ANALISE E DESENVOLVIMENTO DE SISTEMAS

ANDREI TEIXEIRA RODRIGUES

TÍTULO DO TRABALHO:

Relatório de Aula Prática - Linguagem Orientada a Objetos

Novo Hamburgo RS

2025

ANDREI TEIXEIRA RODRIGUES

TÍTULO DO TRABALHO:

Relatório de Aula Prática - Linguagem Orientada a Objetos

Trabalho de portfólio apresentado como requisito parcial para a obtenção de média semestral.

Orientadora: Vinicius Mendes Gomes da Silva

Novo Hamburgo RS
2025

SUMÁRIO

1 INTRODUÇÃO

A Programação Orientada a Objetos (POO) é um dos paradigmas mais utilizados no desenvolvimento de software, proporcionando uma forma eficiente de organizar e estruturar o código. Com ela, é possível representar entidades do mundo real por meio de objetos que possuem atributos (características) e métodos (comportamentos). Esse paradigma facilita a manutenção e a escalabilidade do software, além de promover um código mais modular e reutilizável.

Nesta aula prática, o objetivo é aplicar os conceitos fundamentais da POO para o desenvolvimento de uma aplicação simples de gerenciamento bancário. O sistema será projetado para permitir que o usuário informe seus dados pessoais (como nome, sobrenome e CPF) e realize operações bancárias básicas, como consulta de saldo, depósitos e saques. Durante a atividade, os estudantes serão desafiados a empregar práticas de instanciação de classes Java, utilização de métodos e atributos, além de explorar o uso de estruturas de controle no desenvolvimento da aplicação.

A implementação dessa aplicação ajudará a consolidar o conhecimento sobre a criação e manipulação de objetos em Java, utilizando os principais conceitos da programação orientada a objetos, como encapsulamento, herança e polimorfismo, além de reforçar a importância de se criar sistemas bem estruturados e organizados.

2 MÉTODOS

1. Instalação do ambiente Apache NetBeans IDE 22

2. Classe Cliente:

3. Contém os atributos nome, sobrenome, cpf, e saldo para representar as informações do cliente e o saldo bancário.

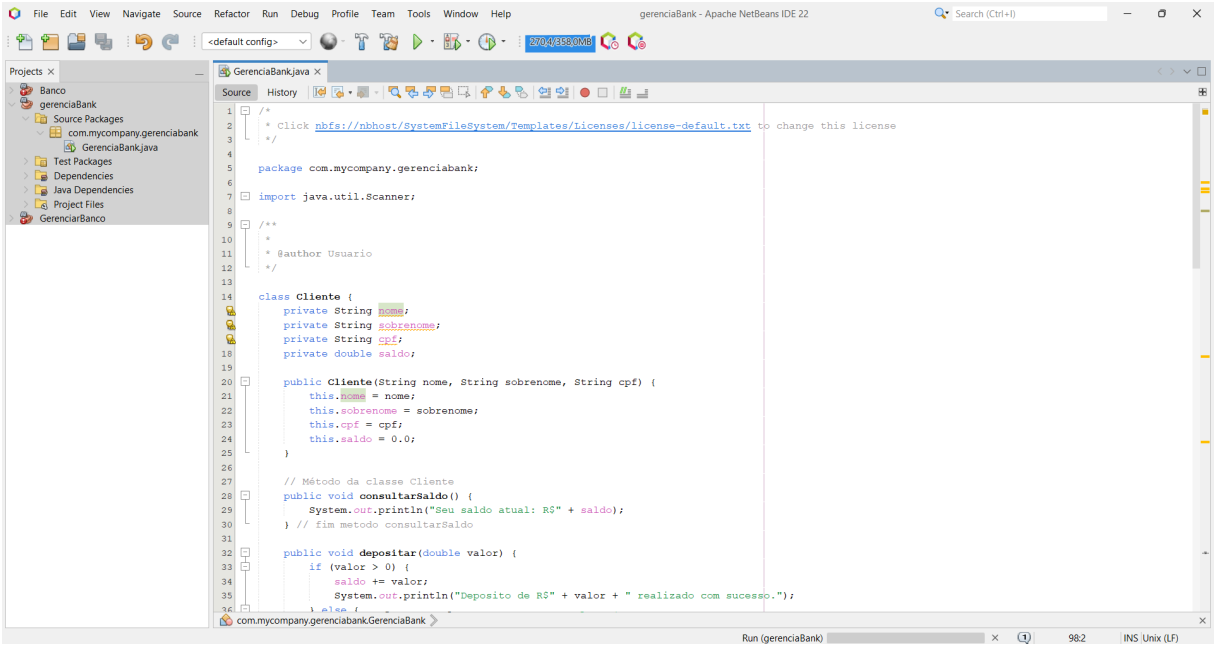
4. Implementa métodos como consultarSaldo(), depositar(double valor) e sacar(double valor) para realizar as operações bancárias.
5. Classe GerenciaBank:

