

**Universidade Pitágoras Unopar**

**Anhanguera**

**ENGENHARIA DE SOFTWARE**

**Elisson Nadson Souza Marques - RA 4372521401**

**PORTFÓLIO - RELATÓRIO DE AULA PRÁTICA**

NOME DA DISCIPLINA: Linguagem Orientada a Objetos

**Elisson Nadson Souza Marques - RA 4372521401**

**PORTFÓLIO - RELATÓRIO DE AULA PRÁTICA**

NOME DA DISCIPLINA: Linguagem Orientada a Objetos

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

Orientador: Vinicius Camargo Prattes

# INTRODUÇÃO

Neste trabalho, desenvolvi uma aplicação prática utilizando **Programação Orientada a Objetos (POO)**, com o objetivo de implementar um sistema de gerenciamento bancário.

A aplicação foi criada utilizando a linguagem **Java** e desenvolvida no **IntelliJ IDEA 2024.2.4**. O sistema permite que o usuário informe seu nome, sobrenome e CPF, além de realizar operações bancárias como consulta de saldo, depósitos, saques e encerramento da aplicação.

A aplicação é modular e segue os princípios de POO, com destaque para o encapsulamento e organização do código. Essas características tornam o sistema escalável e de fácil manutenção.

# OBJETIVO

O objetivo geral deste projeto foi desenvolver uma aplicação funcional e interativa que permitisse:

1. Cadastro de informações do usuário (nome, sobrenome e CPF).
2. Consultar saldo da conta bancária.
3. Realizar depósitos e saques.
4. Encerrar o programa de maneira segura e amigável.

# DESENVOLVIMENTO

* **Preparação do Ambiente**
* **IDE Utilizada**: IntelliJ IDEA 2024.2.4.
* **JDK**: Foi utilizado o Java Development Kit (JDK), devidamente configurado no IntelliJ.
* **Estrutura do Projeto**:
  + O projeto foi estruturado em dois arquivos:
    1. Main.java: Classe principal, que contém o método main e o menu de interação com o usuário.
    2. ContaBancaria.java: Classe responsável pelos dados do usuário e operações bancárias.
* **Construção do Código**

1. **Classe Principal (Main.java)**:
   * Gerencia o fluxo da aplicação e exibe o menu de opções.
   * Utiliza um loop do-while para repetir o menu até que o usuário escolha encerrar o sistema.
   * Implementa as seguintes operações:
     + **Consultar saldo**: Exibe o saldo atual da conta.
     + **Depositar**: Solicita um valor ao usuário e o adiciona ao saldo.
     + **Sacar**: Solicita um valor e verifica se há saldo suficiente antes de realizar o saque.
     + **Encerrar**: Finaliza o programa com uma mensagem de agradecimento.
2. **Classe ContaBancaria**:
   * Encapsula os dados do usuário (nome, sobrenome, CPF e saldo).
   * Possui métodos para:
     + Definir os dados do usuário (setNome, setSobrenome, setCpf).
     + Consultar saldo (getSaldo).
     + Realizar depósito (depositar).
     + Realizar saque (sacar), verificando se há saldo suficiente.

