

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ALONSO AZEVEDO NETO

AULA PRÁTICA LINGUAGEM ORIENTADA A OBJETOS

DIASDÁVILA – BA

02/04/2024

ALONSO AZEVEDO NETO

AULA PRÁTICA LINGUAGEM ORIENTADA A OBJETOS

Aula prática de Linguagem Orientada a Objetos
apresentado como requisito parcial para a obtenção de
média semestral no curso Análise e Desenvolvimento de
Sistemas

Orientador(a): Leonardo Santiago
Tutor: Luana Gomes de Souza

DIASDÁVILA – BA
2024

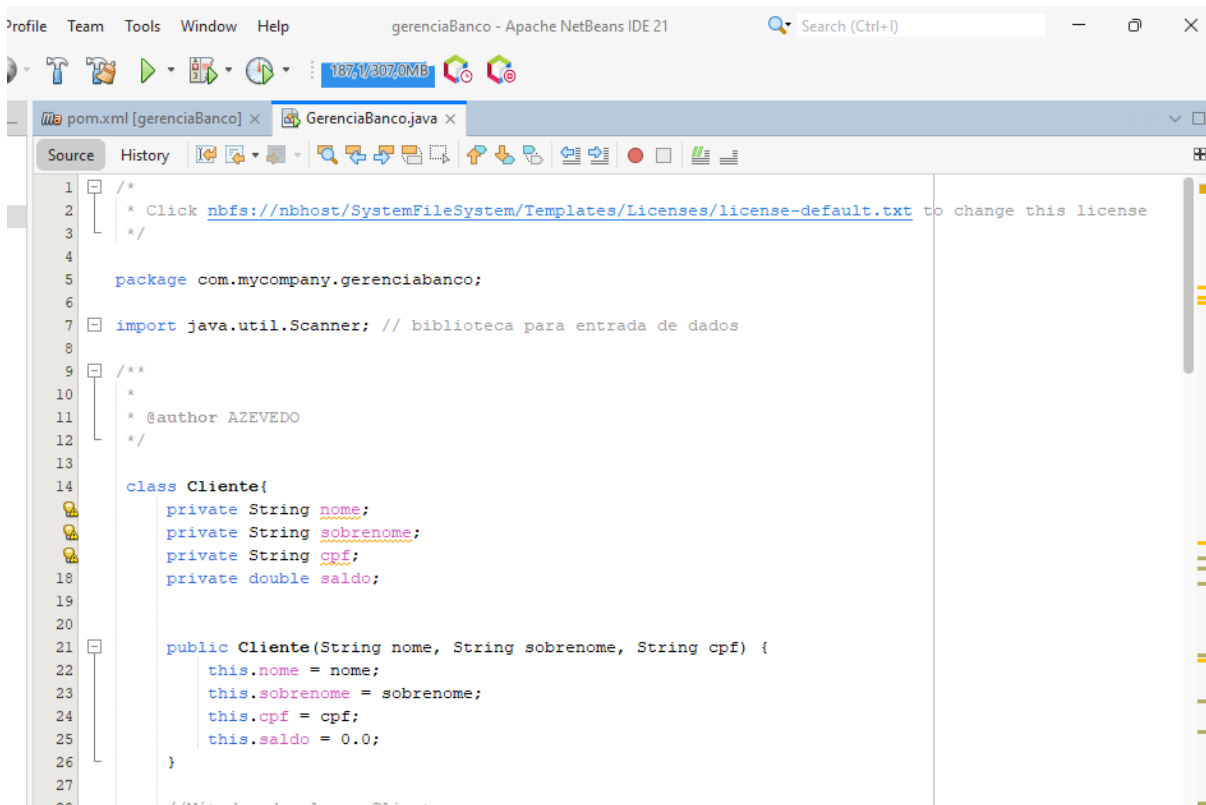
1 INTRODUÇÃO

Na aula prática de Linguagem Orientada a Objetos, exploraremos a implementação de um programa de gerenciamento de banco que permitirá aos usuários informarem seus dados pessoais, como nome, sobrenome e CPF, além de oferecer funcionalidades essenciais, tais como consultar saldo, realizar depósitos e saques. Estas operações estarão disponíveis de forma iterativa, permitindo que o usuário as execute conforme suas necessidades, até que ele opte por encerrar a utilização da aplicação. Através da utilização dos principais pilares da orientação a objetos - encapsulamento, herança, polimorfismo e abstração - este projeto tem como objetivo proporcionar uma experiência interativa e funcional ao usuário, permitindo-lhe interagir com um sistema que simula operações bancárias básicas. Dessa forma, através deste projeto, visamos não apenas a compreensão teórica dos conceitos de orientação a objetos, mas também a aplicação prática desses conceitos em um contexto realista e relevante, proporcionando aos usuários uma experiência interativa e funcional no gerenciamento de suas finanças

