Regras de Escopo, utilizando atributos, métodos e variáveis.

Regras de Escopo

- Regras de Escopo dizem respeito ao limites de utilização de um determinado atributo, variável ou método.

- Ao declarar uma dessas estruturas citadas acima, existirá uma área, espaço ou local de onde está poderá ser utilizada utilizando-se o seu simplesmente o seu nome, e outra onde ela não poderá ser utilizada pelo seu nome, ou não poderá ser utilizada de forma alguma.

- A estes limites chamamos de escopo.

- **Ofuscamento** é o nome que damos quando uma variável ou atributo é ofuscada por outra de mesmo nome dentro de um escopo. O ofuscamento irá impedir que o compilador se confunda, ficando sem saber qual a variável que você está referenciando.

Regras de Escopo

- Atributos e variáveis são conhecidas apenas dentro de um determinado escopo.
- Um escopo em geral é um bloco de código, seja ele um bloco de código de uma classe, método, for, if, ou qualquer outra estrutura que precise declarar um bloco.
- Veremos o escopo de variáveis e atributos.

Regras de Escopo

- Escopo de Atributos
  - Atributos de uma classe são visíveis em toda a classe.
  - Podem ser utilizados dentro de qualquer método.
  - Um atributo da Classe será ofuscada por uma variável de mesmo nome dentro de um bloco em um método ou estrutura de controle, vamos ver...

```java
public class Retangulo {

    //poderei usar os dois atributos abaixo
    //em qualquer lugar da classe.
    private double base;
    private double altura;

    public double calculaArea() {
        //OPA!! Declaração de variáveis com o mesmo
        //nome dos atributos... Os atributos serão
        //ofuscados...
        double base;
        double altura;

        return base*altura;
    }

    public void setBaseEAltura(double base, double altura) {
        //OPA!! Mais uma vez temos variáveis com mesmo nome
        //de atributos, atributos serão ofuscados, o que irá
        //valer são as variáveis deste escopo.

        //??????? funciona? Claro que não!!!
        //a pergunta é: quem é quem?
        base = base;
        altura = altura;
    }
}
```

Vamos identificar os escopos...

```
public class Retangulo {

    //poderei usar os dois atributos abaixo
    //em qualquer lugar da classe.
    private double base;
    private double altura;

    public double calculaArea() {
        //OPA!! Declaração de variáveis com o mesmo
        //nome dos atributos... Os atributos serão
        //ofuscados...
        double base;
        double altura;

        return base*altura;
    }

    public void setBaseEAltura(double base, double altura) {
        //OPA!! Mais uma vez temos variáveis com mesmo nome
        //de atributos, atributos serão ofuscados, o que irá
        //valer são as variáveis deste escopo.

        //??????? funciona? Claro que não!!!
        //a pergunta é: quem é quem?
        base = base;
        altura = altura;
    }
}
```

Escopo da Classe. Em Amarelo.

Escopo do Método calcularArea( )

Escopo do Método setBaseEAltura( )

Ofuscamento de Atributo

- Não se pode evitar o ofuscamento de um atributo que teve o seu nome utilizado na declaração de uma variável em um bloco de código, seja este bloco pertencente a um método ou estruturas de controle como for, if e outros.

- Porém, ainda assim podemos utilizar a variável precedendo-a da palavra this, seguido de um ponto e posteriormente do nome do atributo. Veja o exemplo abaixo.

- ```
  this.altura = 3;
  ```

- Vamos ver como fica o nosso código utilizando a técnica acima para contornar o problema do ofuscamento.
  ```

```java
public class Retangulo {

    //poderei usar os dois atributos abixo
    //em qualquer lugar da classe.
    private double base;
    private double altura;

    public double calculaArea() {
        //OPA!! Declaração de variáveis com o mesmo
        //nome dos atributos... Sem problemas basta
        //usarmos a palavra this.
        double base;
        double altura;

        return this.base*this.altura;
    }

    public void setBaseEAltura(double base, double altura) {
        //OPA!! Mais uma vez temos variáveis com mesmo nome
        //de atributos, atributos serão ofuscados, porém
        //ainda poderemos usá-lo utilizando this

        //funciona? Sim!!!
        //agora sabemos quem é quem.
        this.base = base;
        this.altura = altura;
    }
}
```

Construtores

Construtores

- Construtor é o nome que se dá a um método especial de uma classe. Este método é responsável pela criação de um objeto.
- Até agora quando criamos um objeto o Java está utilizando o construtor padrão, que é criado automaticamente pelo java quando ele verifica que não programamos um construtor para a classe.
- Cada classe possui o seu próprio construtor.
- O construtor é importante para que façamos com que o objeto inicie do jeito que desejamos, colocando por exemplos alguns valores em seus atributos.

Construtores

- Um construtor é um método que possui o mesmo nome da Classe.
- Este método pode receber argumentos ou não.
- Vamos ver um exemplo.

```java
public class Pessoa {

    private String cpf;
    private String nome;
    private Date dataDeNascimento;
    private String endereco;
    private String telefone;

    public Pessoa(String cpf, String nome, String endereco) {
      this.cpf = cpf;
      this.nome = nome;
      this.endereco = endereco;
    }

    //restante do código aqui...
}
```

Construtores

- Agora podemos criar um objeto do tipo pessoa utilizando o construtor que definimos.
- Para isso criamos fazemos:
- ```
  Pessoa p1 = new Pessoa("009234235", "Mathias", "R. XYZ");
  ```
- Ao ser criado o objeto irá executar o código do nosso construtor definindo assim os valores de cpf, nome e endereço.
- Ao começar a utilizar o objeto iremos verificar que ele desde a sua construção já possui preenchido cpf, nome e endereço.
  ```

Construtores

- Notem que os construtores diferente dos outros métodos não possui retorno.

- Note também que ele pode ter ou não argumento, poderíamos ter o seguinte:

```
public Pessoa() {

  this.cpf =  "indefinido";
  this.nome = "sem nome";
  this.endereco = "sem endereço";

 }
```

Construtores

- Podemos ter mais de um construtor para a nossa classe.
- Porém cada um deles deve ser diferente um do outro.
- Isso significa que o tipo e a ordem dos argumentos devem ser diferentes de um construtor para outro.

```java
public class Pessoa {

    private String cpf;
    private String nome;
    private Date dataDeNascimento;
    private String endereco;
    private String telefone;

    public Pessoa(String cpf, String nome, String endereco) {
      this.cpf = cpf;
      this.nome = nome;
      this.endereco = endereco;
    }

   public Pessoa() {

      this.cpf =  "indefinido";
      this.nome = "sem nome";
      this.endereco = "sem endereço";

    }

    //restante do código aqui...
```

Exercício

- Defina os construtores para as classes de formas que estamos desenvolvendo para que elas sejam criadas com valores padrões ao invés de 0.