Fatec
Mogi Mirim
Arthur de Azevedo

PROGRAMAÇÃO ORIENTADA A OBJETOS [POO]

Material 06 – POO_06

Prof. Mestre Marcos Roberto de Moraes [Maromo]

Herança - Generalização / Especialização - Sobrescrita (Overwrite) - Classes Abstratas / Métodos Abstratos - Polimorfismo

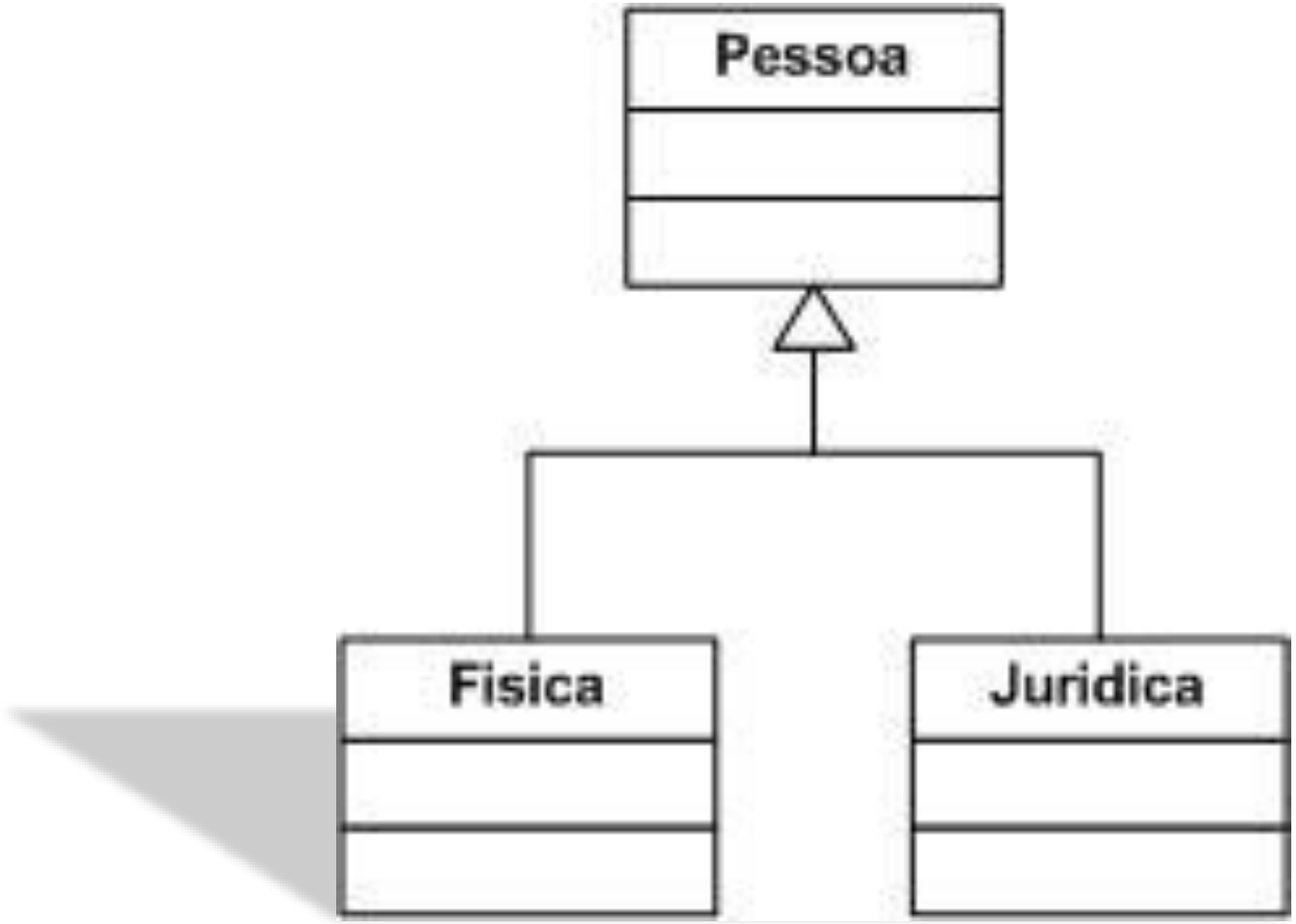
Agenda

- Herança
- Generalização / Especialização
- Exemplo
- Sobrescrita (Overwrite)
- Classes Abstratas / Métodos Abstratos
- Polimorfismo
- Exercício

Herança

- Princípio da Programação Orientada a Objetos que permite que as classes compartilhem atributos e métodos comuns baseados em um relacionamento.
- A herança possibilita a aplicação de vários conceitos de orientação a objetos que não seriam viáveis sem a organização e estruturação alcançada por sua utilização.