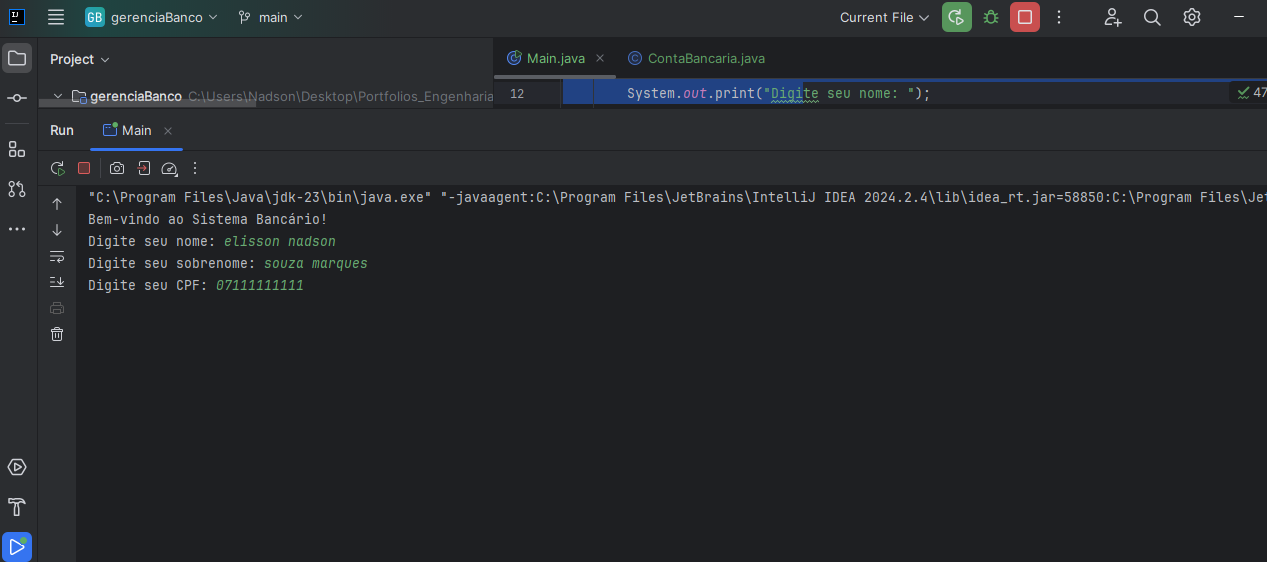
# RESULTADOS

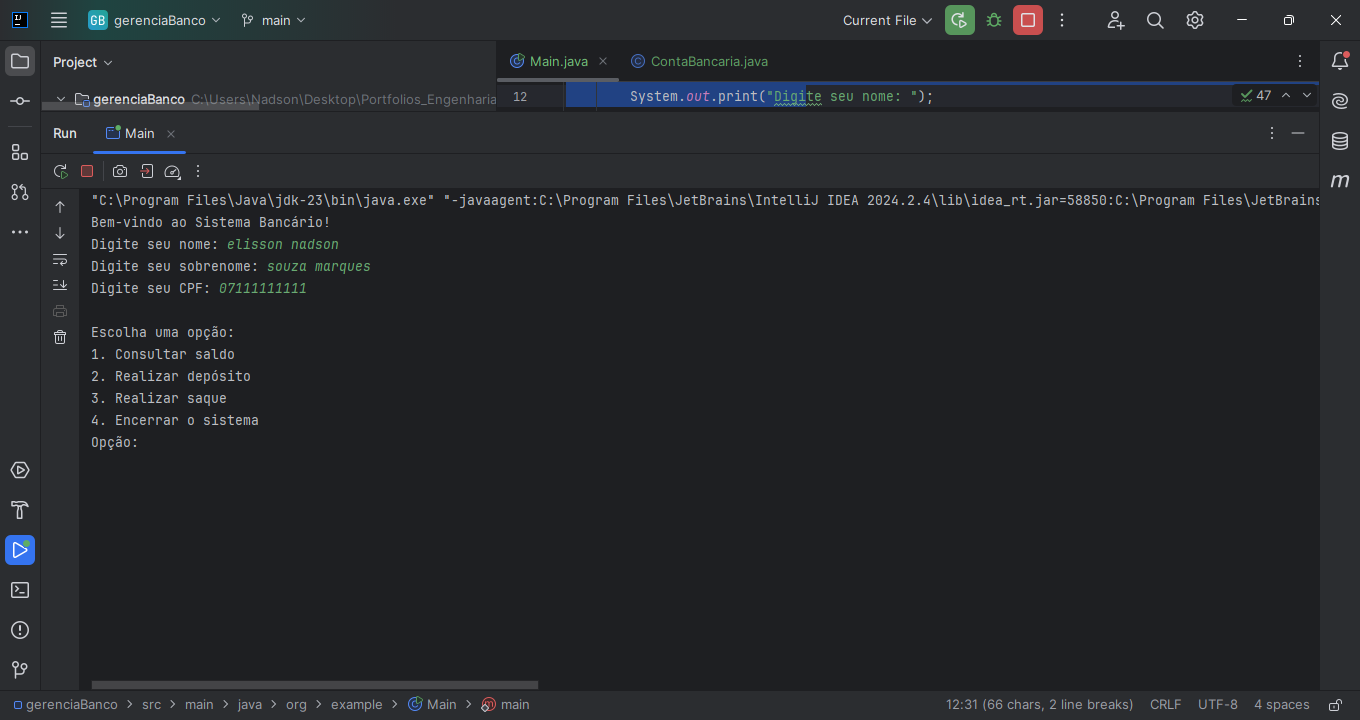
A aplicação foi executada com sucesso, e todas as funcionalidades foram validadas. O sistema mostrou-se funcional, com as seguintes características principais:

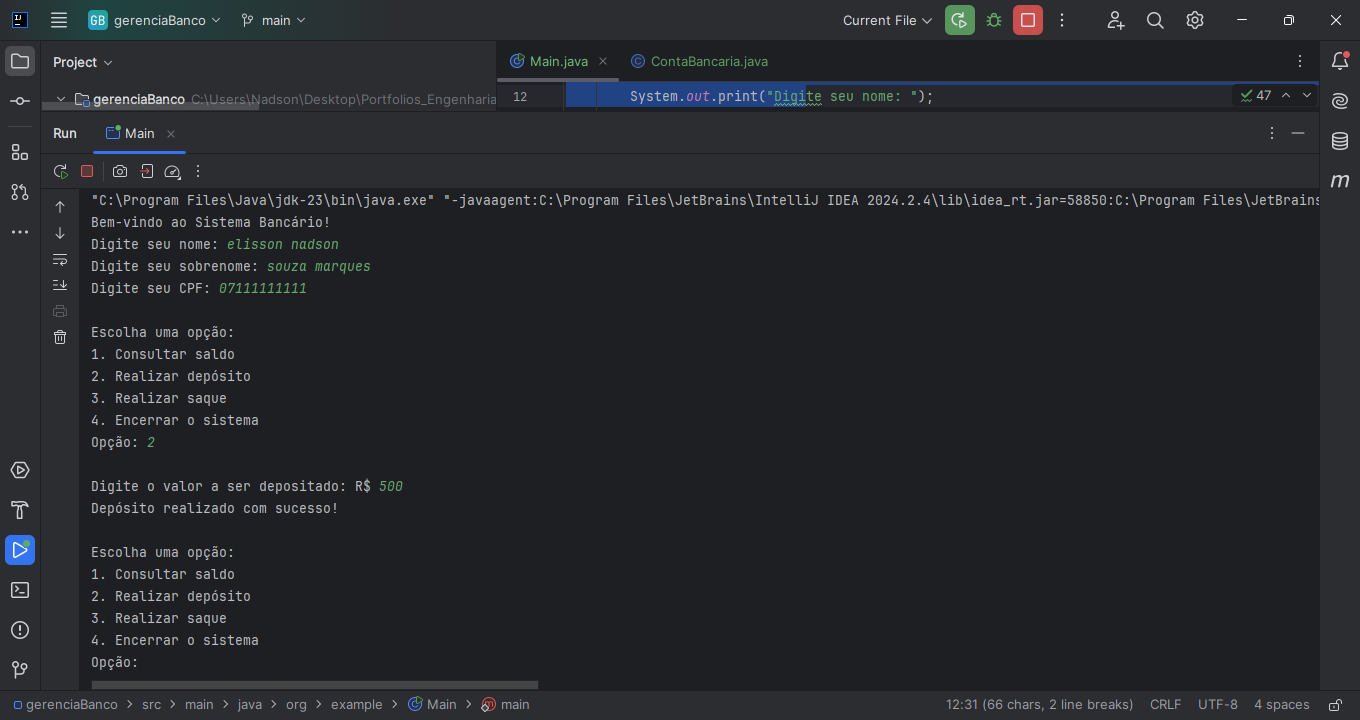
* Cadastro de dados do usuário.
* Operações bancárias básicas, como depósito, saque e consulta de saldo.
* Encerramento seguro da aplicação.