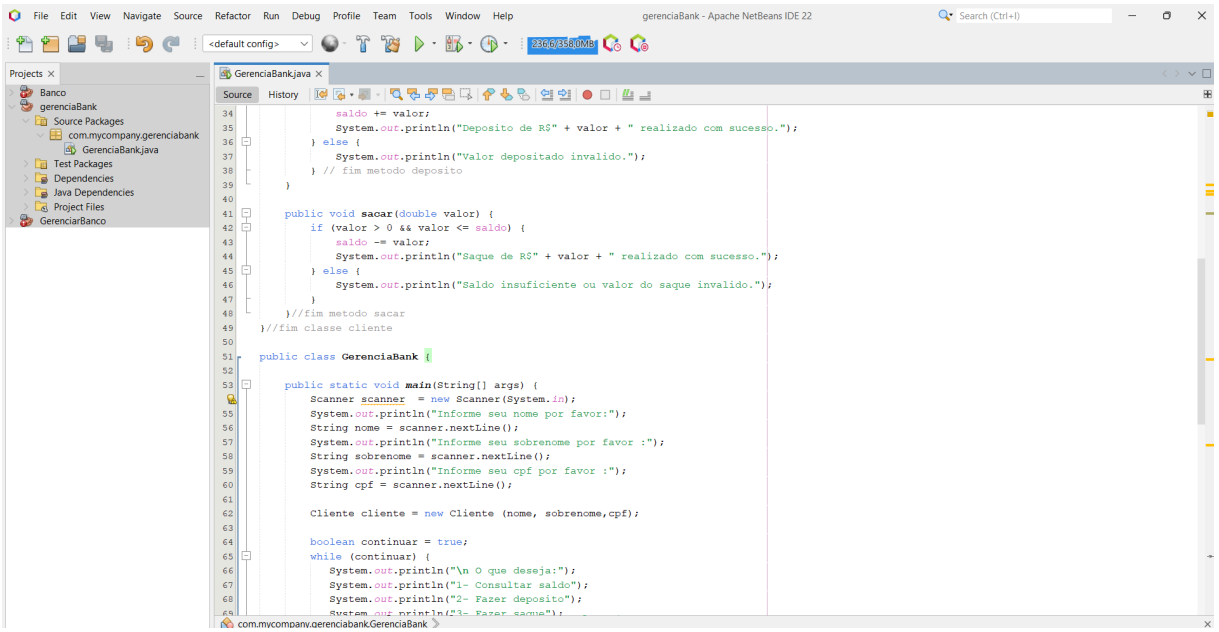
6. A classe principal que contém o método main().