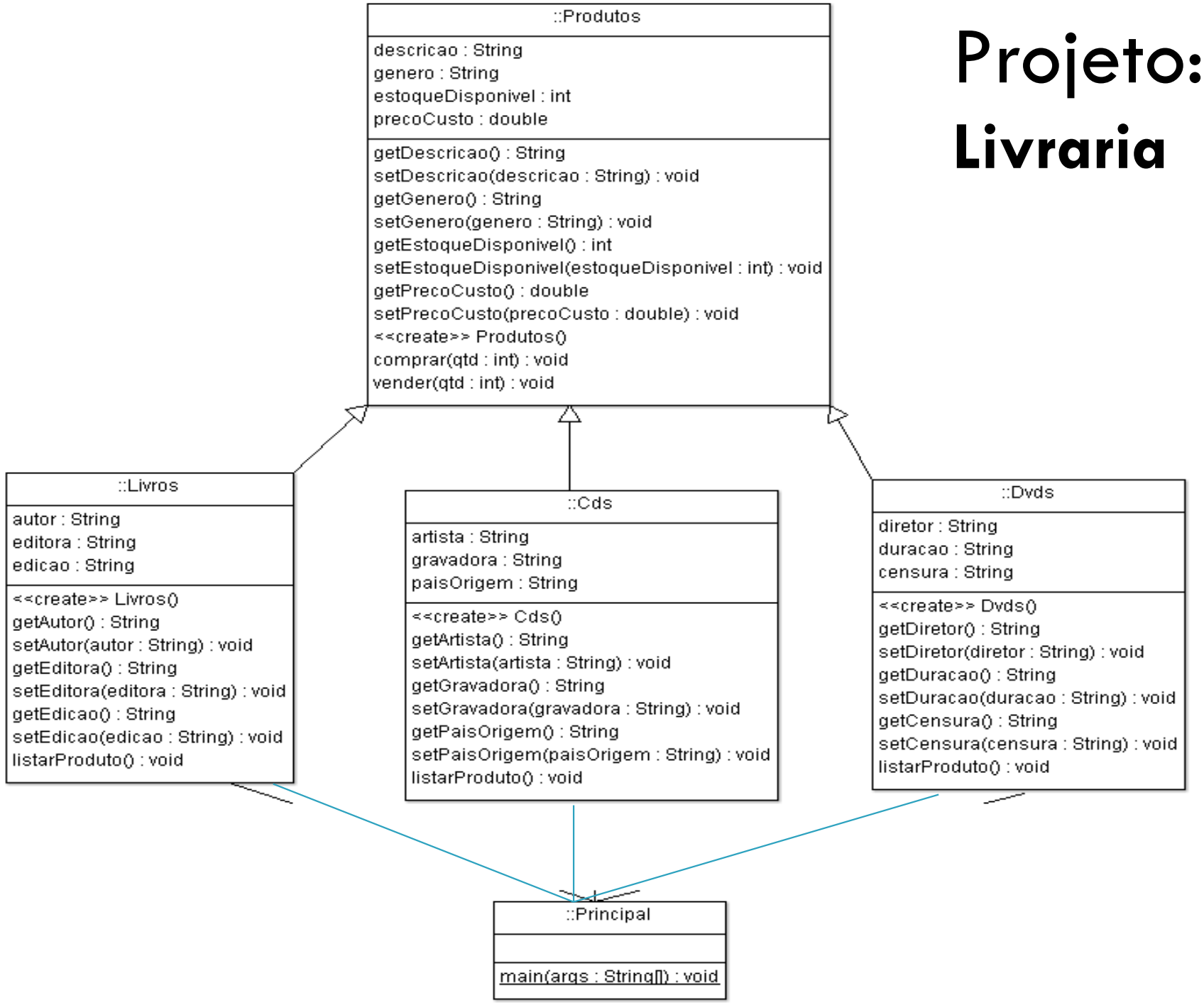
Herança – Representação UML



Generalização / Especialização

- É a 1ª abordagem a ser feita sobre **herança**.
- Possibilita a vantagem mais direta (reutilização de código).
- A **generalização** é o agrupamento de características (atributos) e regras (métodos) comuns em um modelo de sistema.
- A **especialização** é o processo inverso, é a definição das particularidades de cada elemento de um modelo de sistemas, detalhando características e regras especificas de um o objeto.