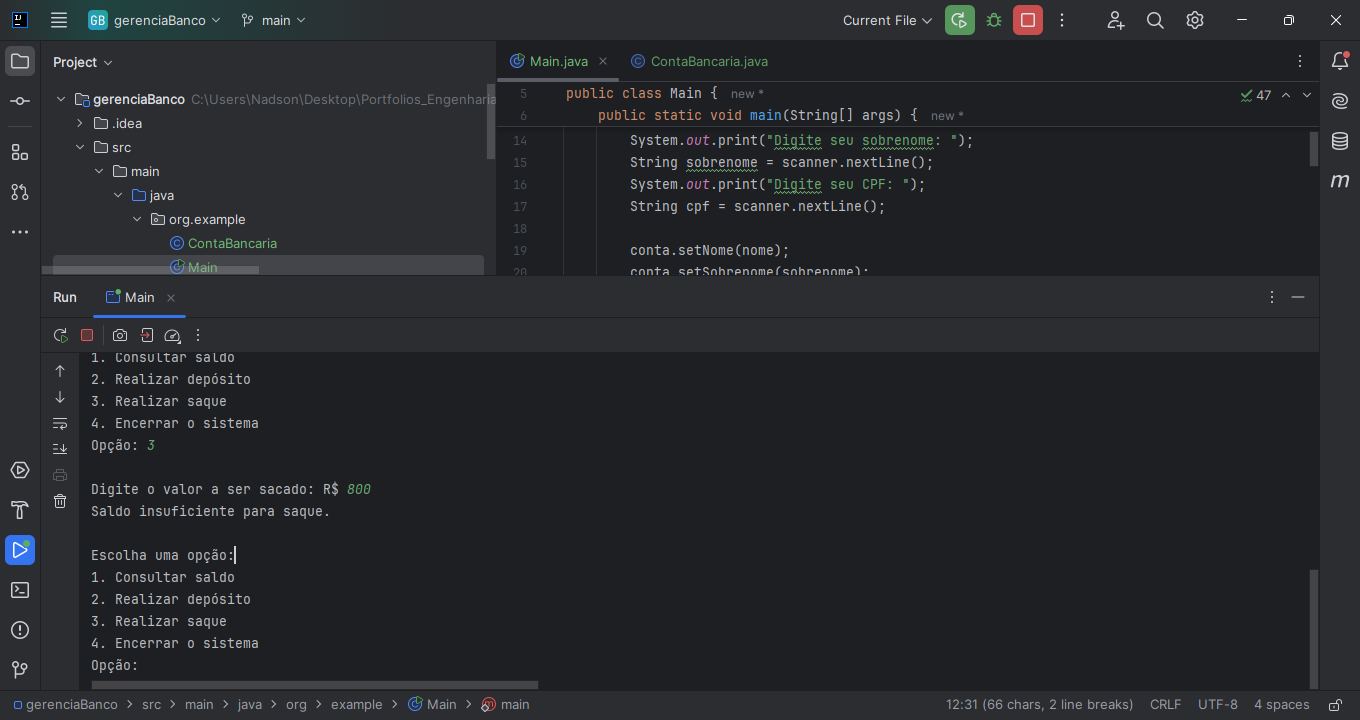
# EXPLICAÇÃO DAS IMAGENS

**Imagem 1:** Mostra o início da aplicação, solicitando o nome, sobrenome e CPF do usuário.

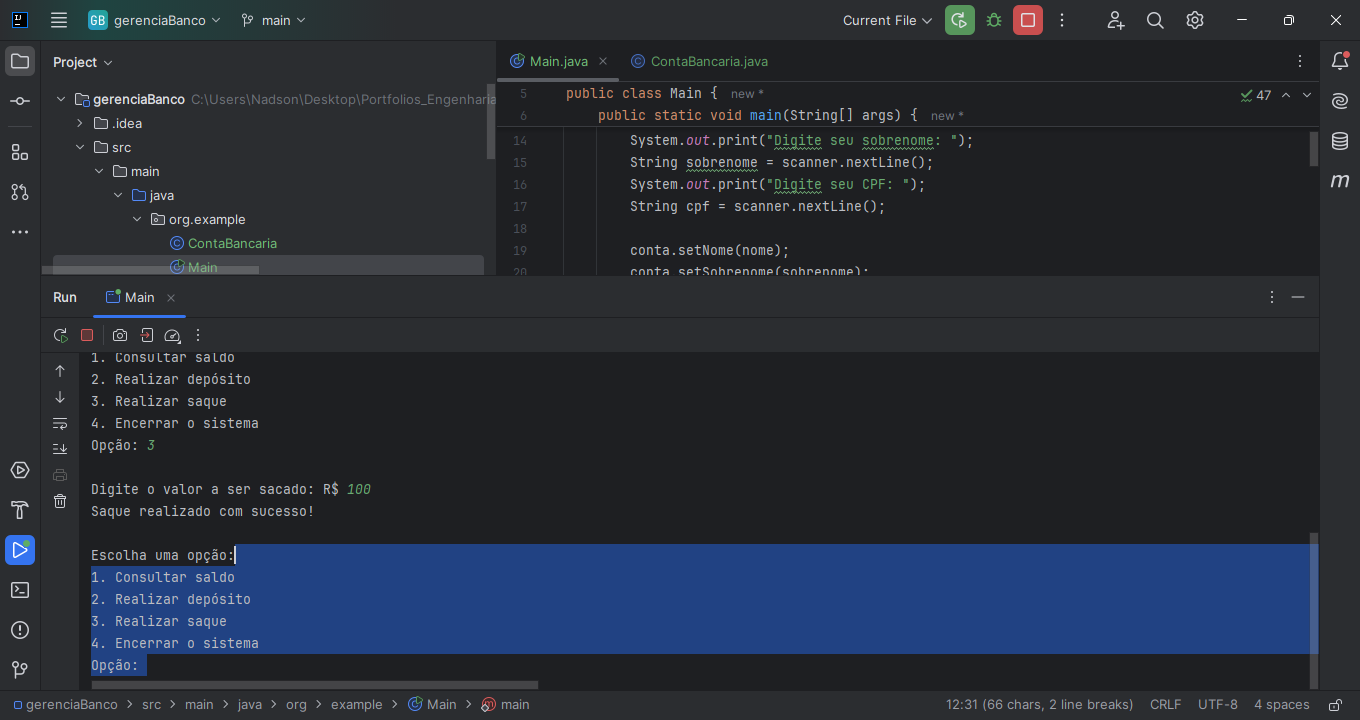