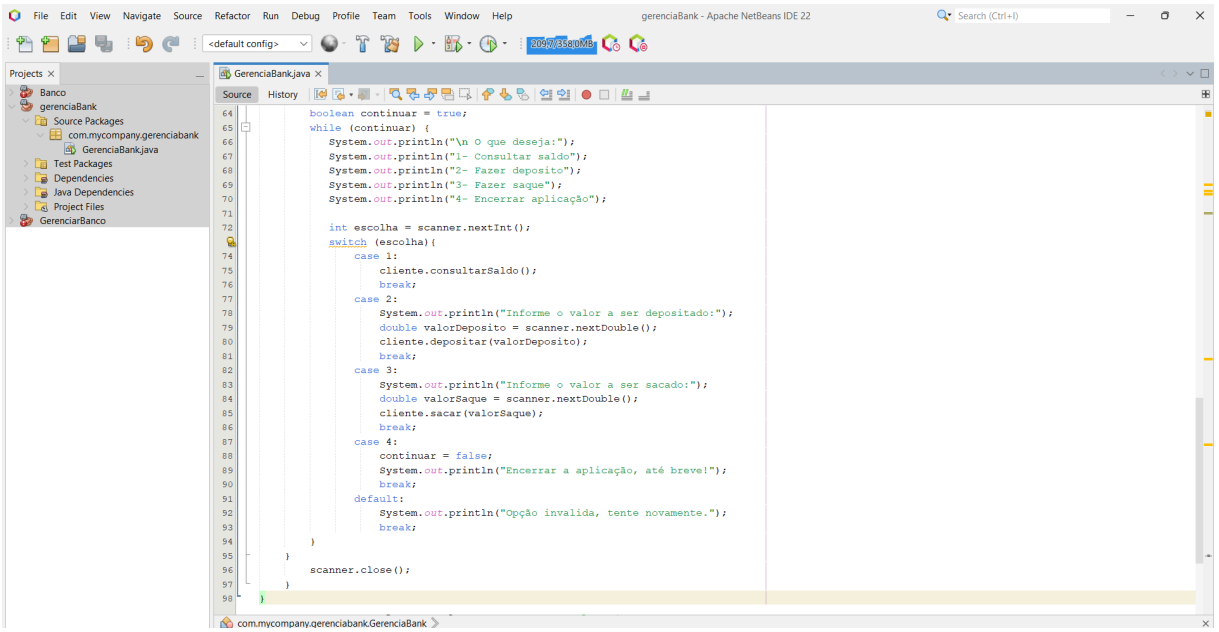
7. Recebe dados do cliente e cria uma instância da classe Cliente.

8. Exibe um menu com opções para o usuário interagir, chamando os métodos apropriados da classe Cliente.

3 RESULTO


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
gerenciaBank - Apache NetBeans IDE 22
Search (Ctrl+I)
<default config>
Projects
Banco
gerenciaBank
Source Packages
com.mycompany.gerenciabank
GerenciaBank.java
Test Packages
Dependencies
Java Dependencies
Project Files
GerenciarBanco
GerenciaBank.java
Source
History
/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
*/
package com.mycompany.gerenciabank;
import java.util.Scanner;
/**
*
* @author Usuario
*/
class Cliente {
private String nome;
private String sobrenome;
private String cpf;
private double saldo;
public Cliente(String nome, String sobrenome, String cpf) {
this.nome = nome;
this.sobrenome = sobrenome;
this.cpf = cpf;
this.saldo = 0.0;
}
// Método da classe Cliente
public void consultarSaldo() {
System.out.println("Seu saldo atual: R$" + saldo);
} // fim metodo consultarSaldo
public void depositar(double valor) {
if (valor > 0) {
saldo += valor;
System.out.println("Deposito de R$" + valor + " realizado com sucesso.");
com.mycompany.gerenciabank.GerenciaBank
Run (gerenciaBank)
98:2
INS Unix (LF)


