se fosse solicitado ao **Objeto1** da classe C a execução do **metodoX()**?
- Conforme você viu nas aulas anteriores sobre Herança, esse método será procurado na hierarquia da classe instanciada pelo **Objeto1**;
- Devemos observar que, nesse caso, o **métodoX ()** que também é implementado na **Classe B,** foi encontrado primeiro que o **metodoX ()** da **Classe A**;
- Nesse caso, o método que de fato será executado será o da Classe B.
- E nesse caso, o **metodoX ()** da **Classe A** jamais será alcançado, a menos que seja criado um objeto da Classe A. Dizemos então que ocorreu uma **SOBREPOSIÇÃO DE MÉTODO ou um POLIMORFISMO DE SOBREPOSIÇÃO**;
- O **método X** da **classe B** sobrepôs (ou redefiniu) o método de sua classe mãe.

POLIMORFISMO DE SOBREPOSIÇÃO em java

- Considere que a classe **Brinquedo** (usada no exercício anterior) possui como descendentes as classes:
  - **Carro**;
  - **Avião;**
  - **Barco**.
- Conforme ilustra o slide a seguir.

POLIMORFISMO DE SOBREPOSIÇÃO



POLIMORFISMO DE SOBREPOSIÇÃO

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

- ✓ Vejamos então como ficam essas classes codificadas em Java:

POLIMORFISMO DE SOBREPOSIÇÃO

```
public class Brinquedo{
        ...
public void mover() {
System.out.println("mover brinquedo");
}
        ...
}
```

```
public class Carro extends Brinquedo{
        ...
public void mover() {
        System.out.println("CORRER");
}
        ...
}
```

```
public class Aviao extends Brinquedo{
        ...
public void mover() {
        System.out.println("VOAR");
}
        ...
}
```

```
public class Barco extends Brinquedo{
        ...
public void mover() {
        System.out.println("NAVEGAR");
}
        ...
}
```

POLIMORFISMO DE SOBREPOSIÇÃO

- Considerando o método **mover()** de cada classe filha, como poderíamos chamar o método mover() do brinquedo (classe) correto?
- Ou seja, como o **Controle Remoto** saberá que método **mover** ele deve chamar se ele tem disponível três tipos de mover diferentes (um para cada brinquedo)?
- Vamos ver inicialmente como fica a implementação do Controle Remoto:

POLIMORFISMO DE SOBREPOSIÇÃO

```
public class ControleRemoto{
private Brinquedo brinquedo;

public ControleRemoto(Brinquedo b) {
        brinquedo = b;
    }

public void mover() {
        brinquedo.mover();
    }
}
```

POLIMORFISMO DE SOBREPOSIÇÃO

- Você lembra que na primeira vez que apresentamos esse exemplo dissemos que a única restrição para um controle remoto tão versátil seria que “quando criado o controle remoto, ele receberia o tipo de brinquedo que iria acionar em um dado instante”?
- Pois é, é aí que está o segredo para o controle remoto saber qual deve ser o método **mover()** que ele deve chamar (CORRER, NAVEGAR ou VOAR);
- O método construtor da classe **ControleRemoto** exige que o controle para ser inicializado receba um parâmetro do tipo **Brinquedo**;

POLIMORFISMO DE SOBREPOSIÇÃO

- E isso acontece quando o atributo **brinquedo** do **ControleRemoto** recebe "**b**"(um objeto do tipo **Brinquedo**);
- Então, quando o método **mover()** da classe **ControleRemoto** for acionado (isso significa dizer que o botão mover foi apertado), esse faz com que seu atributo **brinquedo** chame o método **mover()** correto, dependendo do tipo de **Brinquedo** que recebeu quando foi instanciado;
- O próximo exemplo mostra como fica a classe Principal que instancia um objeto **controleRemoto** e um objeto **carro** para ser controlado automaticamente.