Projeto: **Livraria**



superclasse: Produto

Linha 1 até 25

```
2   //SuperClasse ou Classe Mãe
    public class Produtos {
4       //Membros com acesso (private)
5       private String descricao;
6       private String genero;
7       private int estoqueDisponivel;
8       private double precoCusto;
9
10      //Métodos Modificadores de Acesso aos Campos
11      public String getDescricao() {
12          return descricao;
13      }
14      public void setDescricao(String descricao) {
15          this.descricao = descricao;
16      }
17      public String getGenero() {
18          return genero;
19      }
20      public void setGenero(String genero) {
21          this.genero = genero;
22      }
23      public int getEstoqueDisponivel() {
24          return estoqueDisponivel;
25      }
```

superclasse: Produto

Linha 26 até 52

```
26      public void setEstoqueDisponivel(int estoqueDisponivel) {
27          this.estoqueDisponivel = estoqueDisponivel;
28      }
29      public double getPrecoCusto() {
30          return precoCusto;
31      }
32      public void setPrecoCusto(double precoCusto) {
33          this.precoCusto = precoCusto;
34      }
35
36      //Método Construtor sem parâmetos
37      public Produtos(){
38          this.descricao = "";
39          this.estoqueDisponivel =0;
40          this.genero = "";
41          this.precoCusto=0.0;
42      }
43      //Métodos específicos da Classe:
44
45      public void comprar(int qtd){
46          //Apresenta o estoque atual. E o estoque atualizado.
47          System.out.println("<---- ESTOQUE ---->");
48          System.out.println("Quantidade do Produto " + this.descricao);
49          System.out.println("Anterior: " + this.estoqueDisponivel);
50          this.estoqueDisponivel+=qtd;
51          System.out.println("Atual    : " + this.estoqueDisponivel);
52      }
```

superclasse: Produto

Linha 53 ao final

```
52      }
53      public void vender(int qtd){
54          //Apresenta o estoque atual. E o estoque atualizado.
55          System.out.println("<---- ESTOQUE ---->");
56          System.out.println("Quantidade do Produto " + this.descricao);
57          System.out.println("Anterior: " + this.estoqueDisponivel);
58          this.estoqueDisponivel-=qtd;
59          System.out.println("Atual    : " + this.estoqueDisponivel);
60      }
61  }
62
```

subclasse: Livro

Linha 1 a 24

```
2   public class Livros extends Produtos{
3       //Membros ou atributos (privados)
4       private String autor;
5       private String editora;
6       private String edicao;
7
8       //Construtor
9       public Livros(){
10          this.autor = "";
11          this.editora = "";
12          this.edicao = "";
13      }
14
15      //Métodos gets/sets
16      public String getAutor() {
17          return autor;
18      }
19      public void setAutor(String autor) {
20          this.autor = autor;
21      }
22      public String getEditora() {
23          return editora;
24      }
```

subclasse: Livro

Linha 25 ao final

```
25      public void setEditora(String editora) {
26          this.editora = editora;
27      }
28      public String getEdicao() {
29          return edicao;
30      }
31      public void setEdicao(String edicao) {
32          this.edicao = edicao;
33      }
34      //Método especifico da classe.
35      public void listarProduto(){
36          System.out.println("<---- DADOS DO PRODUTO ---->");
37          System.out.println("Descrição.....: " + this.getDescricao());
38          System.out.println("Gênero.......: " + this.getGenero());
39          System.out.println("Estoque......: " + this.getEstoqueDisponivel());
40          System.out.println("Preço........: R$ " + this.getPrecoCusto());
41          System.out.println("Autor........: " + this.autor);
42          System.out.println("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");
43          System.out.println();
44      }
45  }
```

subclasse: Dvds

Linha 1 a 23

```
2   public class Dvds extends Produtos {
3       //Membros
4       private String diretor;
5       private String duracao;
6       private String censura;
7
8       //Construtor
9       public Dvds(){
10          this.diretor="";
11          this.duracao="";
12          this.censura="";
13      }
14
15      public String getDiretor() {
16          return diretor;
17      }
18    public void setDiretor(String diretor) {
19          this.diretor = diretor;
20      }
21      public String getDuracao() {
22          return duracao;
23      }
```

subclasse: Dvds

Linha 24 ao final

```
24      public void setDuracao(String duracao) {
25          this.duracao = duracao;
26      }
27      public String getCensura() {
28          return censura;
29      }
30      public void setCensura(String censura) {
31          this.censura = censura;
32      }
33       //Método Específico da Classe
34       public void listarProduto(){
35          System.out.println("<---- DADOS DO PRODUTO ---->");
36          System.out.println("Descrição.....: " + this.getDescricao());
37          System.out.println("Gênero........: " + this.getGenero());
38          System.out.println("Estoque.......: " + this.getEstoqueDisponivel());
39          System.out.println("Preço.........: R$ " + this.getPrecoCusto());
40          System.out.println("Diretor.......: " + this.diretor);
41          System.out.println("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");
42          System.out.println();
43      }
44  }
```

