**Adapter**

1)

interface Conversor{

double celsius-Fahrenheit(double celsius

double fahrenheit-Celsius(double fahrenheit);

}

class Celsius {

public double converterParaFahrenheit(double celsius) {

return (celsius \* 9/5) + 32;

}

}

class AdapterTemperatura implements Conversor{

private Celsius conversorCelsius;

public AdaptadorTemperatura() {

this. Celsius = new Celsius();

}

@Override public double celsius-Fahrenheit(double celsius) { return conversorCelsius.converter-Fahrenheit(celsius);

}

@Override public double fahrenheit-Celsius(double fahrenheit) {

return (fahrenheit - 32) \* 5/9;

}

}

2)

interface ManipuladorTransacao {

void realizarTransacao(double quantia);

}

// Sistema antigo

class SistemaFinanceiroAntigo {

public void registrarTransacao(String quantia) {

System.out.println("Transação antiga registrada: R$" + quantia);

}

}

// Teste de funcionamento

public class TesteTransacao {

public static void main(String[] args) {

System.out.println("=== Teste do sistema ===");

// Usando o adaptador

ManipuladorTransacao manipulador = new AdaptadorTransacao();

// Testes

System.out.println("\n1. Transação de R$ 100,00:");

manipulador.realizarTransacao(100.00);

System.out.println("\n2. Transação de R$ 55,75:");

manipulador.realizarTransacao(55.75);

System.out.println("\n3. Transação de R$ 1234,56:");

manipulador.realizarTransacao(1234.56);

// Teste com o sistema legado

SistemaFinanceiroAntigo antigo = new SistemaFinanceiroAntigo();

System.out.println("Sistema antigo direto:");

antigo.registrarTransacao("100,00");

}

}

// Adaptador

class AdaptadorTransacao implements ManipuladorTransacao {

private SistemaFinanceiroAntigo sistemaAntigo;

public AdaptadorTransacao() {

this.sistemaAntigo = new SistemaFinanceiroAntigo();

}

@Override

public void realizarTransacao(double quantia) {

String quantiaFormatada = String.format("%.2f", quantia);

sistemaAntigo.registrarTransacao(quantiaFormatada);

}

}