

## Questão 01 – Lista4 – Técnicas de Programação

João Victor da Costa Izabel

Uma classe em Java é o que define o tipo de um objeto, a fazemos colocando dentro do bloco de comando da classe atributos ou características e métodos ou comportamentos que esse objeto pode ter e desempenhar dentro de sua própria classe ou de outras;

Encapsulamento se refere ao escopo onde os atributos e métodos daquele objeto podem ser visíveis e acessados, o encapsulamento pode ser feito usando as palavras `public` (visível em todo o projeto), `protected` (visível apenas no mesmo pacote) e `private` (visível apenas dentro do próprio objeto) na declaração de variáveis e atributos. Entretanto, ainda é possível modificar os valores de variáveis `private` ou `protected` usando métodos públicos ou protegidos comumente chamados de métodos `get` (que retorna o valor da variável) e `set` (que atribui o valor do parâmetro repassado à variável), melhorando o controle de quais partes do código podem ler ou escrever as variáveis do objeto, bloqueando seu acesso para outras partes do código, relacionado a isso, também existe a possibilidade de inicializar as variáveis do objeto no momento em que ele é instanciado (criado), passando por parâmetro o valor dessas variáveis dentro de construtores do objeto, que geralmente são públicos e tem como característica que o código que estiver dentro de seu bloco de comando sempre será executado durante a criação do objeto;

Um objeto também pode herdar características de outros, que é feita usando a palavra reservada “`extends`”, ou seja, existem subclasses (objetos filhos) que são herdeiras de superclasses (objetos pai), dessa forma, os atributos e métodos da superclasse vão também estar presentes e ter o mesmo comportamento na subclasse, facilitando o trabalho do programador. Além disso, como a subclasse faz tudo que a superclasse faz, é possível fazer polimorfismo, que consiste em aceitar todas as classes filhas de uma classe pai caso seja o tipo do pai que esteja configurando a passagem por parâmetro, dessa forma podemos instituir a máxima: “Onde passa o pai, passa o filho”. Outro recurso importantíssimo da herança é poder mudar o comportamento do filho, ou seja, mudar seus métodos através da :

(1) Sobrescrita de métodos (que consiste em poder declarar um método com o mesmo nome do pai e que será executado em detrimento ao método da superclasse, alterando assim seu comportamento)

(2) Sobrecarga de métodos (onde podemos ter diversos métodos com mesmo nome mas com parâmetros de diferentes tipos ou com diferentes ordens. Outra possibilidade importante no desenvolvimento de programas orientado a objetos é a criação de classes abstratas, que é feito usando a palavra reservada “`abstract`” em sua declaração, dessa forma, não pode ser instanciada, mas deve conter atributos e métodos e ser configurada como superclasse de outras classes, se tornando assim um molde para os objetos filhos, estabelecendo padrões de variáveis e funcionamento que, assim como antes, podem ser modificados usando sobrescrita e sobrecarga além de também poder fazer uso de polimorfismo. Durante a programação em O.O., existem 3 palavras reservadas importantes:

1) `super` -> Usada dentro do construtor das classes filhas para chamar o construtor da classe pai (superclasse) e passar os parâmetros necessários para ele, parâmetros os quais devem ser

do mesmo tipo e estarem separados por vírgula na mesma ordem que o construtor da classe pai pede.

2) `this` -> É usada para referenciar o próprio objeto da classe, pois o “`this`” aponta ao endereço virtual de memória onde o objeto está salvo, dessa forma, é possível retorná-lo dentro de um método ou, sendo um uso mais comum, informar que a variável ou método se refere ao do próprio objeto.

3) `final` -> significa que a classe que é declarada com essa palavra reservada não poderá ser superclasse de nenhuma outra, sendo essa mais uma forma de aprimorar a segurança do código.

As relações que um objeto por ter com o outro podem ser:

1) `Ter` -> É quando um objeto é instanciado e salvo dentro de outro, ou seja, efetivamente a primeira classe toda tem todos os atributos e métodos disponíveis da segunda, podendo usá-la ao seu bem querer;

2) `Usar` -> É quando uma classe usa o método da outra e para isso geralmente instancia a outra classe de forma volátil, dentro do próprio método por exemplo, dessa forma, quando a rotina deste método acabar, o objeto será apagado da memória;

3) `Ser` -> `Ser`: É o conceito de herança, se um objeto X é pai de outro objeto Y, Y “é” X, habilitando todas as propriedades de herança comentadas anteriormente.