**RESUMO DA AULA 07 - POLIMORFISMO**

ALUNO: Denilson José do Bom Jesus Silva de Lima

**Polimorfismo**

Semântica similar com implementação diferente, útil para reuso de código e generalização de rotinas (a capacidade de enxergar um objeto como um dos supertipos e a implementação específica no subtipo de características que estão definidas no supertipo).

Polimorfismo "pobre" - sobrecarga;

Polimorfismo clássico - sobrescrita;

Polimorfismo avançado - sobrescrita com "contrato".

Polimorfismo ad hoc: Sobrecarga - métodos com o mesmo nome com parâmetros (assinaturas) diferentes; independe de Herança.

Conhecido como "overload", trabalha com a ligação estática nas chamadas (em tempo de execução - dynamic binding - é possível saber qual método está sendo chamado, na compilação o compilador associa o novo construtor ao Pai) e é identificado pelos tipos dos valores passados como parâmetros. Métodos sobrecarregados são considerados métodos diferentes. (Ex:)

public void creditar(double valor) { ... }

public void creditar(double valor, double taxa){ ... }

É possível haver sobrecargas de construtores, neste caso, um construtor pode chamar outro construtor sobrecarregado usando o comando "this". (Ex:)

public Conta(int numeros){

this.numero = numero; }

protect Conta(int numero, duble saldo){

this(numero); }

Polimorfismo do subtipo: Sobrescrita - Quando uma classe Filha redefine um método na classe Pai.

Conhecido como "override"; pode existir métodos com o mesmo nome na classe Pai e na classe Filha, porém isso não caracteriza Sobrescrita se eles estiverem com

parâmetros diferentes (caracteriza sobrecarga), a Sobrescrita é caracterizada pelo mesmo nome de método nas duas classes e mesmos parâmetros, porém na classe Filha chama-se atributos que executam coisas diferentes dos presentes no método de mesmo nome na classe Pai, assim ficando disponível o uso pela classe Filha somente do novo método declarado na classe Filha.

Redefinições de métodos devem preservar o comportamento, a semântica, do método original.

É possível o uso do método da classe Pai sobrescrito criando-se um novo método na classe Filha que chama o método da classe Pai. (Ex:)

public class Pai {

public void creditar(double valor){

this.saldo = valor\*0.99;

}

public class Filha extends Pai {

//sobrescrevendo

public void creditar(double valor){

super.creditar(valor\*0.99);

}

//chamando da classe Pai

public void creditarPai(double valor){

super.creditar(valor);

}

}

A decisão de qual método chamar é feita em tempo de execução (dynamic binding) e a chamada do método aponta para a versão implementada no tipo do objeto apontado pela variável. Caso o tipo do objeto não tenha uma implementação para o método, este é sucessivamente procurado nas superclasses, de baixo para cima.

É possível coibir um método de ser sobrescrito se declará-lo com o modificador final. (Ex:)

public final void creditar(double valor) { ... }