Herança

Profª Lucília Ribeiro

Nenhuma herança é tão rica quanto a honestidade.

(William Shakespeare)

O que é Herança?

Mecanismo que permite basear uma nova classe na definição de uma classe previamente existente.
Usando herança, sua nova classe herda todos os atributos e comportamentos presentes na classe previamente existente.
Quando uma classe herda de outra, todos os métodos e atributos que aparecem na interface da classe previamente existente aparecerão automaticamente na interface da nova classe.

```java
public class Empregado {
    private String nome;
    private float salario;

    public Empregado(String nome, float salario) {
        this.nome = nome;
        this.salario = salario;
    }

    public String getNome() {
        return nome;
    }
    public float getSalario() {
        return salario;
    }
}
```

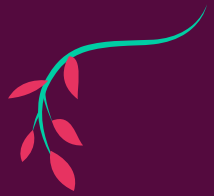
Criar um Empregado Comissionado? (salário base + comissão)

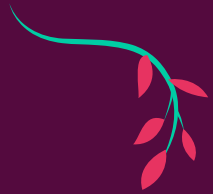
EmpregadoComissionado É UM Empregado

Copiar e colar código não é uma boa solução

Ter uma variável empregado dentro da classe EmpregadoComissionado e DELEGAR todas as mensagens à instância de Empregado

DELEGAÇÃO
(ou troca de mensagens)





É o processo de um objeto passar uma mensagem para outro objeto, para atender algum pedido.

(problema): a delegação obriga a redefinir todos os métodos encontrados na interface de Empregado para passar todas as mensagens

```java
public class EmpregadoComissionado  extends Empregado {
    private float comissao;
    private int unidades;

    public EmpregadoComissionado(String nome, float salario, float, comissao) {
        super(nome, salario);
        this.comissao = comissao;
    }

    public float calculaPagamento() {
        return getSalario() + (comissao * unidades);
    }

    public void adicionaVendas(int unidades) {
        this.unidades = this.unidades + unidades;
    }

    public void zeraVendas() {
        unidades = 0;
    }
}
```

```java
public class TestaEmpregado {
    public static void main(String arg[ ]) {
        EmpregadoComissionado empCom = new EmpregadoComissionado("Andre Ribeiro", 5500f, 530.10f);
        empCom.adicionaVendas(500);
        System.out.println("*******CONTRA CHEQUE*******");
        System.out.println("Nome.........: " + empCom.getNome());
        System.out.println("Salario Base: " + empCom.getSalario());
        System.out.println("TOTAL........: " + empCom.calculaPagamento());
    }
}
```

```
*******CONTRA CHEQUE*******
Nome.........: Andre Ribeiro
Salario Base: 5500.0
TOTAL........: 270550.0
```

Por que Herança?

- Muito mais do que herdar uma interface pública e implementação.
- Permite à classe que está herdando REDEFINIR qualquer comportamento de que não goste.
- Adaptar seu software quando os requisitos mudarem.
- Para fazer alterações, basta escrever uma nova classe que herde a antiga funcionalidade. Então SOBREPONHA a funcionalidade que precisa mudar ou ADICIONE a funcionalidade que está faltando.
- A sobreposição é interessante pois permite mudar a maneira como um objeto funciona sem tocar na definição original.
- Deixa o código original testado e validado intacto.
- A sobreposição funciona mesmo sem o código-fonte original.
- A HERANÇA permite que você AGRUPE classes relacionadas.

"É um" versus "Tem um"

- A herança de implementação permite que as classes herdem a implementação de outras classes. Não é porque uma classe pode herdar de outra que isso deve ser feito – QUANDO USAR HERANÇA?
- Ao considerar o uso de herança, PRIMEIRAMENTE pergunte se a classe que está herdando é do mesmo tipo que a classe que está sendo herdada – É UM?
- É UM descreve o relacionamento em que uma classe é considerada do mesmo tipo de outra (uso de herança para REUSO)
- Outra maneira de reuso: composição e delegação – TEM UM?
- TEM UM descreve o relacionamento em que uma classe contém uma instância de outra classe
- COMPOSIÇÃO: uma classe é implementada usando variáveis internas, que contém instâncias de outras classes

```java
public class Ponto2D {
    private double x;
    private double y;

    public Ponto2D(double x, double y) {
        setX(x);
        setY(y);
    }

    public double getX() {
        return x;
    }
    public void setX(double x) {
        this.x = x;
    }
    public double getY() {
        return y;
    }
    public void setY(double y) {
        this.y = y;
    }

    public String toString() {
        return "Eu sou um ponto BIDIMENSIONAL (2D)\n" +
        "Minha coordenada X = " + getX() + "\n" +
        "Minha coordenada Y = " + getY() + "\n";
    }
}
```