2 DESENVOLVIMENTO

Para o desenvolvimento do código do programa de gerenciamento de banco, primeiramente foi feita a instalação do JDK (Java Development Kit), um pacote de software usado especificamente para desenvolver aplicativos baseados em Java. Logo em seguida, partimos para a instalação da IDE (Integrated Development Environment) proposta no roteiro desta aula prática. Inicialmente, foram feitas algumas configurações nesta IDE, como a definição do tema. Além disso, verificamos se o Java SDK estava corretamente configurado, garantindo que todas as ferramentas necessárias estivessem prontas para iniciar o desenvolvimento. Com o ambiente de desenvolvimento devidamente configurado, prosseguimos para a criação do código-fonte do programa de gerenciamento de banco. Isso envolveu a criação de classes, métodos e lógica de negócios para garantir o funcionamento adequado do sistema. Durante o processo de desenvolvimento, foram realizados testes para verificar se o programa se comportava conforme o esperado. Correções e melhorias foram feitas conforme necessário para garantir a qualidade e a eficácia do software. Ao finalizar o desenvolvimento do código, o programa foi compilado e executado para verificar seu funcionamento em diferentes cenários. Em seguida, o código foi revisado para garantir a clareza, legibilidade e manutenibilidade do mesmo.

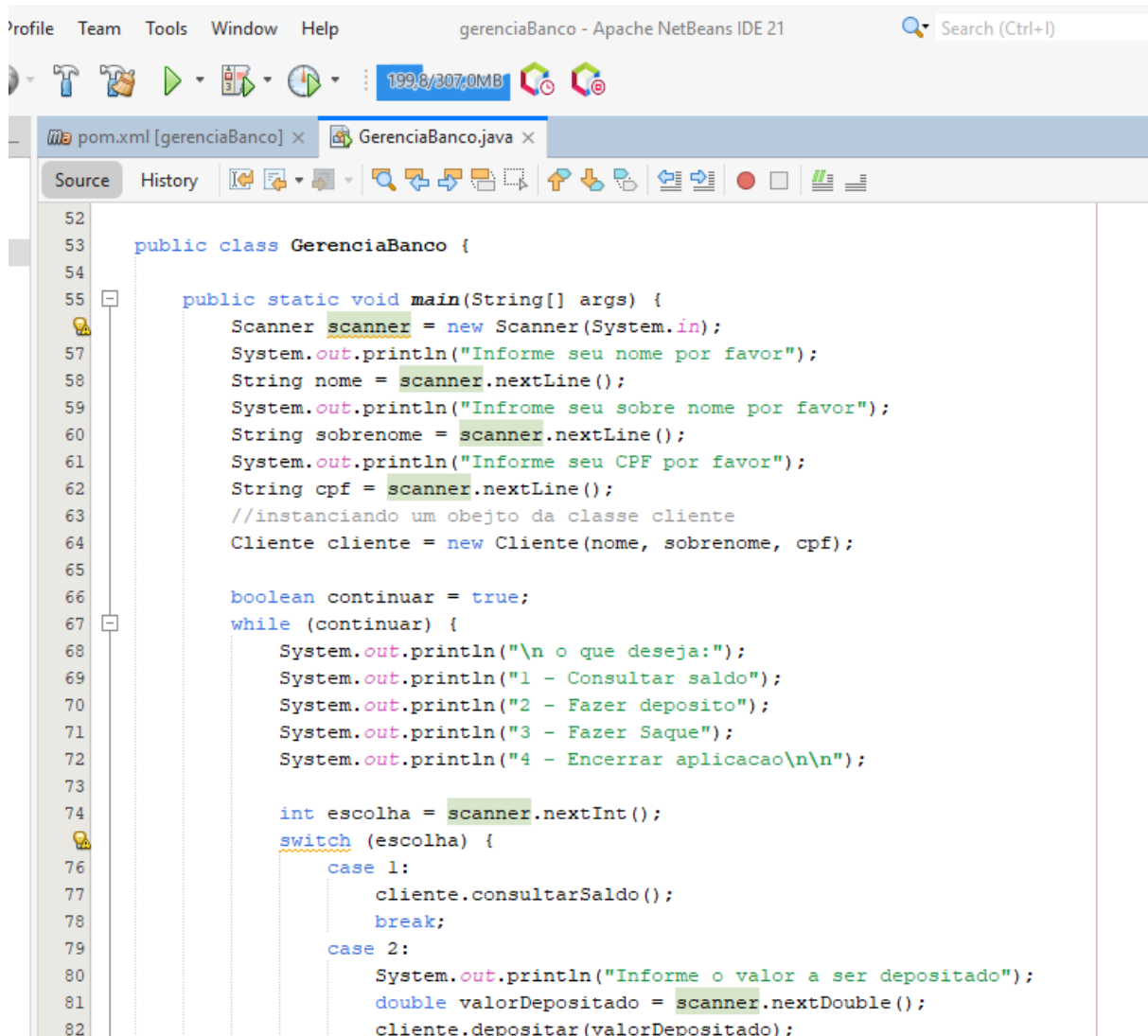
3 RESULTADOS



Conforme ilustrado na imagem acima, a parte inicial do código-fonte define a estrutura básica da classe Cliente. Esta classe contém métodos para inicializar os atributos do cliente, tais como nome, sobrenome, CPF e saldo. Além disso, possui um construtor que recebe essas informações como parâmetros durante a criação de um novo cliente. Adicionalmente, há a declaração das variáveis de instância privadas, tais como "nome", "sobrenome", "CPF" e "saldo", que armazenarão o primeiro nome, sobrenome, CPF e saldo do cliente.