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
gerenciaBank - Apache NetBeans IDE 22
Search (Ctrl+I)
<default config>
Projects
Banco
gerenciaBank
Source Packages
com.mycompany.gerenciabank
GerenciaBank.java
Test Packages
Dependencies
Java Dependencies
Project Files
GerenciarBanco
GerenciaBank.java
Source
History
saldo += valor;
System.out.println("Deposito de R$" + valor + " realizado com sucesso.");
} else {
System.out.println("Valor depositado invalido.");
} // fim metodo deposito
}
public void sacar(double valor) {
if (valor > 0 && valor <= saldo) {
saldo -= valor;
System.out.println("Saque de R$" + valor + " realizado com sucesso.");
} else {
System.out.println("Saldo insuficiente ou valor do saque invalido.");
}
}//fim metodo sacar
}//fim classe cliente
public class GerenciaBank {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.println("Informe seu nome por favor:");
String nome = scanner.nextLine();
System.out.println("Informe seu sobrenome por favor :");
String sobrenome = scanner.nextLine();
System.out.println("Informe seu cpf por favor :");
String cpf = scanner.nextLine();
Cliente cliente = new Cliente (nome, sobrenome,cpf);
boolean continuar = true;
while (continuar) {
System.out.println("\n O que deseja:");
System.out.println("1- Consultar saldo");
System.out.println("2- Fazer deposito");
com.mycompany.gerenciabank.GerenciaBank



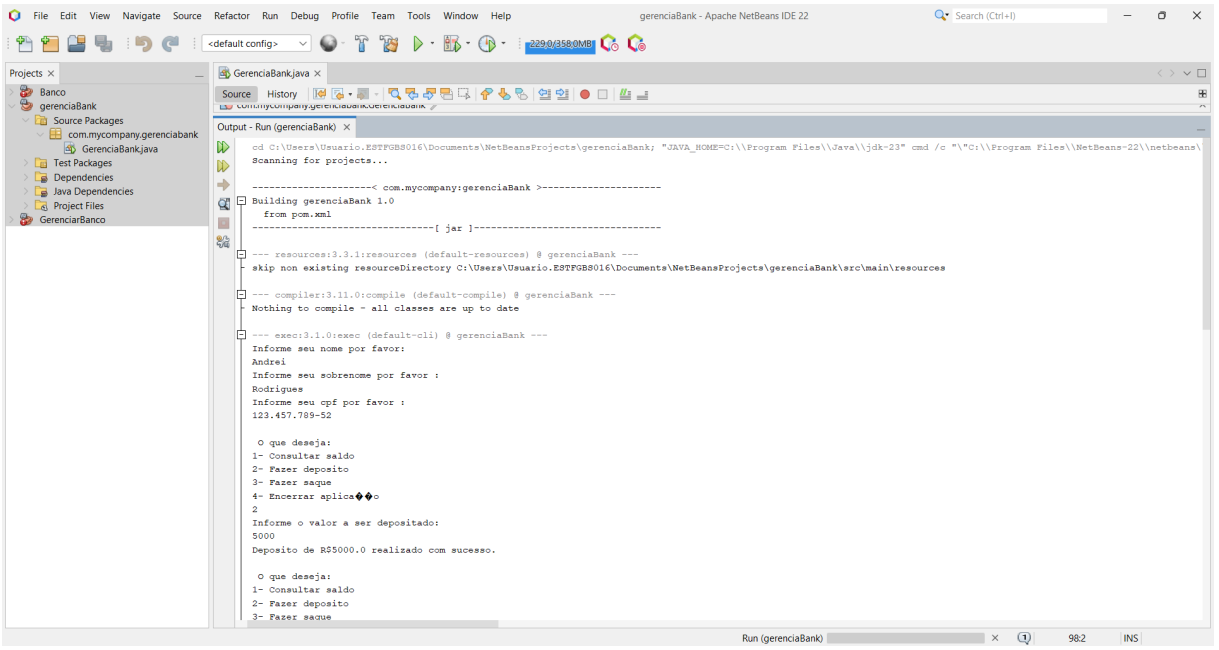
Código:

