

# Programação Orientada a Objetos

Contato:  
anderson.elias@estacio.br



**Estácio**

1

## Modificadores de Acesso

- São padrões de visibilidade de acessos às classes, atributos e métodos;
- Tem como objetivo prover segurança e confiabilidade no acesso às classes, atributos e métodos;
- São palavras-chaves reservadas pelo Java;  
**Public, Private, Protected, Default, Final, Static e Abstract**

2

# Modificadores de Acesso

	private	default	protected	public
mesma classe	sim	sim	sim	sim
mesmo pacote	não	sim	sim	sim
pacotes diferentes (subclasses)	não	não	sim	sim
pacotes diferentes (sem subclasses)	não	não	não	sim

3

# Modificadores de Acesso

- **Public:**

Uma declaração com o modificador **public** pode ser acessada de qualquer lugar e por qualquer entidade que possa visualizar a classe a que ela pertence.

4

# Publico

**Classes:** Todas as demais enxergam.

```
public class MinhaClasse {
    //...
}
```

**Métodos:** Todas as classes o enxergam, desde que enxerguem a classe também.

```
public class MinhaClasse {
    public void meuMetodo() { }
}
```

5

# Publico

**Atributo:** Todas as classes o enxergam, desde que enxerguem a classe também.

```
public class MinhaClasse {
    public int atributo = 1;
}
```

6

# Modificadores de Acesso

- **Private:**

Os membros da classe definidos como **private** não podem ser acessados ou usados por nenhuma outra classe. Esse modificador não se aplica às classes, somente para seus métodos e atributos. Esses atributos e métodos também não podem ser visualizados pelas classes herdadas.

7

## Private

**Classes:** Somente classes no mesmo arquivo fonte enxergam. Use isso quando a implementação serve para algum algoritmo interno da classe principal.

```
public class MinhaClasse {  
    private class MinhaClasseInternaPrivada {  
        //...  
    }  
}
```

Observe que cada arquivo .java pode ter apenas uma classe pública (**classes internas**).

8

## Pausa (Classes Internas)

Podemos definir classes dentro de classes, como se fossem atributos ou métodos;

- **Vantagens:**

- **Legibilidade:** agrupamento por similaridade;
- **Ocultamento:** podem ser privadas ou protegidas;
- **Redigibilidade:** classes internas possuem acesso aos membros privados da classe que a definiu e vice-versa.
- **Código Fonte:** ficam no mesmo arquivo **.java** porém vários **.class**;
- **Tipos de Classes Internas:** **Aninhadas e Instanciadas.**

9

## Private

**Método:** Somente classes no mesmo arquivo fonte enxergam. Use isso quando o método é feito apenas para uso dos outros métodos públicos da classe.

```
public class MinhaClasse {
    private void meuMetodoSecreto() {

    }
    public void meuMetodoPublico() {
        meuMetodoSecreto();
    }
}
```

Métodos privados **não podem ser sobrescritos.**

10

# Private

**Atributos:** Somente classes no mesmo arquivo fonte enxergam. Procure deixar todos os seus atributos privados e dar o acesso **encapsulado** a eles através de **getters e setters**.

```
public class MinhaClasse {  
    private int atributo = 1;  
    public int getAtributo() {  
        return atributo;  
    }  
    public void setAtributo(int atributo) {  
        this.atributo = atributo;  
    }  
}
```

11

# Modificadores de Acesso

- Protected:

O modificador **protected** torna o membro acessível às classes do mesmo pacote ou através de **herança**, seus membros herdados não são acessíveis em outras classes fora do pacote onde estes membros foram declarados.

12

# Protected

**Método:** Métodos protegidos podem ser vistos pelas classes do mesmo pacote ou por subclasses.

```
public class MinhaClasse {  
    protected void meuMetodo() { }  
}
```

Use isso se for fazer algum tipo de biblioteca que permita a outro desenvolvedor estender suas classes e então usar esses métodos especiais, os quais não devem ser chamados por outras classes que usam a sua biblioteca.

13

# Protected

**Atributos:** Atributos protegidos podem ser vistos pelas classes do mesmo pacote ou por subclasses.

```
public class MinhaClasse {  
    protected int atributo = 1;  
}
```

14



# Modificadores de Acesso

- Default:

A classe e/ou seus membros são acessíveis somente por classes do mesmo pacote, **na sua declaração não é definido nenhum tipo de modificador**, sendo este identificado pelo compilador.

15

## Default

**Classes:** Somente classes no mesmo arquivo fonte ou no mesmo pacote (package) enxergam. Use quando a implementação serve apenas para sua biblioteca ou para uma determinada parte do seu programa.

```
class MinhaClasse {  
    //...  
}
```

16



## Default

**Método:** Métodos sem modificador podem ser vistos apenas pelas classes do mesmo pacote.

```
public class MinhaClasse {  
    void meuMetodo() { }  
}
```

Use isso quando um método é usado apenas pelas classes que compõe uma parte do seu programa.

17

## Default

**Atributos:** Atributos sem modificador podem ser vistos apenas pelas classes do mesmo pacote.

```
public class MinhaClasse {  
    int atributo = 1;  
}
```

18

# Modificadores de Acesso

- Final:

Quando é aplicado na classe, **não permite estende-la (bloqueia a herança)**, nos métodos impede que o mesmo seja sobrescrito (**overriding**) na subclasse, e nos valores de variáveis não pode ser alterado depois que já tenha sido atribuído um valor. Torna uma variável/atributo em uma **constante**.