subclasse: Cds

Linha 1 a 25

```
2   public class Cds extends Produtos{
3       private String artista;
4       private String gravadora;
5       private String paisOrigem;
6
7       //Construtor
8       public Cds(){
9           this.artista = "";
10          this.gravadora = "";
11          this.paisOrigem = "";
12      }
13      //Métodos Modificadores de Acesso
14      public String getArtista() {
15          return artista;
16      }
17      public void setArtista(String artista) {
18          this.artista = artista;
19      }
20      public String getGravadora() {
21          return gravadora;
22      }
23      public void setGravadora(String gravadora) {
24          this.gravadora = gravadora;
25      }
```

subclasse: Cds

Linha 26 ao final

```
26      public String getPaisOrigem() {
27          return paisOrigem;
28      }
29      public void setPaisOrigem(String paisOrigem) {
30          this.paisOrigem = paisOrigem;
31      }
32      //Método Específico da Classe
33       public void listarProduto(){
34          System.out.println("<---- DADOS DO PRODUTO ---->");
35          System.out.println("Descrição.....: " + this.getDescricao());
36          System.out.println("Gênero.......: " + this.getGenero());
37          System.out.println("Estoque......: " + this.getEstoqueDisponivel());
38          System.out.println("Preço........: R$ " + this.getPrecoCusto());
39          System.out.println("Artista......: " + this.artista);
40          System.out.println("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");
41          System.out.println();
42      }
43  }
44
```

Classe: Principal

Linha 1 a 23

```
2  import java.util.Scanner;
3  public class Principal {
4      public static void main(String[] args) throws InterruptedException {
5          Scanner sc = new Scanner(System.in);
6          //Criando um objeto livro
7          Livros livro = new Livros();
8          livro.setDescricao("Java com Ênfase em OO");
9          livro.setEdicao("N. 1");
10         livro.setAutor("Douglas Mendes");
11         livro.setEditora("Novatec");
12         livro.setGenero("Programação");
13         livro.setEstoqueDisponivel(10);
14         livro.setPrecoCusto(100.90);
15         //Criando um objeto dvd
16         Dvds dvd = new Dvds();
17         dvd.setDescricao("Acima de Qualquer Suspeita");
18         dvd.setCensura("14 anos");
19         dvd.setDiretor("Michael Douglas");
20         dvd.setDuracao("120 minutos");
21         dvd.setEstoqueDisponivel(5);
22         dvd.setGenero("Drama");
23         dvd.setPrecoCusto(66.98);
```

Classe: Principal

Linha 24 a 50

```
24          //Criando um objeto cd
25          Cds cd = new Cds();
26          cd.setDescricao("Toquinho 10 anos");
27          cd.setGravadora("Sony");
28          cd.setPaisOrigem("Brasil");
29          cd.setArtista("Toquinho");
30          cd.setEstoqueDisponivel(5);
31          cd.setGenero("MPB");
32          cd.setPrecoCusto(14.99);
33          //Menu - Escolha uma opção
34          int opc=0;
35          while(opc!=7){
36              System.out.println("Escolha a opção");
37              System.out.println("1--> Comprar Mais Exemplar do Livro  ");
38              System.out.println("2--> Vender Exemplar do Livro");
39              System.out.println("3--> Comprar Mais um Título do DVD ");
40              System.out.println("4--> Vender Um Título do DVD");
41              System.out.println("5--> Comprar Livro");
42              System.out.println("6--> Vender Livro");
43              System.out.println("7--> Sair");
44              opc = sc.nextInt();
45              switch(opc){
46                  case 1:
47                      System.out.println("Quantidade a comprar do livro...: " +
48                              livro.getDescricao());
49                      livro.comprar(sc.nextInt());
50                      break;
```

Classe: Principal
Linha 51 a 79

```
51                  case 2:
52                      System.out.println("Quantidade a vender de livro...: " +
53                              livro.getDescricao());
54                      livro.vender(sc.nextInt());
55                      break;
56                  case 3:
57                      System.out.println("Quantidade a comprar do DVD...: " +
58                              dvd.getDescricao());
59                      dvd.comprar(sc.nextInt());
60                      break;
61                  case 4:
62                      System.out.println("Quantidade a vender do DVD..: " +
63                              dvd.getDescricao());
64                      dvd.vender(sc.nextInt());
65                      break;
66                  case 5:
67                      System.out.println("Quantidade a comprar do CD..: " +
68                              cd.getDescricao());
69                      cd.comprar(sc.nextInt());
70                      break;
71                  case 6:
72                      System.out.println("Quantidade a vender do CD...: " +
73                              cd.getDescricao());
74                      cd.vender(sc.nextInt());
75                      break;
76                  case 7:
77                      break;
78              }
79      }
```

Classe: Principal

Linha 80 ao final

```
80              System.out.println();
81              System.out.println("Aguardo um momento.....");
82              Thread.sleep(2000);
83              livro.listarProduto();
84              dvd.listarProduto();
85              cd.listarProduto();
86          }
87
88      }
89
```