**Imagem 2:** Exibe o menu principal com as opções disponíveis.  


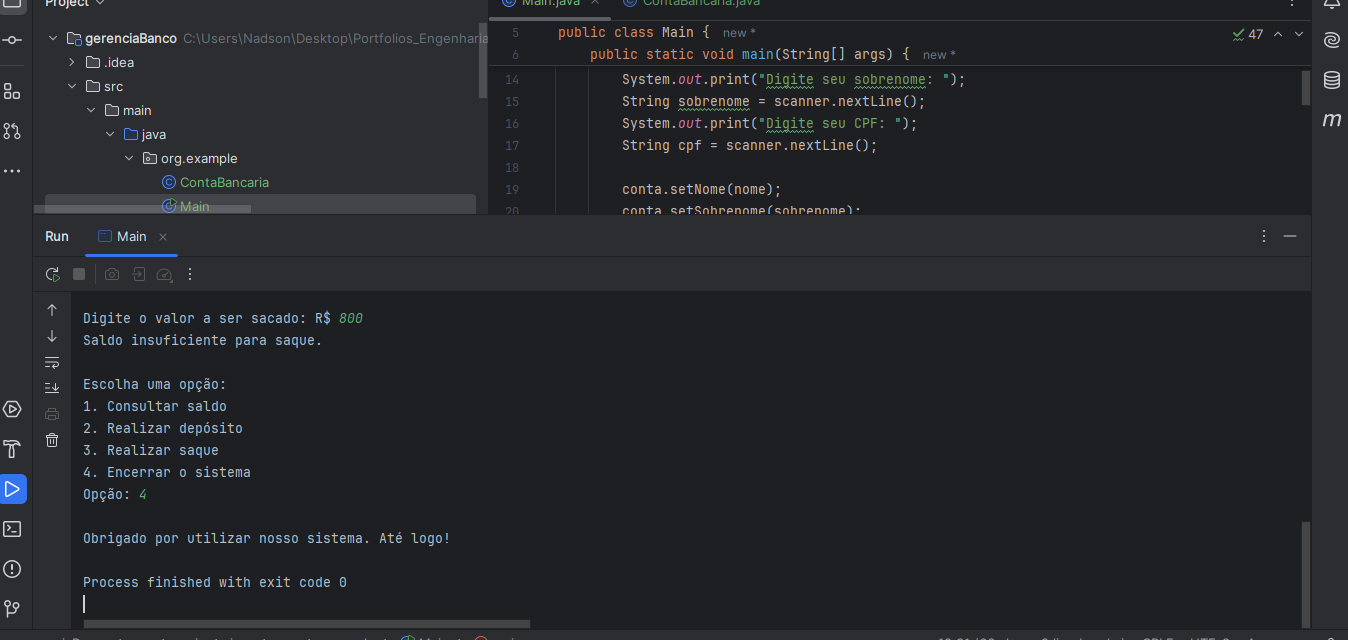
**Imagem 3:** Demonstra a realização de um depósito com sucesso.  
  