19

# Modificadores de Acesso

- Abstract:

Esse modificador não é aplicado nas variáveis/atributo, **apenas nas classes**. Uma classe abstrata não pode ser instanciada, ou seja, não pode ser chamada pelos seus construtores. Se houver alguma declaração de um método como abstract (abstrato), a classe também deve ser marcada como abstract.

20

# Modificadores de Acesso

- Static:

É usado para a criação de uma variável/atributo que poderá ser acessada por todas as instâncias de objetos desta classe como uma variável comum, ou seja, a variável criada será a mesma em todas as instâncias e **quando seu conteúdo é modificado em uma das instâncias, a modificação ocorre em todas as demais instâncias de objetos proveniente da mesma classe**. E nas declarações de métodos ajudam no acesso direto à classe, portanto não é necessário instanciar um objeto para acessar o método.

21

# Modificadores de Acesso

- Static:

- É algo relacionado com constante, algo 'parado' (estático);
- Com uma classe criamos vários objetos dela e cada objeto irá ser uma cópia fiel da classe, porém com suas próprias variáveis e métodos em lugares distintos da memória;
- O objeto tem suas variáveis próprias;
- Quando usamos static em membros de uma classe, temos um comportamento especial: **será o mesmo para todos os objetos daquela classe**.

22

# Modificadores de Acesso

- **Static:**

- Ou seja, não haverá um tipo dela em cada objeto.
- Todos os objetos, ao acessarem e modificarem essa variável, acessarão a mesma variável, o mesmo espaço da memória;
- Com isto, a mudança poderá ser vista em todos os objetos;
- Se usa quando quer ter um controle sobre os objetos ou quando todos os objetos devem compartilhar uma informação;
- Métodos e atributos declarados como estáticos só podem ser acessados no contexto estático.

23

## Static

**Classes:** Classes static são classes declaradas dentro de outra classe que podem ser usadas sem a necessidade de uma instância.

```
public class MinhaClasse {  
    public static classe ClasseInterna { }  
}
```

Então podemos acessar desta forma:

```
MinhaClasse.ClasseInterna instancia = new MinhaClasse.ClasseInterna();
```

24


# Static

**Método:** Os métodos static podem ser chamados sem uma instância. São ótimos como utilitários.

```
public final class Integer extends Number implements Comparable<Integer> {
    public static Integer valueOf(String s, int radix) throws NumberFormatException {
        return new Integer(parseInt(s, radix));
    }
}
```

Você pode chamar assim:

```
Integer valor = Integer.valueOf("FF", 16);
```



25

# Static

**Método:** Métodos static não podem acessar variáveis de instância.

```
public class MinhaClasse {
    int valor = 1;
    public static int estatico() {
        return valor; //erro de compilação aqui!!!
    }
}
```

26

# Static

**Atributos:** Os atributos static possuem o mesmo valor para todas as instâncias de um objeto (dentro de uma mesma JVM, dentro de um mesmo ClassLoader).

```
public class MinhaClasse {
    static int valorGlobal = 1;
    public static int getValorGlobal() {
        return valorGlobal;
    }
}
```

```
public class MinhaClasse2 {
    public static void main(String[] args) {
        MinhaClasse c1 = new MinhaClasse();
        MinhaClasse c2 = new MinhaClasse();
        MinhaClasse.valorGlobal = 2;
        System.out.println(c1.valorGlobal); //imprime 2
        System.out.println(c2.valorGlobal); //imprime 2
    }
}
```

27

# Modificadores de Acesso

- **Static:**
  - **Exemplo:** Cria uma classe "Carro" que informa quando o objeto é criado - através do método construtor padrão main - e incrementa a variável 'total', que vai guardar a informação do número total de objetos/carros criados em sua aplicação.

28

# Modificadores de Acesso

- Static:

```
package carro;
public class Carro {
    public static int total=0;

    Carro(){
        total++;
        System.out.println("Objeto criado. Existem "+total+" objetos dessa classe");
    }
}
```

29

# Modificadores de Acesso

- Static:

```
package carro;

public class StaticTeste {
    public static void main(String[] args) {
        Carro fusca = new Carro();
        Carro ferrari = new Carro();
        Carro jipe = new Carro();
    }
}
```

30



# Modificadores de Acesso

- Static:

**Resultado do exercício:**

Objeto criado. Existem 1 objetos dessa classe

Objeto criado. Existem 2 objetos dessa classe

Objeto criado. Existem 3 objetos dessa classe

31

# Modificadores de Acesso

**DESAFIOS**

32

# Modificadores de Acesso

- **Static:**

- **Desafio 1:** Desenvolva um programa para um setor de vendas de uma empresa onde será gerado um objeto Pedidos que terá as seguintes características: **número, nome do cliente e valor;**
- O atributo **número** será um atributo estático da classe para gerar um número incremental único para todos os pedidos gerado sempre que necessário.
- **Para testar, gere 3 pedidos onde o número tem de ser auto-incrementado e exiba os dados dos 3 pedidos.**

33

## (POO) – Exercícios

### Lista 3

34

# Dúvidas?



## Estácio