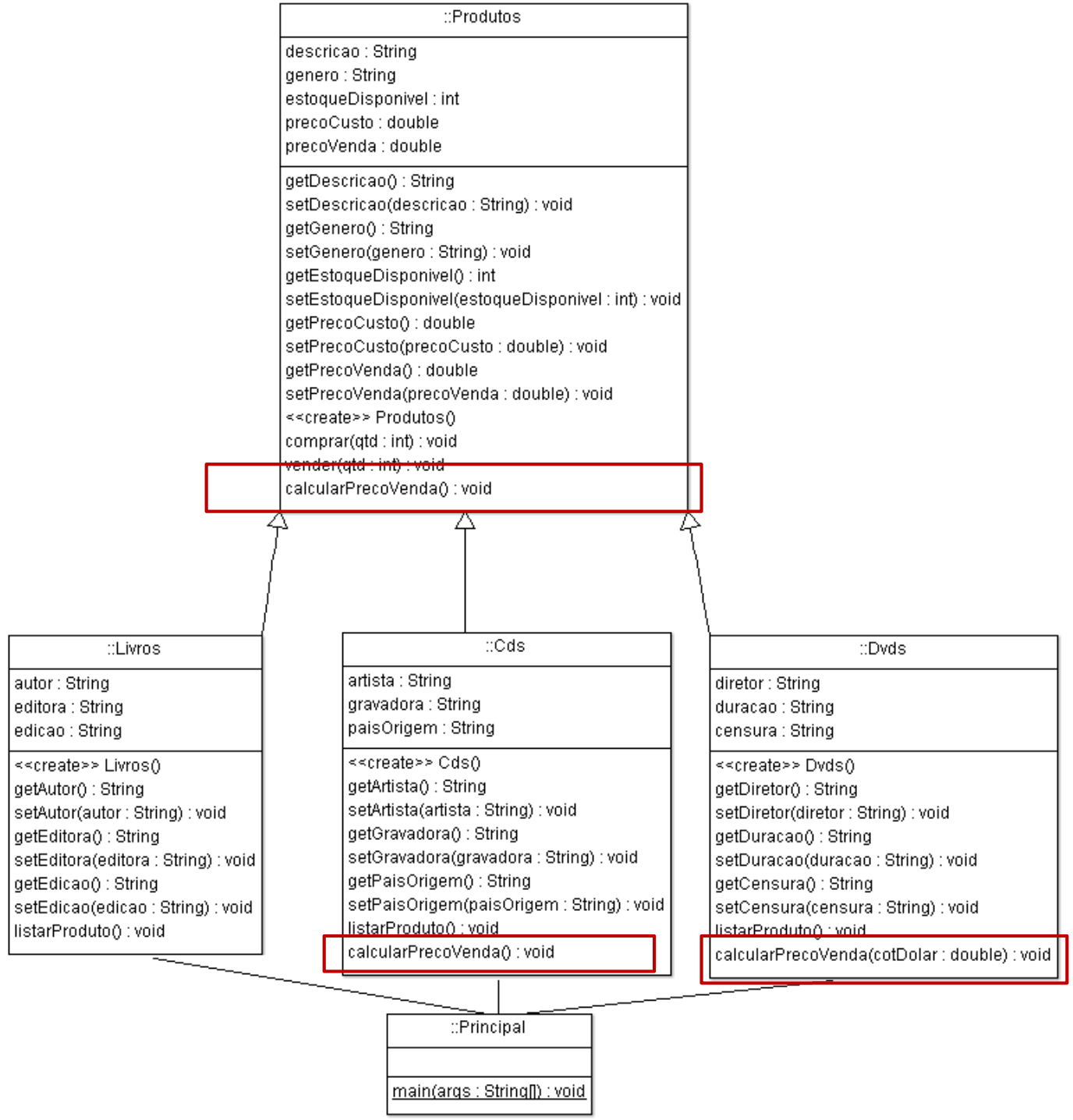
Considerações

- A herança é definida na implementação da subclasse, ou seja, quanto a codificação, a superclasse não apresenta nenhuma diferença.
- O relacionamento de herança é gerado através do comando **extends** que é implementado na assinatura (cabeçalho) da subclasse indicando a sua superclasse.
- Podemos resumir que a *finalidade dos construtores é inicializar os atributos de um objeto na sua instanciação*. Considerando que uma subclasse herda os atributos (além dos métodos) de sua superclasse, devemos portanto, preparar os construtores da subclasse para inicializarem também os atributos herdados.

