MECANISMOS OOP EM JAVA

Gelson N. António / Nº: 1000023700

Faculdade de Engenharia

Universidade Católica de Angola

E-mail: gelsonantonio941@gmail.com

Abstract- Uma classe abstrata em Java define atributos e métodos.

Numa classe abstrata, um método pode ser definido com o modificador 'abstract'. Nesse caso, a classe abstrata não implementa os métodos abstratos.

As classes derivadas devem implementar os métodos abstratos.

Keywords – » public class, bonificação, String, AbstractSetä, classe, AWT (Abstract Window Toolkit), »

I. Introdução

Hoje, como a demanda por softwares novo e mais poderoso está aumentando, construir softwares de maneira rápida, correta e econômica continua a ser um objetivo indefinido. Objetos ou, mais precisamente, as classes de onde os objetos vêm são essencialmente componentes reutilizáveis de software. Há objetos data, objetos data/hora, objetos áudio, objetos vídeo, objetos automóvel, objetos pessoas etc. Quase qualquer substantivo pode ser razoavelmente representado como um objeto de software em termos dos atributos (por exemplo, nome, cor e tamanho) e comportamentos (por exemplo, calcular, mover e comunicar).

Grupos de desenvolvimento de software podem usar uma abordagem modular de projeto e implementação orientados a objetos para que sejam muito mais produtivos do que com as técnicas anteriormente populares como "programação estruturada" programas orientados a objetos são muitas vezes mais fáceis de entender, corrigir e modificar.

II. Conceitos

Hoje, como a demanda por softwares novo e mais poderoso está aumentando, construir softwares de maneira rápida, correta e econômica continua a ser um objetivo indefinido. Objetos ou, mais precisamente, as classes de onde os objetos vêm são essencialmente componentes reutilizáveis de software. Há objetos data, objetos data/hora, objetos áudio, objetos vídeo, objetos automóvel, objetos pessoas etc. Quase qualquer substantivo pode ser razoavelmente representado como um objeto de software em termos dos atributos (por exemplo, nome, cor e tamanho) e comportamentos (por exemplo, calcular, mover e comunicar).

- ### Encapsulamento

Classes (e seus objetos) encapsulam, isto é, contêm seus atributos e métodos. Os atributos e métodos de uma classe (e de seu objeto) estão intimamente relacionados. Os objetos podem se comunicar entre si, mas eles em geral não sabem como outros objetos são implementados os detalhes de implementação permanecem ocultos dentro dos próprios objetos. Esse ocultamento de informações, como veremos, é crucial à boa engenharia de software.

- ### Herança

É a capacidade de redefinir um método com mesma assinatura em sua subclasse

Tomemos como exemplo a classe Funcionário e incluir um novo método **bonificação**. Esse método representa uma bonificação que todos os funcionários recebem no fim do ano e é referente a 10% do valor do salário. Porém, o gerente recebe uma bonificação de 15%.

Quando falamos sobre a herança, temos que ter em conta que a herança tem:

- Capacidade de reutilização de software.
- Criar uma nova classe a partir de uma classe existente: absorvendo os dados e comportamentos da classe existente; e aprimorando-a com novas capacidades.

A subclasse estende a superclasse.

Subclasse:
Grupo mais especializado de objetos.

Comportamentos herdados da superclasse:
Podem se personalizar. Comportamentos adicionais.

- **Interfaces**

O Java também suporta interfaces coleções de métodos relacionados que normalmente permitem informar aos objetos o que fazer, mas não como fazer (veremos uma exceção a isso no Java SE 8). Na analogia do carro, uma interface das capacidades "básicas de dirigir" consistindo em um volante, um pedal de acelerador e um pedal de freio permitiria que um motorista informasse ao carro o que fazer. Depois que você sabe como usar essa interface para virar, acelerar e frear, você pode dirigir muitos tipos de carro, embora os fabricantes possam implementar esses sistemas de forma diferente.

Uma classe implementa zero ou mais interfaces cada uma das quais pode ter um ou mais métodos, assim como um carro implementa interfaces separadas para as funções básicas de dirigir, controlar o rádio, controlar os sistemas de aquecimento, ar-condicionado e afins.

Da mesma forma que os fabricantes de automóveis implementam os recursos de forma distinta, classes podem implementar métodos de uma interface de maneira diferente. Por exemplo, um sistema de software pode incluir uma interface de "backup" que ofereça os métodos save e restore. As classes podem implementar esses métodos de modo diferente, dependendo dos tipos de formato em que é feito o backup, como programas, textos, áudios, vídeos etc., além dos tipos de dispositivo em que esses itens serão armazenados.
Uma classe abstrata, além de definir métodos abstratos, pode implementar alguns métodos.

- Uma interface define apenas um conjunto de métodos abstratos.
- Uma classe pode implementar mais de uma interface.
- A implementação de uma ou mais interfaces não exclui a possibilidade de herança.
- O conceito de polimorfismo também é aplicável às interfaces implementadas por uma classe.

- **Polimorfismo**

Polimorfismo é a capacidade de um objeto poder ser referenciado de várias formas, ou seja, uma subclasse pode redefinir (sobrescrever) um método herdado.

Na herança, vimos que Gerente é um Funcionário, pois é uma extensão deste podemos referenciar a um Gerente como sendo um funcionário. Se alguém precisa falar com um funcionário do banco, pode falar com um Gerente! Por que? Pois Gerente é um funcionário. Essa é a ideia da herança. Polimorfismo não quer dizer que o objeto fica se transformando. Muito pelo contrário, um objeto nasce de um tipo e morre daquele tipo, o que muda é a maneira como nos referenciamos a ele.

Referências bibliográficas

[1] Caelum - Java e Orientação a Objetos
[2] Métropole Digital – Programação Orientada a Objetos:
http://www.metropoledigital.ufrn.br/aulas/disciplinas/poo/aula_10.html
[3] Leonardo Gresta Paulino Murta
leomurta@ic.uff.br
[4] Deitel®.
Java: como programar, 10a edição