**✅ VISÃO GERAL**

O código define um **sistema simples de gerenciamento bancário em Java**, com operações como:

* **Cadastro de contas**
* **Depósito**
* **Saque**
* **Consulta de saldo**
* **Listagem de saldos de todas as contas**

**📦 PACOTES E IMPORTAÇÕES**

java

CopiarEditar

package view;

import model.Conta;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

* A classe Gerenciar pertence ao **pacote view**, o que sugere que é parte da **camada de interface com o usuário (console)**.
* A classe Conta está no pacote model, o que sugere que representa o **modelo de dados** (conta bancária).
* Usa Scanner para entrada de dados e ArrayList para armazenar as contas.

**🧠 CLASSE Gerenciar**

java

CopiarEditar

private List<Conta> contas = new ArrayList<>();

private Scanner input = new Scanner(System.in);

* contas: Lista de objetos Conta criadas pelo usuário.
* input: Scanner para ler entradas do teclado.

**📌 MÉTODO main**

java

CopiarEditar

public static void main(String[] args) {

Gerenciar gerenciar = new Gerenciar();

int opcao = 0;

do {

// Apresenta o menu e lê a opção

switch (opcao) {

case 1 -> gerenciar.execCadastrar();

case 2 -> gerenciar.execSacar();

case 3 -> gerenciar.execDepositar();

case 4 -> gerenciar.execConsultar();

case 5 -> gerenciar.execListar();

case 9 -> System.out.println("Fim do programa");

default -> System.out.println("Opção inválida");

}

} while(opcao != 9);

}

Um **menu interativo de console** que chama diferentes métodos conforme a opção do usuário.

**🔧 FUNCIONALIDADES**

**1. Cadastrar Conta**

java

CopiarEditar

public void execCadastrar() {

Conta conta = new Conta();

conta.setNumero(...);

conta.setBanco(...);

contas.add(conta);

}

* Cria uma nova instância de Conta.
* Solicita número da conta e nome do banco.
* Adiciona à lista contas.

**2. Consultar Saldo de uma Conta**

java

CopiarEditar

public void execConsultar() {

int numero = ...;

Conta conta = buscarContaNaLista(numero);

if (conta == null) ...

else {

System.out.println("Saldo: " + conta.exibirSaldo());

}

}

* Pede o número da conta.
* Busca na lista.
* Mostra o saldo se encontrada.

**3. Listar Todas as Contas**

java

CopiarEditar

public void execListar() {

double geral = 0;

for (Conta conta : contas) {

geral += conta.exibirSaldo();

...

}

}

* Mostra todas as contas.
* Soma o saldo geral de todas.

**4. Depositar**

java

CopiarEditar

public void execDepositar() {

Conta conta = buscarContaNaLista(...);

conta.depositar(valor);

}

* Encontra a conta e realiza um depósito.

**5. Sacar**

java

CopiarEditar

public void execSacar() {

Conta conta = buscarContaNaLista(...);

conta.sacar(valor);

}

* Encontra a conta e realiza um saque.

**🔍 MÉTODO DE APOIO**

java

CopiarEditar

private Conta buscarContaNaLista(int numero) {

for (Conta conta : contas) {

if (conta.getNumero() == numero) return conta;

}

return null;

}

Realiza uma **busca linear** na lista de contas.

**🏗️ CLASSE FALTANTE: Conta**

Com base nas chamadas feitas no código, podemos **reconstruir a estrutura da classe Conta** como segue:

java

CopiarEditar

package model;

public class Conta {

private int numero;

private String banco;

private double saldo = 0.0;

public int getNumero() {

return numero;

}

public void setNumero(int numero) {

this.numero = numero;

}

public String getBanco() {

return banco;

}

public void setBanco(String banco) {

this.banco = banco;

}

public double exibirSaldo() {

return saldo;

}

public void depositar(double valor) {

if (valor > 0) {

saldo += valor;

} else {

System.out.println("Valor inválido para depósito");

}

}

public void sacar(double valor) {

if (valor > 0 && valor <= saldo) {

saldo -= valor;

System.out.println("Saque efetuado com sucesso");

} else {

System.out.println("Saldo insuficiente ou valor inválido");

}

}

}

**Observações:**

* saldo começa com 0.0.
* Métodos de **acesso (get/set)** para numero e banco.
* Métodos de **negócio**:
  + depositar(double valor)
  + sacar(double valor)
  + exibirSaldo() para retornar o saldo atual.

**🧾 CONCLUSÃO**

Esse código apresenta um **pequeno sistema de controle de contas bancárias**, utilizando conceitos de:

* Encapsulamento (atributos privados)
* Programação orientada a objetos (POO)
* Reutilização de código
* Estrutura de dados (List)
* Entrada via terminal (Scanner)
* Lógica de controle (switch, if, do-while)

**📚 Fontes de Confirmação**

* [Documentação oficial do Java - Scanner](https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html)
* [Oracle - Collections Framework](https://docs.oracle.com/javase/8/docs/technotes/guides/collections/index.html)
* [Oracle - Programação Orientada a Objetos em Java](https://docs.oracle.com/javase/tutorial/java/concepts/)