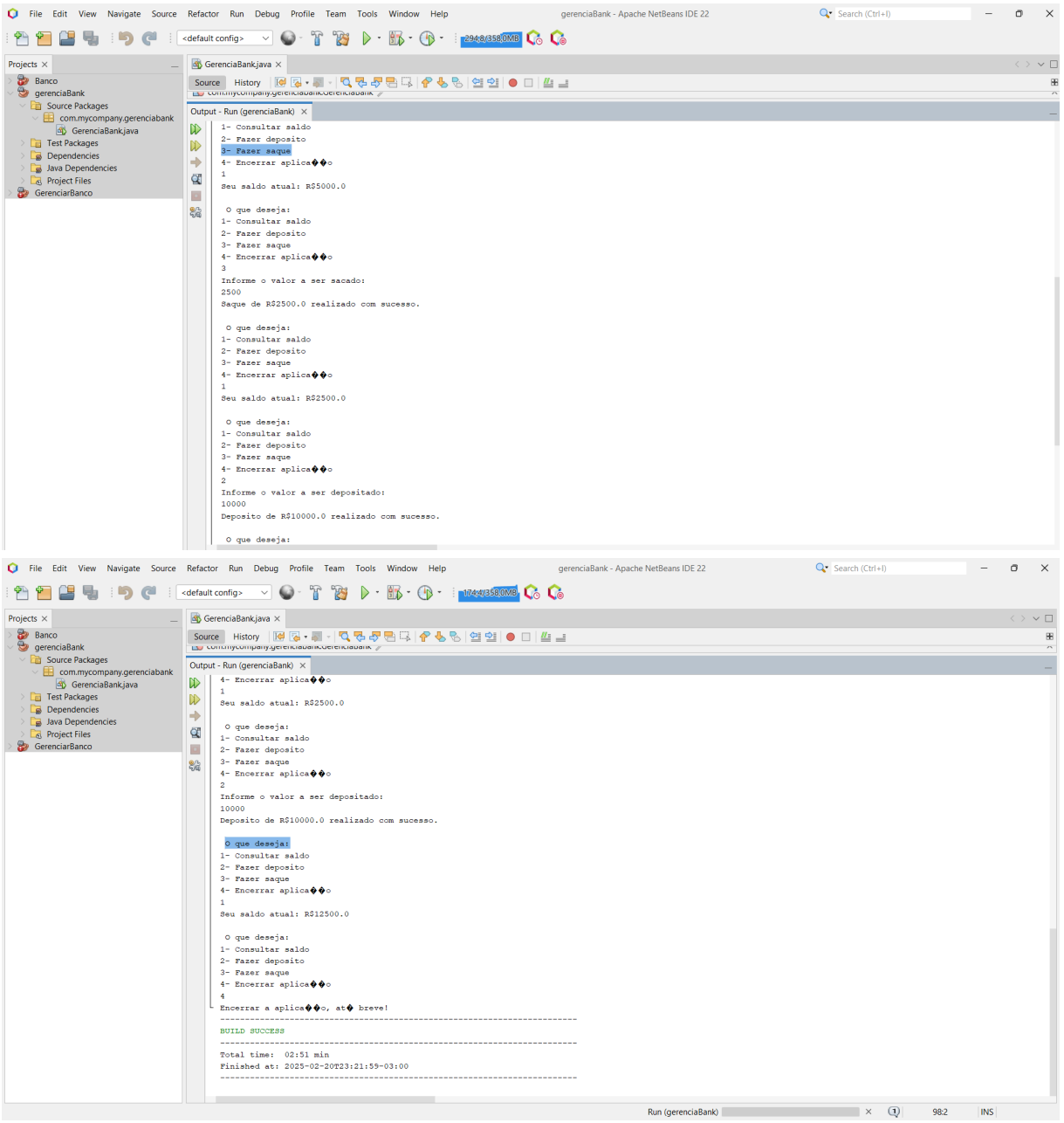
```
/*
   *  Click  nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt  to change this license
 */

package com.mycompany.gerenciabank;

import java.util.Scanner;

/**
 *
 * @author Usuario
 */

class Cliente {
    private String nome;
    private String sobrenome;
    private String cpf;
    private double saldo;

    public Cliente(String nome, String sobrenome, String cpf) {
        this.nome = nome;
        this.sobrenome = sobrenome;
        this.cpf = cpf;
        this.saldo = 0.0;
    }

    // Método da classe Cliente
    public void consultarSaldo() {
        System.out.println("Seu saldo atual: R$" + saldo);
```

```
    } // fim metodo consultarSaldo

    public void depositar(double valor) {
        if (valor > 0) {
            saldo += valor;
            System.out.println("Deposito de R$" + valor + " realizado com sucesso.");
        } else {
            System.out.println("Valor depositado invalido.");
        } // fim metodo deposito
    }

    public void sacar(double valor) {
        if (valor > 0 && valor <= saldo) {
            saldo -= valor;
            System.out.println("Saque de R$" + valor + " realizado com sucesso.");
        } else {
            System.out.println("Saldo insuficiente ou valor do saque invalido.");
        }
    }//fim metodo sacar
}//fim classe cliente

public class GerenciaBank {

    public static void main(String[] args) {
        Scanner scanner  = new Scanner(System.in);
        System.out.println("Informe seu nome por favor:");
        String nome = scanner.nextLine();
        System.out.println("Informe seu sobrenome por favor :");
        String sobrenome = scanner.nextLine();
        System.out.println("Informe seu cpf por favor :");
        String cpf = scanner.nextLine();

        Cliente cliente = new Cliente (nome, sobrenome,cpf);

        boolean continuar = true;
        while (continuar) {
          System.out.println("\n O que deseja:");
          System.out.println("1- Consultar saldo");
          System.out.println("2- Fazer deposito");
          System.out.println("3- Fazer saque");
          System.out.println("4- Encerrar aplicação");

          int escolha = scanner.nextInt();
          switch (escolha){
              case 1:
                  cliente.consultarSaldo();
                  break;
              case 2:
                  System.out.println("Informe o valor a ser depositado:");
                  double valorDeposito = scanner.nextDouble();
```

```
                cliente.depositar(valorDeposito);
                break;
            case 3:
                System.out.println("Informe o valor a ser sacado:");
                double valorSaque = scanner.nextDouble();
                cliente.sacar(valorSaque);
                break;
            case 4:
                continuar = false;
                System.out.println("Encerrar a aplicação, até breve!");
                break;
            default:
                System.out.println("Opção invalida, tente novamente.");
                break;
        }
    }
        scanner.close();
    }
}
```