Sobrescrita
(Overwrite ou Overrinding)

- A sobrescrita ou reescrita de método está diretamente relacionada com herança e é a possibilidade de manter a mesma assinatura de um método herdado e reescrevê-lo na subclasse.
- Na chamada de um método sobrescrito Java considera, primeiro, a classe a partir da qual o objeto foi instanciado, se a superclasse possuir um método com a mesma assinatura este será descartado.

Projeto: Livraria



Regras:

- **Superclasse Produtos:**
- Método calcularPrecoVenda()
  - Calcula 10% sobre o preço de custo e armazena no atributo precoVenda (precoVenda= precoCusto* 1.1).

- **Subclasse Cds:**
- Método sobrescrito calcularPrecoVenda()
  - Calcula 15% sobre o preço de custo e armazena no atributo precoVenda (precoVenda= precoCusto * 1.15).

- **Subclasse Dvds:**
- Método sobrecarregado calcularPrecoVenda(double cotacaoDolar)
  - Calcula o preço de venda utilizando a cotação do dolar + 20%passada por parâmetro (precoVenda = precoCusto * cotacaoDolar * 1.2).

superclasse: Produto

```
60      public void vender(int qtd){
61          //Apresenta o estoque atual. E o estoque atualizado.
62          System.out.println("<---- ESTOQUE ---->");
63          System.out.println("Quantidade do Produto " + this.descricao);
64          System.out.println("Anterior: " + this.estoqueDisponivel);
65          this.estoqueDisponivel-=qtd;
66          System.out.println("Atual    : " + this.estoqueDisponivel);
67      }
        public void calcularPrecoVenda(){
69          this.precoCusto*=1.10;
70      }
71
72
73  }
```

subclasse: Cds

```
33      public void listarProduto(){
34          System.out.println("<---- DADOS DO PRODUTO ---->");
35          System.out.println("Descrição.....: " + this.getDescricao());
36          System.out.println("Gênero.......: " + this.getGenero());
37          System.out.println("Estoque......: " + this.getEstoqueDisponivel());
38          System.out.println("Preço........: R$ " + this.getPrecoCusto());
39          System.out.println("Artista......: " + this.artista);
40          System.out.println("Preco de Venda: " + this.getPrecoVenda());
41          System.out.println("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");
42          System.out.println();
43      }
        @Override
45      public void calcularPrecoVenda(){
46          this.setPrecoVenda(this.getPrecoCusto()*1.15);
47      }
48  }
```

subclasse: Dvds

```
34      public void listarProduto(){
35          System.out.println("<---- DADOS DO PRODUTO ---->");
36          System.out.println("Descrição.....: " + this.getDescricao());
37          System.out.println("Gênero.......: " + this.getGenero());
38          System.out.println("Estoque......: " + this.getEstoqueDisponivel());
39          System.out.println("Preço........: R$ " + this.getPrecoCusto());
40          System.out.println("Diretor......: " + this.diretor);
            System.out.println("Preco de Venda: " + this.getPrecoVenda());
42          System.out.println("=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");
43          System.out.println();
44      }
45      public void calcularPrecoVenda(double cotDolar){
46          this.setPrecoVenda(this.getPrecoCusto()*cotDolar*1.20);
47      }
48  }
```

- **Ressaltando que um objeto do tipo Dvds possuirá dois métodos calcularPrecoVenda, um sem parâmetro (herdado de Produto) e outro recebendo a cotação do dólar (sobrecarregado na classe Dvds)**

Abstração

Classes: Abstratas e Concretas

Métodos: Abstratos e Concretos

Classes abstratas e concretas

- Uma **classe abstrata** é desenvolvida para representar entidades e conceitos abstratos.
- A classe abstrata é sempre uma superclasse que **não possui instâncias**.
- Ela define um modelo para uma funcionalidade e fornece uma implementação incompleta (a parte genérica dessa funcionalidade) que é compartilhada por um grupo de classes derivadas (subclasses).
- Cada uma das classes derivadas completa a funcionalidade da classe abstrata adicionando um comportamento específico.
- Uma **classe concreta** é aquela que pode ser instanciada, ou seja, na prática manipulamos um objeto do seu tipo.

