O que é?

STATIC

É uma palavra chave da linguagem Java, isso significa que é uma palavra reservada que só pode ser utilizada pela própria linguagem.

O próprio método principal da linguagem Java utiliza a palavra static na sua assinatura. Isso garante que haverá somente uma chamada ao método main e haverá somente uma referência dela na memória.

Ao executar um programa java, o primeiro método que a JVM procura é o método main antes de criar um objeto da classe na qual esse método é declarado.

Como é aplicado e em quais elementos, como classes, atributos, métodos e blocos de inicialização?

É aplicado em classes, variáveis e métodos em combinação com outras palavras reservadas da linguagem java.

Ao usar static em variáveis, significa que podemos criar uma instância ou mais dessa classe, mas a variável vai sempre pertencer ao escopo de classe e não a instâncias criadas.

Ao utilizar static em importações podemos omitir o nome da classe quando chamamos seus métodos.

Podemos utilizar em classes que foram chamadas de classes aninhadas.

Podemos utilizar em métodos que podem ser chamadas diretamente pelo nome da classe, ponto e nome do método, sem mesmo criar instância dessa classe.

Pode ser aplicado em atributos, deixando esse atributo uma constante.

Pode ser aplicado em blocos de inicialização que são executados antes mesmo do construtor dessa classe.

Qual sua utilidade?

É útil quando queremos executar chamadas antes de qualquer outra chamada no nosso programa, quando queremos executar métodos que sejam utilizados independente de instância de classes, quando queremos criar certas regras como por exemplo o padrão de projeto singleton.

Limitações?

Eles não podem ser acessado usando as palavras super, this. E não podem acessar variáveis que não sejam estáticas. Também não podem ser utilizada em interface e é o alto acoplamento não permitindo injeção de dependências.

Exemplo de uso?

```java
import static java.lang.Math.sqrt;
public class Codigo {
    public static void main(String[] args) {
        //importação estatica
        double sqrt = sqrt(4);
        System.out.println(sqrt);
        Codigo2 codigo2 = new Codigo2();
        System.out.println(codigo2);
//        Classe interna estatica
        Codigo2.Codigo3 codigo3 = new Codigo2.Codigo3();
        codigo3.print();
        //Acessando metodo estatico
        Codigo2.Codigo3.print2();
    }
}
class Codigo2{
    //Atributo estatico
    private static int numero = 3;
    private int numero2 = numero;
    static {
        System.out.println("Bloco estatico: " + numero);
//        System.out.println("Bloco estatico: " + numero2); //Erro
    }
    public static class Codigo3 {
        public void print(){ System.out.println("Não estatica: " +numero); }
        public static void print2(){ System.out.println("Estatica: "+numero); }
    }}
```

FINAL

O que é?

FINAL

É um operador reservado pela linguagem java como palavra chave. Pode ser usado em métodos, atributos e classes. Pode ser usado para restringir acesso ou não não modificar certos dados.

Como é aplicado e em quais elementos, como classes, atributos, métodos e blocos de inicialização?

Se utilizar final em classes faz com que essa classe não seja superclasse de nenhuma outra, não podendo ser estendida.

Caso exista uma classe e a classe pai (não final) seja estendida como superclasse e houver algum método marcado como final, esse método não poderá ser sobrescrito pelas classes filhas.

Qual sua utilidade?

Pode-se utilizar para criar métodos mais seguros e que não receberam modificações. Podemos aplicar também à classes inteiras que são fechadas para modificações como exemplo da classe String.

Exemplo de uso?

```
//public class CodigoFinal {// ERRO - extends CodigoFinal2{

//public class CodigoFinal extends String{ //String é uma classe final, não pode ser estendida

public class CodigoFinal extends CodigoFinal3{

  //Não pode ser sobrescrito

//    @Override

//    public void print() {

//        super.print();

//    }


   void exibirNumeroClassePai() {

//        this.numero = 1; //final - Não pode ser atribuido nenhum outro valor
```

```
    }

    public static void main(String[] args) {
        CodigoFinal2 codigoFinal2 = new CodigoFinal2();
        System.out.println(codigoFinal2.metodoSecreto(0));
    }
}

final class CodigoFinal2{
    private final int numeroSecreto = 1;
    //Só pode ser alterado internamente
    public int metodoSecreto (int numero){
        return numero * numeroSecreto;
    }
}

class CodigoFinal3{
    public final int numero = 10;
    final void print(){
        System.out.println("chamando CodigoFinal3");
    }
    void print(final String textoExemplo){
        System.out.println(textoExemplo);
//        textoExemplo = ""; //Não pode ser modificado nesse escopo
    }
}
```