Aluno: Analu Sorbara

1 - Elabore um texto sobre os operadores static e final que responda às seguintes questões:

- O que é?

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

- Qual sua utilidade?

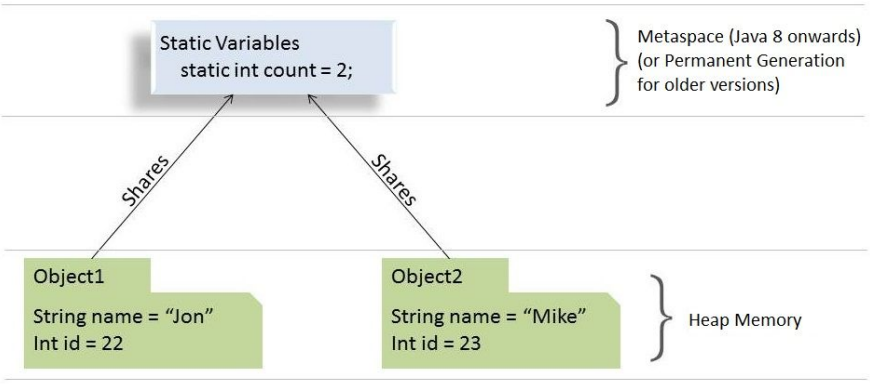
- Limitações?

- Exemplo de uso?

Resposta:

Na linguagem de programação Java, a palavra-chave static indica que o membro específico pertence a um tipo em si, em vez de a uma instância desse tipo.

Isso significa que apenas uma instância desse membro estático é criada, a qual é compartilhada por todas as instâncias da classe. A imagem a seguir ilustra como se comporta uma variável estática dentro de uma classe.



No Java, classes, parâmetros, métodos e blocos de código podem ser do tipo estático.

A principal utilidade de uma entidade estática é desvincular sua criação de uma instância de um objeto, portando, classes, parâmetros, métodos e blocos estáticos não precisam ser instanciados para serem utilizados. Por exemplo, variáveis estáticas são utilizadas para criar um design pattern Singleton em Java pois elas garantem que a classe somente possuirá uma única instância compartilhada por todo o código.

Umas das principais limitações do uso de entidades estáticas é a alocação de memória definitiva que não pode ser removida pelo coletor de lixo do java. Além disso, podem ser um risco para o programa em certas situações pois preservam seu estado até o final da execução do programa, um esquecimento de limpeza ou reinicialização da variável pode introduzir problemas no código.

O exemplo abaixo mostra uma criação do design pattern singleton com variáveis estáticas. Singleton são classes que se auto instanciam e não permitem que processos criem instâncias dela:

```
public final class Singleton {

    private static Singleton instance;

    private String algumValor;

    private Singleton() {

    }

    public static Singleton getInstance() {

        if (instance == null) {

            instance = new Singleton();

        }

        return instance;

    }

    public String getAlgumValor() {

        return algumValor;

    }

    public void setAlgumValor(String algumValor) {

        this.algumValor = algumValor;

    }

}

public class StaticTests

{
```

```
    public static void main(String[] args) {
        // obtém uma referência da instância da classe
        Singleton singleton = Singleton.getInstance();
        // obtém a mesma referência anterior
        Singleton singleton2 = Singleton.getInstance();
        // coloca algum valor na primeira referência
        singleton.setAlgumValor("teste");

        // le o mesmo valor mas na mesma referência
        System.out.println(singleton2.getAlgumValor());
    }
}
```

2 - Acerca dos assuntos INTERFACE, ABSTRACT e PACOTES em Java, elabore um texto que responda às seguintes questões:

- O que é?

- Como é aplicado?

- Qual sua utilidade?

- Limitações?

- Exemplo de uso?

Resposta:

Interface é uma abstração de classes em Java que permite definir um contrato o qual a classe que implementa a interface deve seguir. Esse contrato são métodos mínimos que a classe deve possuir e suas assinaturas.
Abstract é um tipo especial de classe que pode ser herdada mas nunca pode ser instanciada, todas as classes que herdam (extend) dessa classe possuirão todos os recursos que estão implementados na classe abstrata.
Pacotes é um conceito em java para agrupar uma ou mais classes, interfaces ou até mesmo outros pacotes com o mesmo mesmo propósito de funcionalidade. Esse recurso também

permite que em um único código coexistem classes ou interfaces de mesmo nome, funcionalidade semelhante ao namespace do C# ou C++.
Uma interface é aplicada para garantir que duas implementações de classes distintas possuam os mesmos métodos e assinaturas. Isso garante que uma parte do código que está esperando um determinado tipo de interface não precisa se preocupar como a interface foi implementada, podendo até mesmo, trocar alternar entre implementações sem qualquer tipo de incompatibilidade.
Classe abstrata pode ser utilizada para assegurar que não haja instância dessa em todo o código. Isso é desejável para casos onde uma classe carrega implementações que são úteis para todo o código mas ao mesmo tempo não possui qualquer utilidade quando instanciada sozinha.
Pacotes possuem várias aplicações, podem ser utilizados para isolar determinada funcionalidade, permitir que terceiros utilizem funcionalidade e até mesmo para permitir que duas classes com mesmo nome coexistem dentro de uma mesma aplicação.
Uma interface é limitada a possuir somente assinaturas, ela não pode carregar qualquer implementação de método, isso pode deixar o código extenso para alguns casos.
Classe abstrata possui a desvantagem de carregar para todas as classes que herdam dela implementações de métodos adicionais que muitas não são utilizados em determinados contextos.
Pacotes possuem a desvantagem de trazer muitas classes que não são utilizadas para determinado contexto, não sendo possível removê-las sem alterar o pacote.

```
package animal;

public interface Animal {
    String tipo();
}
```

```
package animal;

public abstract class AnimalImpl implements Animal {
    public String tipo() {
        return "mamífero";
    }
}
```

```
package animal;

public interface Gato extends Animal {
    String nome();
    int idade();
}
```

```
package animal;

public class Gato1Impl extends AnimalImpl implements Gato {
    @Override
    public String nome() {
```

```
        return "tito";
    }

    @Override
    public int idade() {
        return 7;
    }
}

package animal;

public class Gato2Impl extends AnimalImpl implements Gato {
    @Override
    public String nome() {
        return "fifo";
    }

    @Override
    public int idade() {
        return 4;
    }
}

// importações do pacote animal
import animal.Gato;
import animal.Gato1Impl;
import animal.Gato2Impl;

public class TesteIntAbsPacote {
    public static void main(String[] args) {
        Gato gato = new Gato1Impl();

        // imprime o nome do gato 1
        System.out.println(gato.nome());
        // possui o método da classe abstrata animal.Animal
        System.out.println(gato.tipo());

        // imprime o nome do gato 2
        // utilizando a mesma variável, ou seja
        // a mesma assinatura de interface
        gato = new Gato2Impl();
        System.out.println(gato.nome());
        // possui o método da classe abstrata animal.Animal também
        System.out.println(gato.tipo());

        // erro, uma classe abstrata não pode ser instanciada
//        Animal animal = new Animal();
    }
```

```
}
```