O construtor da classe Cliente recebe três parâmetros: nome, sobrenome e cpf, e é usado para criar uma nova instância de Cliente com esses valores.

this.nome = nome atribui o valor do parâmetro "nome" à variável de instância "nome" da classe Cliente. this.sobrenome = sobrenome atribui o valor do parâmetro "sobrenome" à variável de instância "sobrenome" da classe Cliente. this.cpf = cpf atribui o valor do parâmetro "cpf" à variável de instância "cpf" da classe Cliente. this.saldo = 0.0 Inicializa a variável de instância saldo com o valor 0.0, ou seja, inicializa o saldo do cliente como zero quando uma nova instância de Cliente é criada.



```
52
53 public class GerenciaBanco {
54
55     public static void main(String[] args) {
56         Scanner scanner = new Scanner(System.in);
57         System.out.println("Informe seu nome por favor");
58         String nome = scanner.nextLine();
59         System.out.println("Informe seu sobre nome por favor");
60         String sobrenome = scanner.nextLine();
61         System.out.println("Informe seu CPF por favor");
62         String cpf = scanner.nextLine();
63         //instanciando um objeto da classe cliente
64         Cliente cliente = new Cliente(nome, sobrenome, cpf);
65
66         boolean continuar = true;
67         while (continuar) {
68             System.out.println("\n o que deseja:");
69             System.out.println("1 - Consultar saldo");
70             System.out.println("2 - Fazer deposito");
71             System.out.println("3 - Fazer Saque");
72             System.out.println("4 - Encerrar aplicacao\n\n");
73
74             int escolha = scanner.nextInt();
75             switch (escolha) {
76                 case 1:
77                     cliente.consultarSaldo();
78                     break;
79                 case 2:
80                     System.out.println("Informe o valor a ser depositado");
81                     double valorDepositado = scanner.nextDouble();
82                     cliente.depositar(valorDepositado);
```

