




**AULA e3**



# Programação Orientada a Objetos

e-Aula 03

Prof. Maristela Weinfurter

**Vamos lembrar o que é  
Herança!**



- **Superclasses**
- **Subclasses**
- **extends**

- **Superclasse direta:** é aquela a partir da qual a subclasse herda explicitamente.
- **Superclasse indireta:** é qualquer classe acima da superclasse direta na hierarquia de classes.

- **Java só suporta HERANÇA ÚNICA.**
- **Logo, somente diretas.**


- **A anotação @Override indica que um método deve sobrescrever um método da superclasse**

- Quando o compilador encontra uma notação @Override ele compara a assinatura do método com as assinaturas dos métodos da superclasse.


**Agora vamos revisar sobre  
POLIMORFISMO!**









**A palavra Polimorfismo vem do grego poli morfos e significa “muitas formas”.**



É exatamente isso que este paradigma significa: a existência de várias formas, ou implementações, para os métodos de uma classe.



**O polimorfismo é uma grande contribuição para a programação orientada a objetos, visto que permite que adequemos as ações das classes ao que elas se propõem.**




**Considere que temos uma classe chamada Funcionario com um método para calcular o salário com bonificação.**

# e-Aula 03 - POO

Polimorfismo  
DINÂMICO

```
public class Funcionario {  
    private double salario;  
    public void calculaBonificacao() {  
        salario = salario + (salario *  
0,15);  
    }  
}
```



**Essa classe Funcionario será base para outras classes, que terão seus próprios cálculos de bonificação.**


# e-Aula 03 - POO

Polimorfismo  
DINÂMICO

```
public class Gerente extends Funcionario {  
    public void calculaBonificacao() {  
        salario = salario + (salario * 0,2);  
    }  
}
```

**Essa maneira de polimorfismo é chamada de polimorfismo dinâmico ou sobrescrita de método, pois temos duas classes envolvidas, a base e a derivada, e um método com a mesma assinatura nas duas classes.**





O polimorfismo exige que os métodos tenham a mesma assinatura, pois, do contrário, serão tratados como métodos diferentes pelas linguagens que implementam a POO.

**Além desta forma de polimorfismo, temos a situação em que uma determinada ação de uma classe pode ser executada de forma diferente dependendo do tipo de parâmetro que o método receba**

# e-Aula 03 - POO


```
public class Funcionario {  
    private String nome;  
    private double salario;
```

```
public Funcionario() {  
}
```

```
public Funcionario(String nome) {  
    this.nome = nome;  
}
```


```
public Funcionario(String nome, double salario) {  
    this.nome = nome;  
    this.salario = salario;  
}
```

Polimorfismo  
ESTÁTICO



**Nesta classe, temos uma sobrecarga do construtor, que tem três assinaturas diferentes que executam ações diferentes.**

- o primeiro é construtor padrão
- o segundo inicializa o atributo nome
- o terceiro inicializa todos os atributos da classe.
-



**Neste caso, temos o polimorfismo estático, no qual apenas uma classe e o método sobrecarregado tem parâmetros diferentes, sendo que o retorno e o nome do método continuam iguais.**

# e-Aula 03 - POO