**Imagem 4:** Mostra a tentativa de saque com saldo insuficiente.  


**Imagem 5:** Mostra a tentativa de saque com saldo suficiente



**Imagem 6:** Apresenta a finalização do programa com mensagem de agradecimento.



# CÓDIGO FINAL

* **Main.java**

java

Copiar código

package org.example;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

ContaBancaria conta = new ContaBancaria();

System.out.println("Bem-vindo ao Sistema Bancário!");

System.out.print("Digite seu nome: ");

String nome = scanner.nextLine();

System.out.print("Digite seu sobrenome: ");

String sobrenome = scanner.nextLine();

System.out.print("Digite seu CPF: ");

String cpf = scanner.nextLine();

conta.setNome(nome);

conta.setSobrenome(sobrenome);

conta.setCpf(cpf);

int opcao;

do {

exibirMenu();

opcao = scanner.nextInt();

switch (opcao) {

case 1:

System.out.println("\nSaldo atual: R$ " + conta.getSaldo());

break;

case 2:

System.out.print("\nDigite o valor a ser depositado: R$ ");

double valorDeposito = scanner.nextDouble();

conta.depositar(valorDeposito);

System.out.println("Depósito realizado com sucesso!");

break;

case 3:

System.out.print("\nDigite o valor a ser sacado: R$ ");

double valorSaque = scanner.nextDouble();

if (conta.sacar(valorSaque)) {

System.out.println("Saque realizado com sucesso!");

} else {

System.out.println("Saldo insuficiente para saque.");

}

break;

case 4:

System.out.println("\nObrigado por utilizar nosso sistema. Até logo!");

break;

default:

System.out.println("\nOpção inválida. Tente novamente.");

}

} while (opcao != 4);

scanner.close();

}

private static void exibirMenu() {

System.out.println("\nEscolha uma opção:");

System.out.println("1. Consultar saldo");

System.out.println("2. Realizar depósito");

System.out.println("3. Realizar saque");

System.out.println("4. Encerrar o sistema");

System.out.print("Opção: ");

}

}

* **ContaBancaria.java**

java

Copiar código

package org.example;

public class ContaBancaria {

private String nome;

private String sobrenome;

private String cpf;

private double saldo;

public void setNome(String nome) {

this.nome = nome;

}

public void setSobrenome(String sobrenome) {

this.sobrenome = sobrenome;

}

public void setCpf(String cpf) {

this.cpf = cpf;

}

public double getSaldo() {

return saldo;

}

public void depositar(double valor) {

saldo += valor;

}

public boolean sacar(double valor) {

if (valor <= saldo) {

saldo -= valor;

return true;

} else {

return false;

}

}

}

# CONCLUSÃO

Essa atividade foi essencial para consolidar os conceitos de Programação Orientada a Objetos e a utilização do IntelliJ IDEA. A modularidade do código e o encapsulamento garantiram clareza e organização, facilitando futuras expansões do sistema. Essa prática reforça a importância de aplicar boas práticas no desenvolvimento de software.

# REFERÊNCIAS

* IntelliJ IDEA Documentation: <https://www.jetbrains.com/idea/documentation/>
* Java Documentation: <https://docs.oracle.com/javase/>