**MODELAGEM DE SISTEMAS COMPUTACIONAIS**

**RESUMO: FUNDAMENTOS DO DESENHO ORIENTADO A OBJETO COM UML**

**ALUNOS: GUSTAVO FURINI E LEONARDO NERVINO**

**ENCAPSULAMENTO:**

O encapsulamento serve para o agrupamento de ideias em um escopo, o objetivo é deixar o software com mais flexibilidade assim sendo mais fácil de mudar o código e criar novas implementações. O encapsulamento orientado a objeto é mais sofisticado pois possui operações e atributos que representa o estado em um tipo de objeto, no qual fica mais acessível e modificável.

**OCULTAÇÃO DE INFORMAÇÕES E IMPLEMENTAÇÕES:** Você pode escolher quais informações deseja mostrar ou ocultar da visão pública, como códigos da implementação ou informações privadas de softwares. Normalmente as ocultações são de implementações e geralmente as informações são reveladas, como de costume na orientação a objeto.

**RETENÇÃO DE ESTADO:** Quando um módulo de procedimento é requisitado ele volta ao seu estado sem quaisquer mudanças, depois entra em estado ocioso deixando apenas o seu resultado, quando é chamando novamente nada do que foi feito anteriormente ira afetar no próximo procedimento

**IDENTIDADE DE OBJETO:** Cada objeto possui uma identidade pelo qual é identificado e tratado de forma distinta. O objeto também tem algo singular que é fornecido pelo identificador, que possuem algumas regras, a quais são que o mesmo identificador não pode referenciar a dois objetos e também ele permanece com o objeto para sempre.

**MENSAGENS:** Por meio das mensagens que um objeto se comunica com o outro e solicita a execução das atividades e seus métodos.

**- Estruturas de mensagens:** O objeto que recebe a mensagem armazena o identificador do objeto que o solicitou.

**- Argumentos de mensagens:** Na orientação a objeto os argumentos são identificadores de objetos, assim sendo objetos.

**- Os papéis dos objetos em mensagens:** Em um ambiente de pura orientação a objeto, os objetos exercem quatro tarefas e duas dessas são as de serem os destinatários e os remetentes das mensagens.

**- Tipos de mensagem:** Existem três tipos de mensagens, a informativa, interrogativa e imperativa. A informativa tem como função atualizar os objetos caso eles estejam com as informações passadas, interrogativa tem o objetivo de mostrar a informação presente no objeto no momento e a imperativa realiza algo que ainda não ocorreu para si próprio, para outro objeto ou até mesmo para o sistema.

**Classes**: Uma classe é o estêncil a partir do qual são criados objetos. Todo objeto tem a mesma estrutura e comportamento da classe na qual ele teve origem. Se o objeto 'abc' pertence a classe X, dizemos que "abc é uma instância de X".

Existem duas diferenças entre objetos de mesma classe. Cada objeto tem um identificador diferente e, a determinada hora, cada objeto provavelmente terá um diferente estado, o que significa, valores diferentes armazenados em suas variáveis.

* Classe é o que você desenha e programa;
* Objeto é o que você cria.

**Herança:** A herança (de Y a partir de X) é a habilidade que uma classe Y tem implicitamente definida em cada um dos atributos e operações da classe X, como se esses atributos e operações tivessem sida definidos com base na própria classe Y. X é caracterizada como uma superclasse de Y. Logo, Y é caracterizada como uma subclasse de X.

A herança representa outro caminho muito importante no qual a orientação a objeto diverge das abordagens dos sistemas convencionais. Ela efetivamente permite que você construa de forma incrementada, de acordo com as instruções.

**Polimorfismo**: Polimorfismo é a habilidade pela qual uma única operação ou nome de atributo pode ser definido em mais de uma classe e assumir implementações diferentes em cada uma dessas classes.

Polimorfismo é a propriedade por meio da qual um atributo ou variavel pode apontar para objetos de diferentes classes em horas diferentes.

**Generalização:** Generalização é a construção de uma classe X de forma que uma ou mais das classes que ela utiliza internamente é fornecida somente em run-time.