POLIMORFISMO DE SOBREPOSIÇÃO

```
public class Principal {

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

POLIMORFISMO DE SOBREPOSIÇÃO

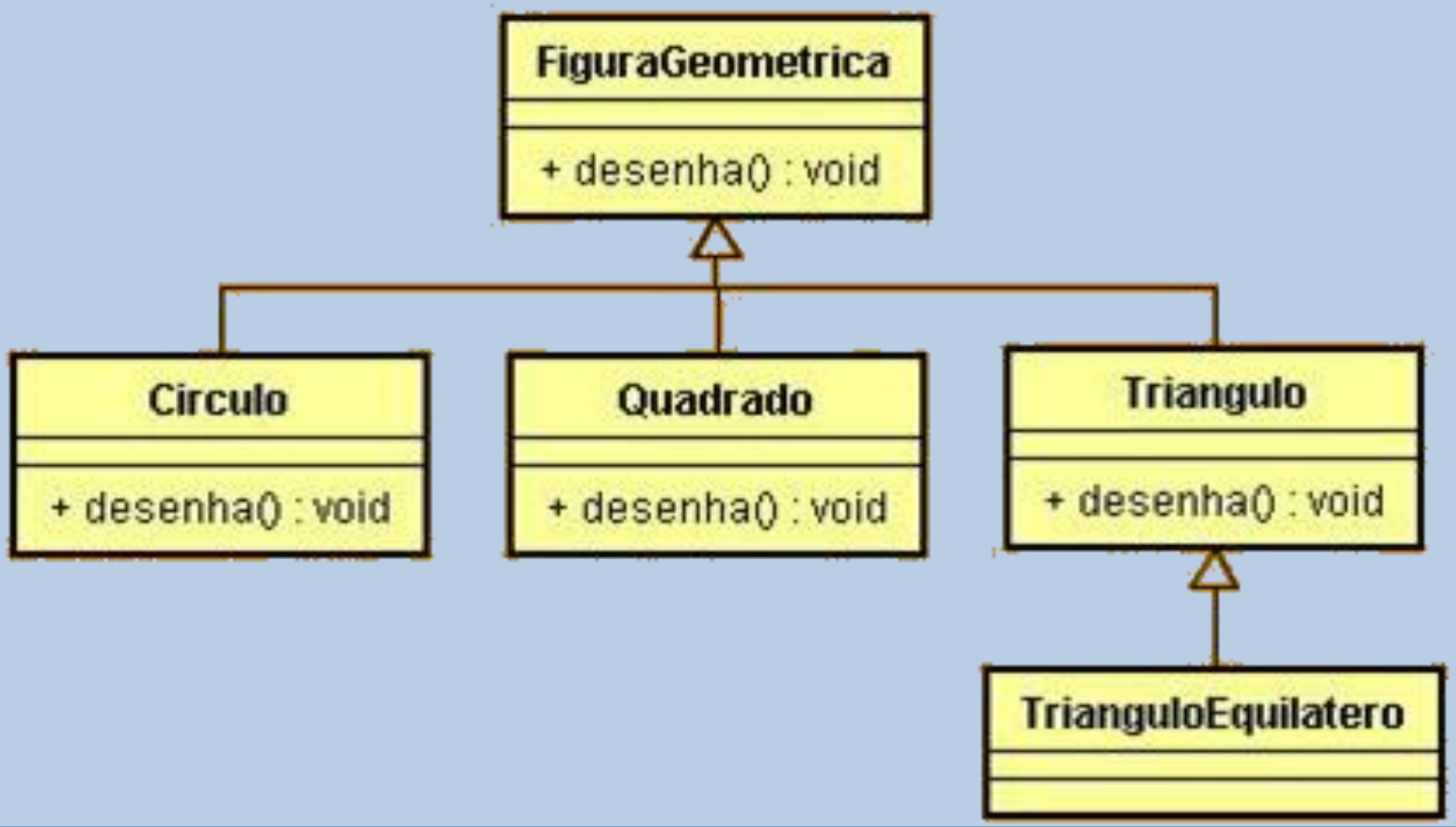
- Observe que criamos um brinquedo do tipo Carro, e quando criamos o **ControleRemoto,** enviamos esse objeto carro para o objeto **controleRemoto** através da chamada ao seu construtor.
- Assim, quando acionado o comando **controleRemoto.mover(),** será chamado o método **mover()** do **carro**.
- O resultado será a impressão da palavra:

CORRER

POLIMORFISMO DE SOBREPOSIÇÃO - ATIVIDADE

- Implemente as classes da hierarquia da classe **FiguraGeometrica** mostrada no próximo slide em Java, aplicando o polimorfismo de sobreposição para o **métododesenha()**.
- Em seguida, crie uma classe Principal com um método main que cria um objeto de cada uma das classes e chama seus respectivos **métodosdesenha().**

POLIMORFISMO DE SOBREPOSIÇÃO - ATIVIDADE



TIPOS DE POLIMORFISMO

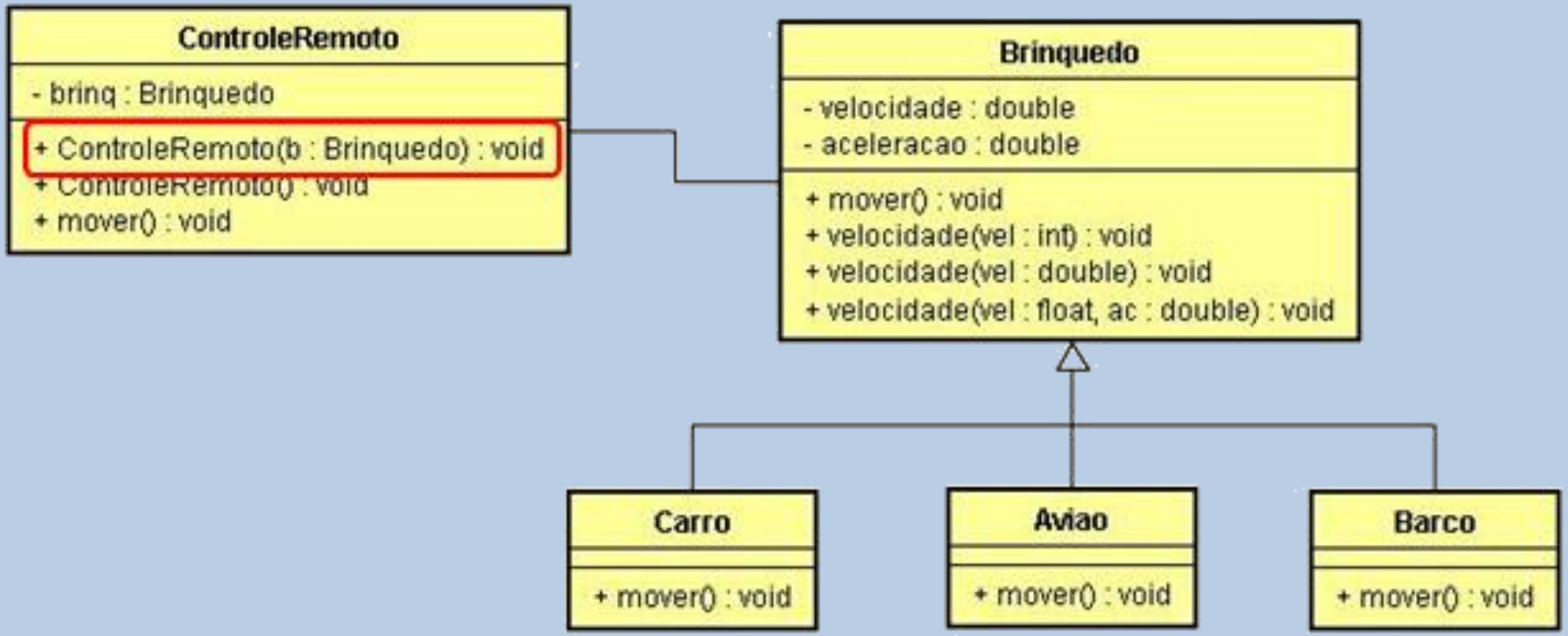
- Polimorfismo de Sobrecarga;
- Polimorfismo de Sobreposição;
- **Polimorfismo de Inclusão.**

Prof. Gunther Graf
gunthergrafpr@gmail.com

POLIMORFISMO DE INCLUSÃO

- Polimorfismo de inclusão usa a capacidade de substituição da Herança, de uma classe mãe por qualquer classe descendente, para permitir um comportamento polimórfico nos métodos que usam a classe mãe.
- No exemplo visto na seção anterior, onde criamos um objeto do tipo **Carro** e outro do tipo **ControleRemoto**, nós utilizamos o comportamento polimórfico do polimorfismo de inclusão.
- Fizemos isso quando substituímos a classe **Brinquedo** (mãe) pela classe **Carro** (filha) dentro da classe **ControleRemoto**.
