## 🧠 **1. Visão Geral do Projeto**

**Nome:** Sistema Bancário Simples  
**Objetivo:** Simular operações bancárias básicas usando POO  
**Foco:** Herança, encapsulamento, polimorfismo, abstração e estrutura de dados

## 📦 **2. Entidades Principais**

### 🧍‍♂️ Cliente

**Atributos:**

* Nome
* CPF
* E-mail
* Telefone
* (Opcional) Endereço

### 🏦 Conta (classe abstrata)

**Atributos comuns:**

* Número da conta (numConta)
* Cliente associado (cliente)
* Saldo (saldo)
* Senha
* Agência (opcional)

**Métodos esperados:**

* depositar
* sacar
* transferir
* consultarSaldo
* exibirDados
* validarSenha

### 💳 ContaCorrente (herda de Conta)

**Atributos específicos:**

* Taxa de manutenção (taxaManutencao)
* Limite do cheque especial (limiteChequeEspecial)
* Juros sobre cheque especial (jurosChequeEspecial)
* Taxa de saque (taxaSaque)
* Permite cheque especial? (permiteChequeEspecial)
* Está usando cheque especial? (usandoChequeEspecial)
* Última data de aplicação dos encargos

**Regras:**

* Pode sacar além do saldo, até o limite do cheque especial
* Aplica juros mensais se estiver no negativo
* Taxa de manutenção mensal

### 💰 ContaPoupanca (herda de Conta)

**Atributos específicos:**

* Taxa de rendimento mensal (taxaRendimento)
* Data de aniversário da conta (dataAniversario)
* Aplicação automática de rendimento? (rendimentoAutomatico)
* Dias de carência para saque após depósito (carenciaDias)

**Regras:**

* Aplica rendimento mensal no aniversário da conta
* Bloqueia saque se não cumprir a carência

### 🏛 Banco

**Responsabilidades:**

* Armazena lista de contas (HashMap<Integer, Conta>)
* Cria novas contas
* Busca conta por número
* Lista todas as contas
* Valida login e senha
* Intermedia transferências

## 🔁 **3. Fluxo de Funcionalidades**

### ✅ Criar conta

* Recebe dados do cliente
* Escolhe tipo de conta
* Gera número aleatório (único)
* Salva no sistema

### 🔍 Consultar conta

* Mostra dados da conta e saldo
* Pode pedir autenticação

### ➕ Depositar

* Adiciona saldo
* Atualiza data para cálculo de rendimento/carência

### ➖ Sacar

* Verifica saldo e permissões
* Conta Corrente: permite cheque especial
* Conta Poupança: respeita carência

### 🔄 Transferir

* Verifica saldo da origem
* Debita da origem
* Credita no destino

### 📃 Listar contas

* Mostra todos os dados de todas as contas

## 💡 **4. Conceitos de POO Aplicados**

| Conceito | Aplicação |
| --- | --- |
| **Encapsulamento** | private nos atributos + getters/setters |
| **Herança** | ContaCorrente e ContaPoupanca herdam Conta |
| **Polimorfismo** | Métodos como sacar() e transferir() usando Conta |
| **Abstração** | Conta define o modelo base, mas não é instanciável |
| **Coleções** | Map<Integer, Conta> armazena todas as contas criadas |

## 🚀 **5. Evoluções Futuras (Roadmap de Ideias)**

* 🔒 Login com autenticação
* 💾 Integração com banco de dados (MySQL via JDBC ou JPA)
* 🧾 Histórico de transações por conta
* 📱 Interface gráfica (Swing/JavaFX)
* 🌐 API REST com Spring Boot
* 🧠 Testes automatizados com JUnit
* 📤 Exportar dados para CSV/JSON
* 🧩 Criar perfis de administrador (para ver todas as contas)