**Princípios básicos da orientação a objectos**

Autor: António Nhanga Damião. ID: 100013052

Universidade católica de angola

Faculdade de engenharia

E-mail: smslemos@hotmail.com

- Para haver interacção entre dois objectos é necessoario apenas que, um conheça as operações disponíveis do outro (por meio da **Interface** ) para poder enviar e receber informações

* **Objecto:** tudo que pode representar um dado elemento do mundo real.
* **Classe**: conjunto de Objectos com características comuns

**1.3 –Herança.**

Em POO, herança é um mecanismo pelo qual uma classe obtem as **característica** e **métodos** de outra para expandi-la ou especializá-la de alguma forma. Isso significa que uma classe pode herdar as características métodos e atributos de outras classes, tal como acontece na genética dos animais.

## **Herdeira** H herda os métodos e atributos de A **Ancestral**

## Em POO, a herança pode ser :

- **Simples :** quando uma classe herda as características apenas de uma super classe.

**Multiplas:** quando se herda as características de duas ou mais classes

**1.5-** **Polimorfismo:** capacidade que diferentes objectos têm para responder a uma mesma mensagem . Exemplo prático: uma filmadora e um computador; ambos gravam, mas de diferentes modos. Em POO, polimorfismo ocorre quando um método já definido no ancestral é redefinido no herdeiro com um comportamento diferente. Essa característica do polimorfismo é importante pois, liga directamente as classes.

#### **1.5 Interface:**

#### É um ambiente estabelecido como modelo, para permite que classes que não pertencem a uma mesma hierarquia sejam tratados como iguais.

Em POO, este ambiente é fundamental pois, permite a reutilização de métodos, estabelecendo um contrato que define o comportamento de uma classe.

## **Bibliografia:**

## **- COAD**, Piter e **YOURDON** Edward, **analise baseada em objectos**, Rio de Janeiro Campos, 1999.

## - **RICART**, Ivan Luis Marques. Programação Orientada a Objectos: uma abordagem com o java. Campinas 2001.

## - **TOFNER** Marcon e **CORREIA**, Carlos. **Manual de análise orietada a objectos.** Blumenau – SC.

#### **Palavas chaves: “**Abstracção, Encapsulamento, Herança, Polimorfimo e interfaces**”.**

1. **Introdução**

Antigamente, as técnicas oferecidas para o desenvolvimento de softwares não atendia as necessidades desejadas para a concepção de tais aplicações. Houve então a necessidade de se adotar uma nova técnica de programar: “**programação orientada a objectos**”

Actualmente os softwares são desenvolvidos para obter uma interface com o usuário pelo uso da interface gráfico. Foi necessário criar uma técnica que se articulasse a velocidade de mudanças de hardware, interacção entre sistemas, etc, proporcionando um ambiente intuitivo e interactivo.

#### **Principios básicos da orientada a objectos**

#### A Programação orientada a objectos abarca uma serie de conceitos tais como: **Abstracção, Encapsulamento Herança , Polimorfimo e Interfaces .**

**1.1 -Abstracção**: (significa separar mentalmente um ou mais elementos de uma totalidade complexa)

Na POO, muitas vezes nos são apresentados problemas extremamente complexos que para a sua resolução precisamos dividi-lo em pequenos pedaços, isso quer dizer resolve-se o problema pedaço a pedaço ( de um problema maior para um problema menor). Isto implica que durante a resolução de cada pedaço do problema, abstrai-se o que não será usado de momento.

***“ O principio da abstracção consiste em isolar o objecto que se quer representar, do ambiente complexo a que se situa ” .*** *Abstair 🡪 ocultar mentalmente.*

* 1. **Encapsulamento:** consiste na ideia de que cada componente de um programa deve agregar toda informação relevante para a sua manipulação; isto é , cada componente deve ser visto como uma cápsula de informação.

- Em POO, usas-se o mecanismo de encapsulamento para ocultar detalhes de uma extrutura complexa que podem interferir durante a sua análise.

- A vantagem desse mecanismo consiste em disponibilizar o objecto com toda sua funcionalidade sem a necessidade de sabermos como esse objecto funciona.

- Outro sim, permite a modificação interna de um objecto, como, acrescentar métodos sem afectar os outros componentes do sistema que usam este mesmo objecto.

# Classe X

(herdeira)

#### .