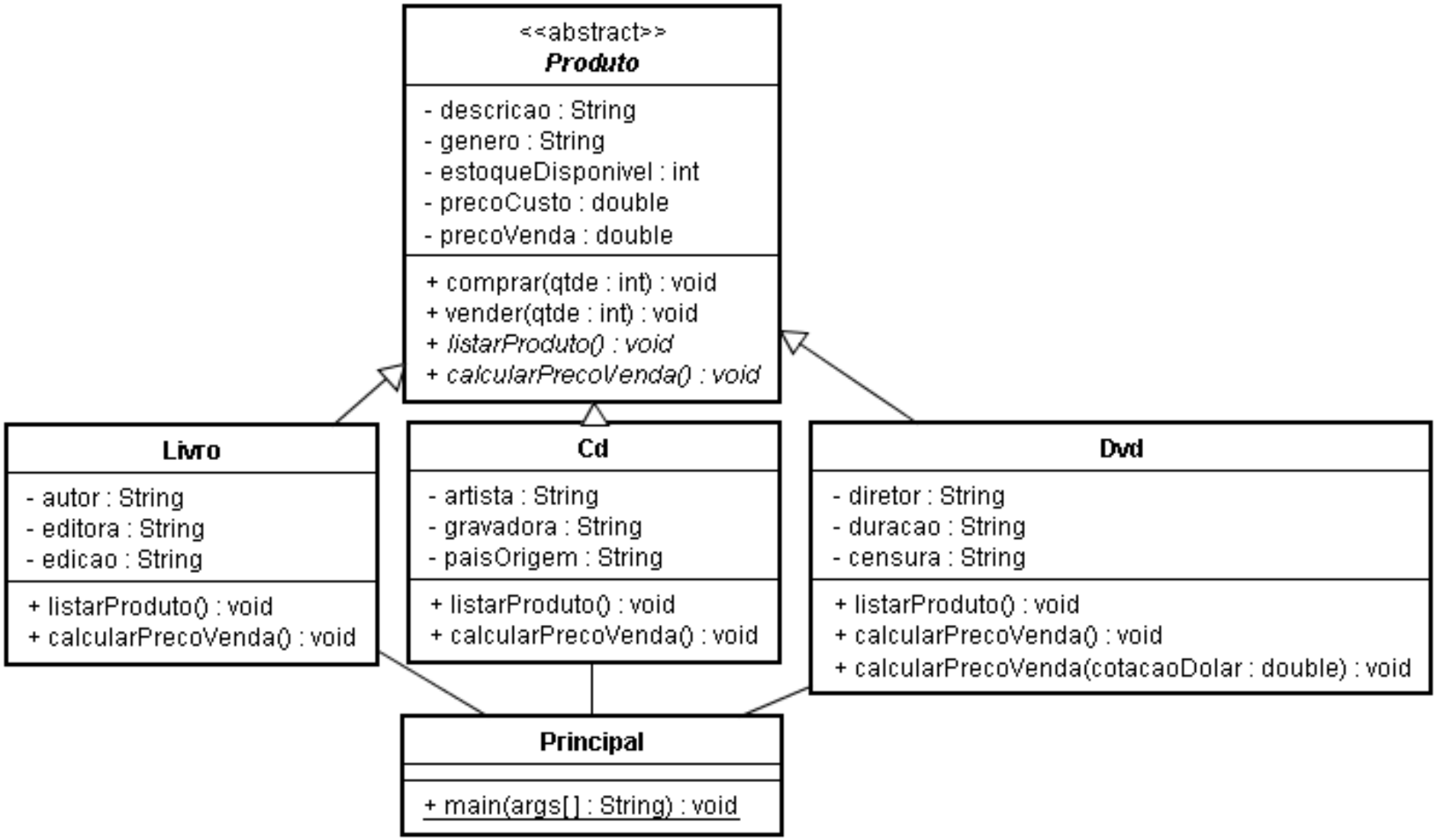
Métodos abstratos

- Uma classe abstrata pode conter métodos concretos, porém, um método abstrato só pode ser definido em uma classe abstrata.
- Esses métodos são implementados nas suas subclasses (concretas) com o objetivo de definir um comportamento (regras) específico.
- Um método abstrato define apenas a assinatura do método e, portanto, não contém código.
- Um método concreto, por sua vez, possui comportamento definido (código escrito).

Métodos abstratos

- **No exemplo da livraria, a classe Produtos nunca será utilizada para instanciar um objeto porque os produtos efetivamente comercializados são livros, cds e dvds.**
- **A finalidade da classe Produtos é somente a generalização dos produtos, portanto, conceitualmente ela deve ser definida como uma classe abstrata.**
- **Partindo desse princípio, o método calcularPrecoVenda também é um forte candidato a ser um método abstrato, porque, nesse exemplo, cada produto tem sua própria regra de cálculo.**

Projeto: Livraria



```java
package modulo05livraria;
//SuperClasse ou Classe Mãe
public abstract class Produtos {
```

superclasse: Produtos

```java
    public abstract void calcularPrecoVenda();

    public abstract void listarProdutos();
```

subclasse: Dvd

```java
 //Método Específico da Classe
public void calcularPrecoVenda(double cotDolar){
    this.setPrecoVenda(this.getPrecoCusto()*cotDolar*1.20);
}

@Override
public void calcularPrecoVenda() {

}
```

Polimorfismo

- O termo polimorfismo é originário do grego e significa "muitas formas" (poli = muitas, morphos = formas).
- O polimorfismo permite que objetos de diferentes subclasses sejam tratados como objetos de uma única superclasse.
- É a possibilidade de manipular um objeto como sendo outro.
- O polimorfismo não quer dizer que o objeto se transforma em outro. Um objeto sempre será do tipo que foi instanciado o que pode mudar é a maneira como nos referimos a ele.

