Paradigma de Orientação à Objetos

<< Osvaldo Filipe \ Nº 1000022084 >>

Faculdade de Engenharia

Universidade Católica de Angola

<< osvaldofilipe369@gmail.com >>

**Neste respectivo trabalho vou abordar sobre os principais pilares dos paradigma de orientação à objetos que são: Encapsulamento, Herança, Abstração, Interfaces, Polimorfismo.**

**«Poli = muitas, Morfo = formas, get, set, is»**

INTRODUÇÃO

Linguagens orientadas a procedimentos, que foi o paradigma de programação q mais se desenvolveu em 1970, possuem como foco principais os subprogramas e bibliotecas de subprogramas. Neste caso, dados são enviados para serem processados nos subprogramas.

CONCEITOS

**Encapsulamento**

O encapsulamento tem a função de proteger as suas variáveis, criando um campo de força (barreira). A maioria dos valores que utilizamos dentro das classes são codificados com algumas definições de limite. O importante no encapsulamento é quando você cria esses métodos para acessar a sua variável, a idéia é que ao criar esses métodos pode mudar de idéias. Eu pude entender em encapsulamento que quando estamos criando um método para captura de informação utilizamos o prefixo “GET” e quando estamos a criar um de configuração utilizamos um prefixo “SET”, mas para variáveis booleanas o método de captura começa com o prefixo “Is”. Concluindo o meu resumo neste parágrafo eu pude concluir que o encapsulamento não é obrigatório, mas é uma boa pratica para produzir classes mais eficientes e tem como vantagens tornar mudanças invisíveis.

**Herança**

A herança tem a função de permitir basear uma nova classe na definição de uma outra classe previamente existente ou seja herdar características de outra classe mãe. Ela será aplicada tanto para as características quanto para os comportamentos. As classes filhas também podem ser chamadas de subclasses que por sua vez são classes oriundas das suas superclasses e para fazer uma subclasse não é necessário detalhar os atributos e comportamentos que vêm da sua super classe (isto já é automático).

**Abstração**

As classes abstratas elas podem ter métodos abstratos mais também podem implementar, alguns métodos,

**Interfaces**

A interface pode ser definida como uma lista de serviços fornecidos por um componente, ou seja é o contato com o mundo exterior que define o que pode ser feito com um objeto dessa classe. A mesma possui varias características, assim como:

* Elas são utilizadas para padronizar as interações das suas aplicações, definem e padronizam como coisas, pessoas e sistemas interagem entre si (os controles em um rádio servem como uma interface entre os usuários do rádio e os seus componentes internos da rádio);
* Só é composta de métodos e não tem atributos;
* Todos os métodos de uma interface são abstratos;

**Polimorfismo**

Polimorfismo permite que um mesmo nome representa vários comportamentos diferentes. Neste meu presente resumo vou realçar sobre dois tipos de polimorfismo:

* **Sobreposição**;
* **Sobrecarga**;

**Sobreposição**: Acontece quando “substituímos” um método de uma superclasse na sua subclasse, usando a “mesma assinatura”. Ou seja, é quando conseguimos substituir um método que veio da classe mãe diretamente na sua classe filha (mesma assinatura e classes diferentes).

**Sobrecarga:** Acontece quando usamos “assinaturas diferentes” e mesma classe.