Funcionamento do código:





Resultado:

cd C:\Users\Usuario.ESTFGBS016\Documents\NetBeansProjects\gerenciaBank; "JAVA_HOME=C:\\Program Files\\Java\\jdk-23" cmd /c "\"C:\\Program Files\\NetBeans-22\\netbeans\\java\\maven\\bin\\mvn.cmd\" -Dexec.vmArgs= \"-Dexec.args=${exec.vmArgs} -classpath %classpath ${exec.mainClass} ${exec.appArgs}\" -Dexec.appArgs= -Dexec.mainClass=com.mycompany.gerenciabank.GerenciaBank \"-Dexec.executable=C:\\Program Files\\Java\\jdk-23\\bin\\java.exe\" \"-Dmaven.ext.class.path=C:\\Program Files\\NetBeans-22\\netbeans\\java\\maven-nblib\\netbeans-eventspy.jar\" --no-transfer-progress process-classes org.codehaus.mojo:exec-maven-plugin:3.1.0:exec"
Scanning for projects...

```
----------------------< com.mycompany:gerenciaBank >----------------------
Building gerenciaBank 1.0
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ gerenciaBank ---
skip non existing resourceDirectory
C:\Users\Usuario.ESTFGBS016\Documents\NetBeansProjects\gerenciaBank\src\mai
n\resources

--- compiler:3.11.0:compile (default-compile) @ gerenciaBank ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ gerenciaBank ---
Informe seu nome por favor:
Andrei
Informe seu sobrenome por favor :
Rodrigues
Informe seu cpf por favor :
123.457.789-52

 O que deseja:
1- Consultar saldo
2- Fazer deposito
3- Fazer saque
4- Encerrar aplica��o
2
Informe o valor a ser depositado:
5000
Deposito de R$5000.0 realizado com sucesso.

 O que deseja:
1- Consultar saldo
2- Fazer deposito
3- Fazer saque
4- Encerrar aplica��o
1
Seu saldo atual: R$5000.0

 O que deseja:
1- Consultar saldo
2- Fazer deposito
3- Fazer saque
4- Encerrar aplica��o
3
Informe o valor a ser sacado:
2500
Saque de R$2500.0 realizado com sucesso.
```

O que deseja:
1- Consultar saldo
2- Fazer deposito
3- Fazer saque
4- Encerrar aplica��o
1
Seu saldo atual: R$2500.0

O que deseja:
1- Consultar saldo
2- Fazer deposito
3- Fazer saque
4- Encerrar aplica��o
2
Informe o valor a ser depositado:
10000
Deposito de R$10000.0 realizado com sucesso.