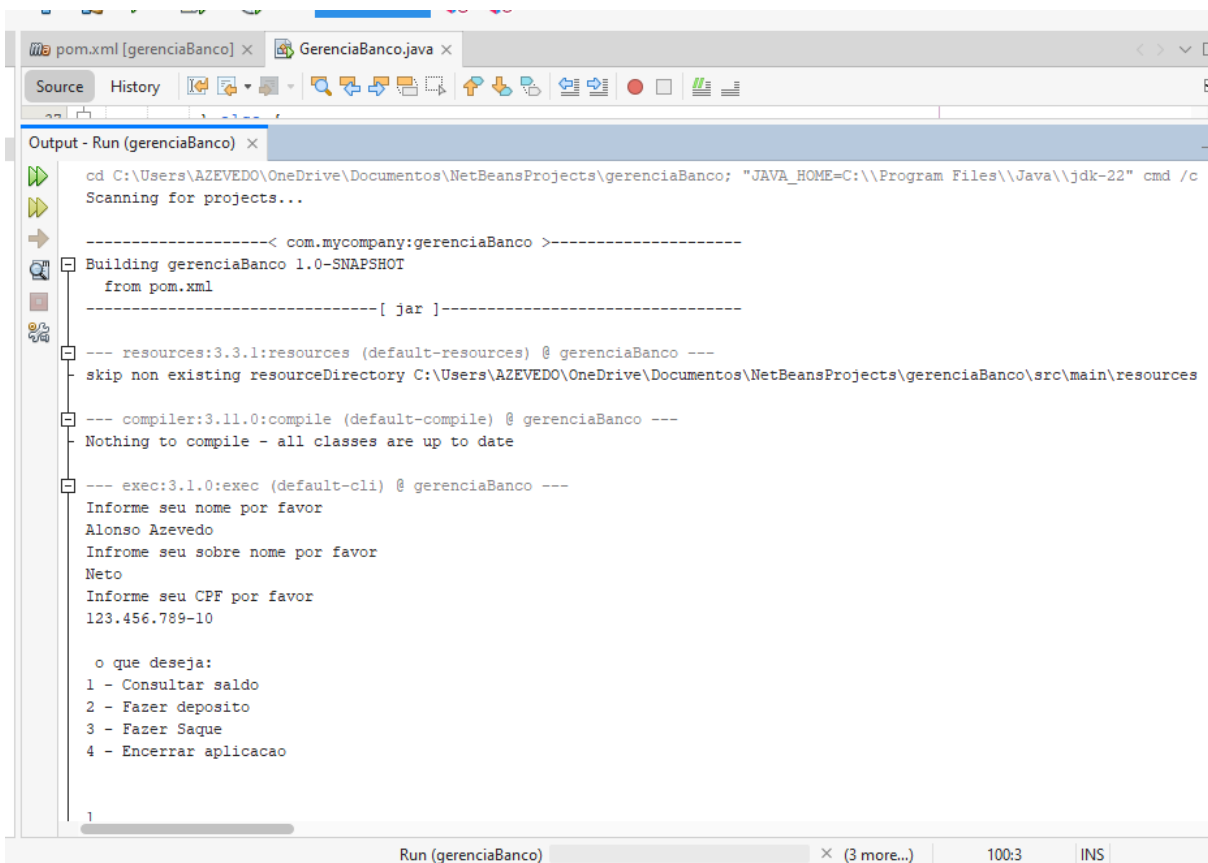
Essa parte do código estabelece a estrutura básica do programa, solicitando informações do cliente, criando um objeto Cliente com essas informações e fornecendo um menu de opções para interação do usuário com o sistema bancário. É feita a declaração da classe chamada "GerenciaBanco". O método "main(String[] args)" é o ponto de entrada do programa, onde a execução começa; é neste ponto que o código principal é executado. O try-with-resources é uma construção do Java para lidar com recursos que precisam ser fechados após o uso, como é o caso do Scanner. O Scanner é utilizado para ler entradas do usuário a partir do console "System.in". O programa solicita ao usuário que informe seu nome, sobrenome e CPF; cada informação é lida usando o método nextLine() do Scanner e armazenada nas variáveis nome, sobrenome e cpf, respectivamente. Um objeto da classe Cliente é criado usando o construtor que recebe o nome, sobrenome e CPF como parâmetros. Esses valores informados pelo usuário são passados para o construtor da classe

Cliente. Um loop while é utilizado para manter o programa em execução enquanto a variável continuar for verdadeira. A variável continuar é inicializada como true, indicando que o programa deve continuar em execução. No loop, o programa exibe um menu de opções para o usuário escolher o que deseja fazer.

O usuário é apresentado com um menu contendo as opções:

- Consultar saldo;
- Fazer depósitos;
- Fazer saque;
- Encerrar aplicação;

