Java e Orientação a Objetos

O que é Java?

Java é uma linguagem de programação desenvolvida pela antiga Sun Microsystems, hoje pertencente a Oracle, para ser usada em qualquer plataforma – 'Crie uma vez, e rode em qualquer lugar' – No começo ela era usada para applets na web. Em Java, é compilado um código da sua aplicação chamado de bytecode, ele é lido pela Máquina Virtual Java (JVM) instalado em qualquer sistema operacional, assim, gerando sua aplicação com o pacote JRE (Java Runtime Environment).

JVM -> Java Virtual Machine – Usado pela ler os códigos em bytecode.

JRE -> Java Runtime Environment – Ambiente de execução Java, nele está presente a JVM e outras bibliotecas, todo o necessário para rodar uma aplicação.

JDK -> Java Development Kit – Ambiente de desenvolvimento Java, ele possui a JRE e outras ferramentas para desenvolver as aplicações.

Fundamentos Java

Variaveis -> Blocos na memória, onde determinado tipo é definido e somente valores do tipo determinado são permitidos.

Exemplos de tipos primitivos:

Int -> valor inteiro (0, 1, 5, 64, 231, 4355);

float -> numeros flutuantes ou com casas decimais (1.5, 3.8, 250.5, 0.2);

char -> caratere unico ('A', 'B', 'C', '6', '9', '-');

boolean -> verdadeiro ou falso (true, false);

Exemplo de Objetos:

String -> Sequencia de caracteres ("Pedro Kauã Silva dos Santos", "Hello, World!");

If e else

O if e o else são dois blocos de códigos, onde se pode definir uma condição em cada um. Caso a condição definida no bloco (if) for verdadeira, o primeiro bloco será executado, e caso exista um bloco (else) e a condição for falsa, o mesmo será executado.

Exemplo:

```
Int value = 5
If (value == 5){
        System.out.println("Condição Verdadeira");
} else {
```

```
        System.out.println("Condição Falsa");
}
```

No exemplo, a condição em que (value == 5) é verdadeira, pois a variável possui o valor 5, sendo assim, será exibido na tela: "Condição Verdadeira".

Loops

Em Java, os mais usados blocos para definir um loop são: While(Enquanto) e For(Para). Os loops são blocos onde a execução dos códigos se repete enquanto uma determinada condição for verdadeira.

```
while (A < B) {
        System.out.println("Condição ativa");
        A++;
}
```

Enquanto A for menor que B, a mensagem será exibida na tela, e será somado +1 para A.

```
for (int contador = 0; contador < 10; contador++){
        System.out.println("Condição ativa");
}
```

Para a 'condição' execute o bloco. (Enquanto o contado for menor que 10, execute o bloco e adicione +1 para o contador);

Orientação a Objetos

A linguagem Java é 100% orientada a objetos, até o simples System.out.println é moldado pelo paradigma. Com o paradigma OO, tudo pode ser criado e representado como objetos: ações, objetos, pessoas, animais, clima, espaço, etc. De forma resumida, os objetos são representados em classes, que possuem atributos e métodos.

Exemplo:

```
Classe(Objeto)  Caneta {
        Atributo  String corDaCaneta = "Transparente"
        Atributo String CorDaTinta = "Azul"
        Atributo boolean estaAberta = false

        Metodo escrever (){
                if (estaAberta == true) {
                        System.out.println("Escrevendo");
                }
        }
}
```

Aqui está um exemplo de como uma caneta pode ser moldada no Paradigma OO, ela possui atributos como a cor da caneta e a cor da tinta, e também, um método escrever, onde existe uma condição, caso a caneta esteja aberta, será exibida uma mensagem na tela.

Pilares da POO

Abstração -> A abstração em POO não é diferente da definição da palavra – É filtrar apenas o importante ou necessário para a formação do 'objeto' em seu código.

Encapsulamento -> Tornar privado partes importantes do código, como variáveis e métodos, para garantir a segurança e melhorar modificações e implementações. Dito isso, quase sempre são usados métodos para gerências atributos privados, chamados Getters and Setters, com eles a modificação e implementação fica muito mais segura, sem afetar outras partes do código.

Herança -> Como um filho que recebe traços da mãe, as classes mães, ou super classes, passam características, como seus métodos e atributos as classes filhas. Quando a classe precisar ser instanciada, a classe mãe pode servir como referência, mas nunca o contrário.

Polimorfismo -> O polimorfismo funciona de muitas maneiras, mas seu conceito é o mesmo nelas todas: O mesmo método com funções diferentes. Com o polimorfismo, os métodos podem iguais mas também distintos de certa maneira, como um exemplo de herança, onde uma classe filha recebe o método de uma classe mãe, mas seu método funciona de maneira diferente, assim, utilizando o @Override. Também existem situações onde o mesmo método é feito de diferentes maneiras em uma mesma classe, mudando apenas seu 'retorno' ou seus 'parâmetros'.

Classes Abstratas e Interfaces

As classes Abstratas são classes feitas apenas com o proposito de serem uma super classe, assim criando novas classes para herdarem suas características.

As interfaces são definidas ao nível de 'classe', no topo. As mesmas, não possuem valores em suas instâncias nem 'regras de negócio' ou qualquer código em seus métodos, são apenas para servir como uma arquitetura pré-moldada antes da classe, ou para servir como visão externa para os usuários, elas são implementas nas classes, assim, todos os métodos e atributos também são herdados (implementados).

Exceções e controle de erros

As exceções são representações de falhas no código em determinada situação, também definidas como objetos de exceções, como, por exemplo: quando uma variável nula é usada para cálculos ou outra interação, é lançada uma exceção padrão chamada 'NullPointerException'.

Em Java, o tratamento de erros de código são tratados por blocos de códigos chamados try-catch. 'Try' é o bloco onde é escrito o código que precisa de tratamento caso ocorra algum erro, já o bloco 'Catch' é usado para tratar o erro, também com código ou

lançando uma exerção para aviso ao Desenvolvedor. No bloco ‘Catch’ é passado a exceção que deve ser tratada, caso ocorra outra exceção, o bloco não será acionado, em casos de lançamento de exceções, o Java já possui várias em suas bibliotecas, com o comando ‘throw’ é possível lançar elas no console.

- ➔ Uma prática muito usada, lançar exceções propositais, e as tratar com outro bloco try-catch, assim tendo um melhor controle do código e dos erros.