O que deseja:
1- Consultar saldo
2- Fazer deposito
3- Fazer saque
4- Encerrar aplica��o
1
Seu saldo atual: R$12500.0

O que deseja:
1- Consultar saldo
2- Fazer deposito
3- Fazer saque
4- Encerrar aplica��o
4
Encerrar a aplica��o, at� breve!

BUILD SUCCESS

Total time: 02:51 min
Finished at: 2025-02-20T23:21:59-03:00
---

4 CONCLUSÃO

A aplicação desenvolvida para o gerenciamento bancário proporcionou uma excelente oportunidade para a prática dos conceitos fundamentais da Programação Orientada a Objetos (POO). Ao implementar uma estrutura que envolve encapsulamento, métodos e atributos, foi possível criar um sistema funcional que permite ao usuário realizar operações básicas como consultar saldo, depositar e sacar dinheiro, de maneira simples e eficiente.

Através da construção da classe Cliente, foi possível isolar a lógica de negócios, como o controle de saldo e a validação de operações, promovendo um código mais modular, reutilizável e de fácil manutenção. A interação com o usuário foi realizada de forma clara por meio de um menu interativo, utilizando estruturas de controle adequadas, como switch-case, para garantir uma navegação intuitiva.

Além disso, a implementação da classe principal GerenciaBank demonstrou a importância de uma interface simples para facilitar a experiência do usuário, e o uso de estruturas de repetição e controle para garantir que o programa continue funcionando até que o usuário decida encerrá-lo.

Esse projeto reforça a importância de utilizar boas práticas de programação e organização do código para o desenvolvimento de sistemas mais complexos no futuro, além de proporcionar uma base sólida para avançar no aprendizado e no uso do paradigma orientado a objetos em outras linguagens e projetos.

Por fim, a aplicação foi implementada com sucesso, cumprindo os requisitos propostos, e oferece uma visão clara de como conceitos de POO podem ser aplicados para construir sistemas funcionais e escaláveis.

REFERÊNCIAS

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

Este livro é fundamental para entender os padrões de projeto em sistemas orientados a objetos. Ele aborda uma série de padrões que ajudam na criação de sistemas mais robustos e reutilizáveis.

LARMAN, C. Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientados a Objetos. 3ª ed. Porto Alegre: Bookman, 2007.

Um ótimo material para aprender sobre a aplicação de UML e como estruturar sistemas orientados a objetos, além de explicar conceitos importantes de análise e design de sistemas.

MARTIN, R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008.

Este livro é uma excelente referência sobre boas práticas de programação e como escrever códigos limpos e eficientes, algo fundamental ao trabalhar com a Programação Orientada a Objetos.

KERNIGHAN, B.; RITCHIE, D. M. The C Programming Language. 2ª ed. Prentice Hall, 1988.

Embora seja um clássico voltado para a linguagem C, muitos conceitos de programação podem ser aplicados ao Java. É uma boa leitura para quem deseja compreender os fundamentos da programação.

HORSTMANN, C. Core Java Volume I – Fundamentals. 11ª ed. Prentice Hall, 2018.

Um excelente livro para quem deseja aprender a fundo a linguagem Java. Cobre os conceitos fundamentais de Java, incluindo a programação orientada a objetos e a criação de sistemas reais com a linguagem.

Um dos livros mais clássicos sobre programação, que explora desde os conceitos mais básicos de lógica de programação até tópicos mais avançados. Embora seja sobre C, é uma excelente introdução aos conceitos de programação de sistemas.