Modificadores de Acesso

Modificadores de Acesso

- A maioria das linguagem orientadas a objetos possuem o conceito de controle de acesso aos membros de uma classe.
  - Python é um exemplo de linguagem OO que não possui este conceito implementado.
- Os modificadores indicam se um membro pode ser acessado diretamente via o operador (.) ou não.
- Na maior parte das linguagens são 3 os modificadores de acesso, o **public**, o **protected** e o **private**.

Modificadores de Acesso - Public

- Se você declara um atributo ou um método utilizando o modificador de acesso public, você estará:
  - permitindo que este membro seja acessado dentro do escopo onde o objeto desta classe foi declarado, bastando para isso utilizar o operador (.).
- Vamos ver na prática…

Modificadores de Acesso - Public

```
public class Retangulo {
        public float base;
        public float altura;

        public float calcularArea() {
                return base*altura;
        }
}
```

```
public class GerenciadorDeFormas
{
    private Retangulo retangulo = new Retangulo();

    public GerenciadorDeFormas()
    {
        retangulo.base = 3;
        retangulo.altura = 4;
    }
}
```

Um objeto do tipo ***Retangulo*** declarado no escopo do ***GerenciadorDeFormas*** permite acesso direto aos atributos públicos ***base, altura*** e ***calcularArea*** utilizando o .

Modificadores de Acesso - Public

- Não se recomenda definir atributos como sendo públicos por se entender que ele quebra o encapsulamento dos dados.
- Caso você permita que outras classes manipulem os atributos diretamente, fica complicado em caso de erro identificar a origem da mudança.
- Portando, entende-se que, o valor de um atributo deve ser alterado dentro da classe que a define.
- Porém caso não exista uma lógica específica a ser executada durante a mudança do valor a mudança direta e a indireta possuem o mesmo efeito de quebra do encapsulamento.

Modificadores de Acesso - Public

- Ainda assim, é importante manter encapsulado o acesso. Mas porque?
- Se eventualmente alguma operação necessite ser feita no futuro, como por exemplo:
  - Tratamento de concorrência de acesso à variável.
  - Um eventual checagem dos dados antes da mudança.
  - etc.
- Todo as outras classes já usam os métodos ***get*** e ***set*** e assim basta acrescentar a lógica necessária em apenas um local.

Modificadores de Acesso - Private

- Para restringir o acesso aos atributos de um objeto, declaramos estes atributos com o modificador de acesso ***private***.
- Isso faz com que o valor do atributo só possa ser modificado dentro do escopo da classe que o declara.
- Acesso direto aos atributos são negados inclusive no escopo de classes que herdem a classe dona do atributo.
- Vejamos…

Modificadores de Acesso - Private

```
public class Retangulo {
        private float base;
        private float altura;

        public float calcularArea() {
                return base*altura;
        }
}
```

```
public class GerenciadorDeFormas
{
    private Retangulo retangulo = new Retangulo();

    public GerenciadorDeFormas()
    {
        retangulo.base = 3;
        retangulo.altura = 4;
    }
}
```

Erro Aqui

Veja que definimos lado e altura agora como ***private***. O compilador irá acusar um erro em ***GerenciadorDeFormas,*** pois este está tentando acessar os atributos ***lado*** e ***altura*** diretamente.

Modificadores de Acesso - Private

```
public class Retangulo {
        private float base;
        private float altura;

        public float calcularArea() {
                return base*altura;
        }

        public float setBase(float base) {
                this.base = base;
        }

        public float setAltura(float altura) {
                this.altura = altura;
        }


}
```

```
public class GerenciadorDeFormas
{
    private Retangulo retangulo = new Retangulo();

    public GerenciadorDeFormas()
    {
        retangulo.setBase(3);
        retangulo.setAltura(4);
    }
}
```

OK!

Para resolver o problema, criamos um método público dentro no retângulo que modifica o valor.

Modificadores de Acesso - Protected

- O modificador ***protected*** tem o mesmo efeito do ***private*** com a exceção de que ele permite a manipulação direta do atributo a subclasses da classe onde foi declarada.

- Ou seja, acesso direto dentro do escopo de uma classe qualquer não é permitido, porém se essa classe herda uma classe com um atributo ***protected*** ela pode ser acessada no escopo desta subclasse em específico.

- Vejamos um exemplo…

Modificadores de Acesso - Protected

```
public class Retangulo {
        protected float base;
        protected float altura;

        public float calcularArea() {
                return base*altura;
        }
}
```

```
public class GerenciadorDeFormas
{
    private Retangulo retangulo = new Retangulo();

    public GerenciadorDeFormas()
    {
        retangulo.base = 3;
        retangulo.altura = 4;
    }
}
```

Erro Aqui

Veja que definimos lado e altura agora como ***private***. O compilador irá acusar um erro em ***GerenciadorDeFormas,*** pois este está tentando acessar os atributos ***lado*** e ***altura*** diretamente.

Modificadores de Acesso - Protected

```
public class Retangulo {
        protected float base;
        protected float altura;

        public float calcularArea() {
            return base*altura;
        }
}
```

```
public class Quadrado extends Retangulo {

        public float calcularArea() {
            return this.base*this.base;
        }

        public float setBase(float base) {
            this.base = base;
        }
}
```

OK!

OK!

```
public class GerenciadorDeFormas
{
    private Retangulo retangulo = new Retangulo();

    public GerenciadorDeFormas()
    {
        retangulo.base = 3;
        retangulo.altura = 4;
    }
}
```

Erro Aqui

O atributo ***protected*** permite acesso direto a classes que herdem a a classe onde ele foi declarado.

Modificadores de Acesso - Protected

- Quando usar o ***protected***?
- A existência do ***protected*** é de certa forma controversa, pois pode se obter o mesmo efeito sobrecarregando o método ***set*** superclasse contendo o atributo ***protected***.
  - Poderia se dizer aqui que o acesso direto seria mais cômodo do que utilizar um método de acesso.
- De certa forma o ***protected*** "reencapsula" o atributo e passa a ditar as regras de acesso ao atributo. Mas ainda assim o mesmo efeito pode ser obtido com os métodos de acesso.
- De certa algumas linguagens não possuem o modificador de acesso ***protected***.

Métodos de Acesso - **getters** e **setters**

Setters e getters

- Como se resolve então o acesso a atributos privados ***(private)***?
- Como vimos anteriormente criamos métodos responsáveis por modificar e retornar o valor do atributo.
- A esse métodos chamamos de:
    - ***getters:*** métodos que retornam o valor de um atributo.
    - ***setters:*** métodos que modificam o valor de um atributo.

Setters e getters

- Por convenção os métodos de acesso tem os nomes dos atributos que acessam, precedidos da palavra…
  - get - para o método que retorna o valor do atributo (ex. getLado)
  - set - para o método que modifica o valor do atributo (ex. setLado)

Setters e getters

```
public class Retangulo {
  private float base;
  private float altura;

  public void setBase(float base) {
    this.base = base
  }

  public float getBase() {
    return this.base
  }
}
```

- Ao lado definimos os métodos de acesso para o atributo privado **base**.
- Se temos uma variável ***retangulo1*** do tipo ***Retangulo*** modificaremos os valores com:
  - `retangulo1.setBase(4);`
- E acessaremos o valor com:
  - `retangulo1.getBase();`
- Note que estes métodos são públicos.

Exercício

- Aperfeiçoe os exercícios feitos até aqui criando os métodos de acesso para os atributos das classes.
- Em caso de herança analise possíveis atributos ***protected*** nas superclasses.