- Assim, o atributo interno do tipo Brinquedo da classe **ControleRemoto** pode receber qualquer objeto que seja de uma classe filha de Brinquedo, vejamos o exemplo:

POLIMORFISMO DE INCLUSÃO



POLIMORFISMO DE INCLUSÃO

- Observe 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 descendentes de **Brinquedo,** elas podem substituir a classe **Brinquedo** em qualquer método que a utilize.
- Nesse caso, isso foi feito explicitamente, através da passagem de um objeto da classe Carro para o método construtor de **ControleRemoto**.
- Caso o programador deseje mudar o controle remoto para interagir com algum outro tipo de brinquedo, bastaria passar um objeto da classe Avião ou Barco na chamada ao construtor da classe ControleRemoto.
- A capacidade do objeto (brinq) do tipo Brinquedo da classe **ControleRemoto** de receber qualquer um objeto de subclasses da classe Brinquedo é que caracteriza o **polimorfismo de inclusão**.

POLIMORFISMO – LEITURA COMPLEMENTAR

- http://pt.wikipedia.org/wiki/Polimorfismo
- http://www.dsc.ufcg.edu.br/~jacques/cursos/p2/html/oo/o_que_e_polimorfismo.htm
- http://www.caelum.com.br/apostila-java-orientacao-objetos/heranca-reescrita-e-polimorfismo/

POLIMORFISMO - RESUMO

- Você aprendeu que, com o polimorfismo, objetos de tipos diferentes podem responder à mesma mensagem (solicitação de método com o mesmo nome) de maneiras diferentes. Você estudou quais são e como funcionam os tipos de polimorfismo desde seus conceitos até sua codificação na linguagem Java. Você viu também diferentes exemplos para ilustrar as diferentes situações na qual o polimorfismo pode ser usado.

That's all Folks

Prof. Gunther Graf
gunthergrafpr@gmail.com