Abaixo estará o código em funcionamento



```
cd C:\Users\AZEVEDO\OneDrive\Documentos\NetBeansProjects\gerenciaBanco; "JAVA_HOME=C:\\Program Files\\Java\\jdk-22" cmd /c
Scanning for projects...

-----< com.mycompany:gerenciaBanco >-----
Building gerenciaBanco 1.0-SNAPSHOT
from pom.xml
-----[ jar ]-----

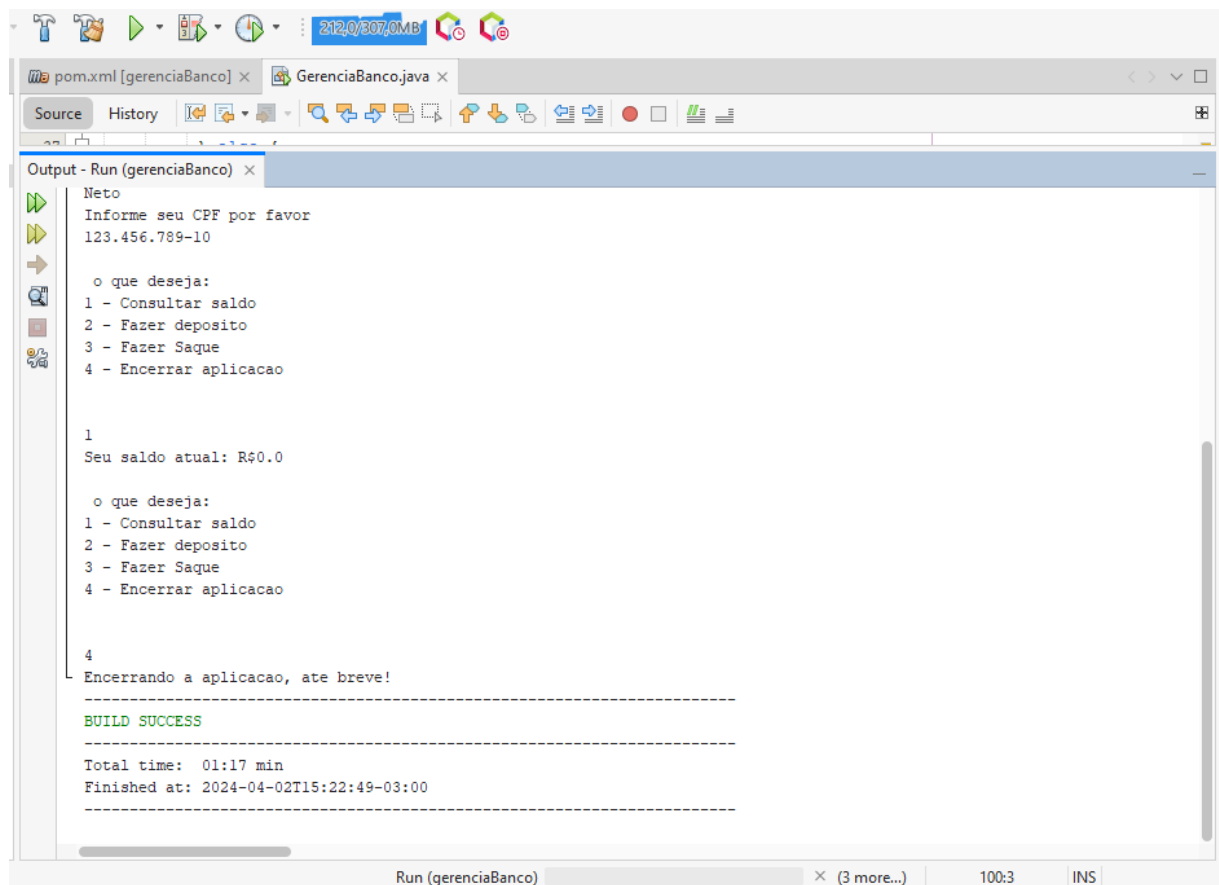
--- resources:3.3.1:resources (default-resources) @ gerenciaBanco ---
skip non existing resourceDirectory C:\Users\AZEVEDO\OneDrive\Documentos\NetBeansProjects\gerenciaBanco\src\main\resources

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

--- exec:3.1.0:exec (default-cli) @ gerenciaBanco ---
Informe seu nome por favor
Alonso Azevedo
Informe seu sobre nome por favor
Neto
Informe seu CPF por favor
123.456.789-10

o que deseja:
1 - Consultar saldo
2 - Fazer deposito
3 - Fazer Saque
4 - Encerrar aplicacao

1
```



4 CONCLUSÃO

Foi apresentado o ambiente de desenvolvimento integrado NetBeans IDE, utilizado para facilitar a criação, edição e depuração de códigos Java, bem como a sua integração com o gerenciador de dependências Maven. Por meio de exemplos práticos, foram explorados os conceitos de herança, polimorfismo, encapsulamento e abstração, demonstrando como essas características são aplicadas em projetos Java e como podem contribuir para o desenvolvimento de aplicações robustas, flexíveis e escaláveis. Ao final, uma aplicação de gerenciamento bancário foi criada, utilizando os conceitos e técnicas abordados ao longo do portfólio, como a criação de classes, métodos e estruturas de decisão. Essa aplicação permitiu praticar os conhecimentos adquiridos e consolidar o aprendizado de forma mais concreta. Conclui-se que o Java é uma linguagem de programação bastante popular e utilizada em diversos setores da indústria de software, desde a criação de aplicações desktop e jogos, até o desenvolvimento de sistemas web e mobile. Portanto, compreender seus conceitos fundamentais e técnicas avançadas é essencial.