- Biblioteca

Classe: Livro

1. Interface: recebe preço, disponibilidade

Public

1. Abstrata:

Public abstract class Livro {

Private string código;

}

1. Real: 1989, romance, 39.90, melhor do que nos filmes (instancias)

Classe: Cliente

1. Interface: compra livro, faz cadastro, paga multa
2. Abstrata: data nasc, cpf, telefone, nome

Public abstract class Livro {

Private String cpf;

}

1. Real: 2004, 000.000.000-00, 61 902101989, Taylor Scott (instancias)

Classe: Funcionário

1. Interface: registra livro, faz cadastro
2. Abstrata: data nasc, cpf, telefone, nome

Public abstract class Livro {

Private String cpf;

}

1. Real: 2004, 000.000.000-00, 61 902101989, Louis Styles (instancias)

Classe: Estoque

1. Interface: armazena livro

// Interface para Livro

interface Livro {

void receberPreco(double preco);

void verificarDisponibilidade();

}

// Classe abstrata para Livro

abstract class LivroBase implements Livro {

private String codigo;

// Método construtor

public LivroBase(String codigo) {

this.codigo = codigo;

}

// Implementação dos métodos da interface Livro

@Override

public void receberPreco(double preco) {

System.out.println("Preço do livro definido como: R$" + preco);

}

@Override

public void verificarDisponibilidade() {

System.out.println("Livro está disponível para compra.");

}

}

// Classe para Livro (Real)

class LivroReal extends LivroBase

class LivroReal extends LivroBase {

private int anoPublicacao;

private String genero;

private double preco;

private String titulo;

// Método construtor

public LivroReal(String codigo, int anoPublicacao, String genero, double preco, String titulo) {

super(codigo);

this.anoPublicacao = anoPublicacao;

this.genero = genero;

this.preco = preco;

this.titulo = titulo;

}

// Getters e setters

// Implementação dos métodos da classe LivroBase, se necessário

}

// Interface para Cliente

interface Cliente {

void comprarLivro();

void fazerCadastro();

void pagarMulta();

}

// Classe abstrata para Cliente

abstract class ClienteBase implements Cliente {

private String dataNascimento;

private String cpf;

private String telefone;

private String nome;

// Método construtor

public ClienteBase(String dataNascimento, String cpf, String telefone, String nome) {

this.dataNascimento = dataNascimento;

this.cpf = cpf;

this.telefone = telefone;

this.nome = nome;

}

// Implementação dos métodos da interface Cliente, se necessário

}

// Classe para Cliente (Real)

class ClienteReal extends ClienteBase {

// Método construtor

public ClienteReal(String dataNascimento, String cpf, String telefone, String nome) {

super(dataNascimento, cpf, telefone, nome);

}

// Implementação dos métodos da interface Cliente, se necessário

}

// Interface para Funcionário

interface Funcionario {

void registrarLivro();

void fazerCadastro();

}

// Classe abstrata para Funcionário

abstract class FuncionarioBase implements Funcionario {

private String dataNascimento;

private String cpf;

private String telefone;

private String nome;

// Método construtor

public FuncionarioBase(String dataNascimento, String cpf, String telefone, String nome) {

this.dataNascimento = dataNascimento;

this.cpf = cpf;

this.telefone = telefone;

this.nome = nome;

}

// Implementação dos métodos da interface Funcionario, se necessário

}

// Classe para Funcionário (Real)

class FuncionarioReal extends FuncionarioBase {

// Método construtor

public FuncionarioReal(String dataNascimento, String cpf, String telefone, String nome) {

super(dataNascimento, cpf, telefone, nome);

}

// Implementação dos métodos da interface Funcionario, se necessário

}

// Interface para Estoque

interface Estoque {

void armazenarLivro();

}

// Classe para Estoque

class EstoqueReal implements Estoque {

// Implementação do método da interface Estoque

@Override

public void armazenarLivro() {

System.out.println("Livro armazenado no estoque.");

}

}