UNIVERSIDADE FEDERAL DO CEARÁ
TÉCNICAS DE PROGRAMAÇÃO – ALUNO: PEDRO RICKSON.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Classe: É como uma blueprint, um molde para o objeto. É aqui que definimos todas as estruturas de nosso objeto.
Objeto: É a instância da classe. A classe em uso, em execução, servindo a um propósito no código orientado a objetos.
Método e Atributo: São estruturas internas de uma classe. Podemos dizer de maneira análoga que são como variáveis e funções de nossa classe.
Encapsulamento: Pilar da programação orientada a objetos que consiste em restringir o acesso a algum atributo ou método dentro de uma classe, normalmente com o intuito de proteger as estruturas internas da classe do acesso externo.
Visibilidade de atributos e métodos: É um dos recursos oferecidos pela linguagem java para lidar com o encapsulamento. Um método ou variável pode ser:
Public – Pode ser acessado fora da classe que a possui, fora do pacote e em herança;
Protected – Pode ser acessado dentro da mesma classe, dentro do mesmo pacote e em herança;
Private – Apenas pode ser acessado dentro da mesma classe;
Herança: Pilar da orientação a objetos que visa ter uma hierarquia de objetos, uns herdando estruturas dos outros. Normalmente chamamos a classe herdeira de classe filho ou subclasse e a classe que foi herdada de superclasse ou classe pai.
Polimorfismo: Pilar da orientação a objetos que nos permite ter múltiplas formas para as estruturas de nosso código, assim podendo alterá-lo de acordo com nossas próprias necessidades. Normalmente o polimorfismo é usado em conjunto com a Herança. Imagine o seguinte exemplo: Você possui uma classe **Carro** e está para implementar uma nova classe **Ferrari** que vai herdar a classe **Carro**. A classe **Carro** possui um método publico **acelerar()**, o que significa que com base no pilar da **Herança**, nossa classe **Ferrari** ira herdar o método **acelerar()**. Porém não é isso que queremos, pois nossa classe **Ferrari** deve ter uma regra de negócio diferente para acelerar (Verificar a saúde do motor, inspecionar válvulas, ou que quer que seja. Use a sua imaginação, o importante é entender que ambos os métodos tem implementações diferentes.), para implementar essa nova regra que ira ter efeito apenas sobre a classe **Ferrari**, nós podemos apenas sobrescrevê-la com uma nova lógica dentro de **Ferrari** e então a classe filha herdará tudo que a classe pai tem porém com a nova implementação de um método sobrescrito da classe pai.
Classes Abstratas: De maneira bem simples, são classes que não podem ser instanciadas e portanto apenas podem ser usadas como superclasses ou tipos.

Construtores: São como métodos inicializadores das estruturas de nosso objeto. Eles são responsáveis pelo primeiro estado atribuído durante a criação do objeto.
São chamados usando a palavra ***new nomeDaClasse()***.
Get e Set: São artifícios usados em muitas linguagens orientadas a objeto e tem como função ler(get) ou escrever(set) nas estruturas internas de classe, uma vez que estas devem estar protegidas pelo Encapsulamento faz se necessário a criação de dois métodos públicos (get e set) para acessar esses atributos.
Sobrecarga de Métodos: Uma das features da linguagem Java que nos permite atingir o polimorfismo. Consiste em ter mais de um método com o mesmo nome, mas com assinaturas diferentes (assinatura de um método: Sintaxe do nome do método + sequencia de tipos de parâmetros).
Sobrescrita de Métodos: Outra feature que também nos permite atingir o polimorfismo. Usamos ela no exemplo acima(**Carro** – **Ferrari**), consiste em reimplementar um método herdado usando o mesmo nome mas com corpo e lógica diferentes.
This: Referência a própria classe em que estamos. É usado principalmente quando temos variáveis e atributos de mesmos nomes e então usamos ***this.atributo*** para distingui-los.
Super: Referência a classe pai(superclass) permitindo passar argumentos que são usados por ela.
Final: De maneira bem simples, significa que aquela estrutura está em seu estado final. Tem um comportamento bem diferente a depender de onde é usado. Ele previne a herança se usado em classes, impedi a sobrescrita se usado em métodos e impede a alteração se usado em variáveis.
Ter, usar e ser: O objeto pode possuir uma estrutura, usá-la ou o mesmo ser ela. Esse é um exercício mental importante para abstração em programação orientada a objetos.