Polimorfismo

- Voltemos ao nosso projeto Livraria.
- Os produtos efetivamente comercializados, e que devem ser gerenciados, são **livro, cd e dvd**.
- Contudo, a afirmação: "Livro, cd e dvd são produtos" está correta.
- O conceito de polimorfismo **possibilita que tratemos um livro, cd ou dvd como um produto** (pois, afinal de contas, eles são produtos), mas sem perdermos as suas características específicas porque apesar de serem produtos eles não deixam de ser um livro, um cd ou um dvd (especializações).

Polimorfismo

- Partindo desse raciocínio podemos definir métodos que reconheçam objetos através de suas superclasses mas que mantenham seus atributos e métodos implementados nas subclasses de origem.
- Nota:
  - Esse tipo de polimorfismo apresentado é o que conhecemos por polimorfismo utilizando-se da hierarquia de classes, em aula futura veremos o conceito sem a utilização de polimorfismo com o uso de Interface.

Nova classe: GerenciadorEstoque

```
2  import java.util.Scanner;
3  public class GerenciarEstoque {
4      public void comprar(Produtos prod)
5      {
6          Scanner sc = new Scanner(System.in);
7          System.out.println("Digite a quantidade comprada: ");
8          int quantidade = sc.nextInt();
9          System.out.println("Estoque anterior: " + prod.getEstoqueDisponivel());
10         int saldo = quantidade + prod.getEstoqueDisponivel();
11         prod.setEstoqueDisponivel(saldo);
12         System.out.println("Estoque atual: " + prod.getEstoqueDisponivel());
13     }
14     public void vender(Produtos prod)
15     {
16         Scanner sc = new Scanner(System.in);
17         System.out.println("Digite a quantidade vendida: ");
18         int quantidade = sc.nextInt();
19         System.out.println("Estoque anterior: " + prod.getEstoqueDisponivel());
20         int saldo =  prod.getEstoqueDisponivel() - quantidade;
21         prod.setEstoqueDisponivel(saldo);
22         System.out.println("Estoque atual: " + prod.getEstoqueDisponivel());
23     }
```

Nova classe: GerenciadorEstoque

```
24      public void encomendar(Produtos prod)
25      {
26          Scanner sc = new Scanner(System.in);
27          System.out.println("Digite a quantidade desejada: ");
28          int quantidade = sc.nextInt();
29          if(quantidade < prod.getEstoqueDisponivel()){
30              System.out.println("Encomenda do produto: " + prod.getDescricao() +
31                      " realizada com pronta entrega!");
32          }else{
33              System.out.println("Encomenda do produto: " + prod.getDescricao() +
34                      " em análise - realizando pedido com fornecedores");
35          }
36      }
37  }
```

classe: Principal

Alterar

```
int opc=0;
while(opc!=7){
    System.out.println("Escolha a opção");
    System.out.println("1--> Operações com Livro");
    System.out.println("2--> Vender Exemplar do Livro");
    System.out.println("3--> Comprar Mais um Título do DVD ");
    System.out.println("4--> Vender Um Título do DVD");
    System.out.println("5--> Comprar Cd");
    System.out.println("6--> Vender Cd");
    System.out.println("7--> Sair");
    opc = sc.nextInt();
    GerenciarEstoque controle = new GerenciarEstoque();
    switch(opc){
        case 1: //livro
            System.out.println("1) Consultar -  2) Comprar - "
                    + "3) Vender - 4) Reajuste - 5) Encomendar");
            int operador = sc.nextInt();
            if(operador==1) livro.listarProdutos();
            if(operador==2) controle.comprar(livro);
            if(operador==3) controle.vender(livro);
            if(operador==4) livro.calcularPrecoVenda();
            if(operador==5) controle.encomendar(livro);
            break;
        case 2:
```

Exercício

- Realize as modificações nos outros itens de menu, para que Cds e Dvds tenham as mesmas funções.

Referências

- Bibliográficas:
  - Mendes – Java com Ênfase em Orientação a Objetos [Exercícios do Capítulo 1]
  - Deitel – Java, como programar – 6º edição.
  - Arnold, Gosling, Holmes – A linguagem de programação Java – 4º edição.
  - Apostilas Caelum
  - Material do Curso de Capacitação Java do CPS
- Internet
  - http://java.sun.com
  - http://www.guj.com.br
  - http://www.portaljava.com

FIM

Obrigado,
Maromo