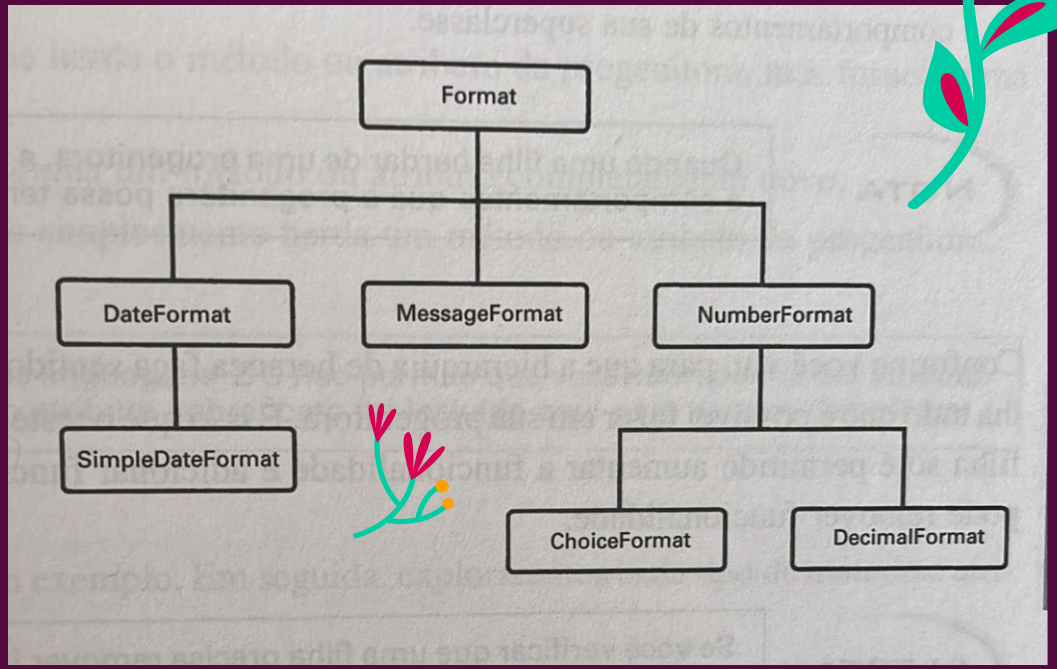
```java
public class Ponto3D extends Ponto2D {
    private double z;

    public Ponto3D(double x, double y, double z) {
        super(x, y);
        setZ(z);
    }

    public double getZ() {
        return z;
    }
    public void setZ(double z) {
        this.z = z;
    }

    public String toString() {
        return "Eu sou um ponto TRIDIMENSIONAL (3D)\n" +
        "Minha coordenada X = " + getX() + "\n" +
        "Minha coordenada Y = " + getY() + "\n" +
        "Minha coordenada Z = " + getZ() + "\n";
    }
}
```

```java
public class PontoTeste {
    public static void main(String arg[ ]) {
        Ponto2D dois = new Ponto2D(1, 2);
        Ponto3D tres = new Ponto3D(3, 4, 5);

        System.out.println(dois.toString());
        System.out.println(tres.toString());
    }
}
```

```
Eu sou um ponto BIDIMENSIONAL (2D)
Minha coordenada X = 1.0
Minha coordenada Y = 2.0

Eu sou um ponto TRIDIMENSIONAL (3D)
Minha coordenada X = 3.0
Minha coordenada Y = 4.0
Minha coordenada Z = 5.0
```

Hierarquia de Classes

Herança é um mecanismo que permite estabelecer relacionamento É UM entre classes. Permite também que uma subclasse herde os atributos e comportamentos de uma superclasse



Dicas

- Deve ser possível fazer na filha, tudo que é possível fazer na mãe.
- A uma filha só é permitido melhorar ou adicionar funcionalidades. (a filha pode aprender a tocar piano, ou ser melhor em matemática que a mãe)
- OBS: Se precisar remover funcionalidades da filha, isso indica que ela deve estar antes da mãe na hierarquia.
- Por definição, a linguagem Java não permite herança múltipla, ou seja: cada filha tem somente uma mãe.

Maneiras de usar Herança

01

Para reuso de implementação

02

Para diferença

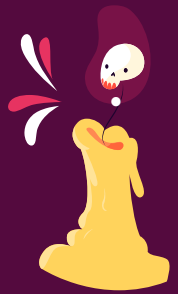
03

Para substituição de tipos

Herança para Diferença

- PROGRAMAÇÃO POR DIFERENÇA significa herdar uma classe e adicionar apenas o código que torne a nova classe diferente da classe herdada.
- Permite que programe por incrementos. Códigos mais SIMPLES, MENORES e com MENOS ERROS

- ESPECIALIZAÇÃO: é o processo de uma classe filha ser projetada em termos de como ela é diferente de sua progenitora.



Quando você percorre uma hierarquia pra baixo, você ESPECIALIZA. Quando percorre para cima, você GENERALIZA.

```java
public class Linha {
    private Ponto2D p1;
    private Ponto2D p2;

    public Linha(Ponto2D p1, Ponto2D p2) {
        this.p1 = p1;
        this.p2 = p2;
    }
    public Ponto2D getPonto1() {
        return p1;
    }
    public Ponto2D getPonto2() {
        return p2;
    }
    public double getDistancia() {
        double x, y, distancia;
        x = Math.pow((p2.getX() - p1.getX()), 2);
        y = Math.pow((p2.getY() - p1.getY()), 2);
        distancia = Math.sqrt(x + y);
        return distancia;
    }
    public Ponto2D getPontoMedio() {
        double coordX, coordY;
        coordX = (p1.getX() + p2.getX()) / 2;
        coordY = (p1.getY() + p2.getY()) / 2;
        return new Ponto2D(coordX, coordY);
    }
}
```

```java
public class LinhaTeste {
    public static void main(String arg[ ]) {
        Ponto3D p1 = new Ponto3D(12, 22, 2);
        Ponto2D p2 = new Ponto2D(16, 19);
        Linha linha = new Linha(p1, p2);
        System.out.println("Ponto Medio: " + linha.getPontoMedio());
        System.out.println("Distancia: " + linha.getDistancia());
    }
}
```

```
Ponto Medio: Eu sou um ponto BIDIMENSIONAL (2D)
Minha coordenada X = 14.0
Minha coordenada Y = 20.5

Distancia: 5.0
```

Herança para Substituição de Tipos

Obrigada!

professora@lucilia.com.br