LABORATÓRIO: HERANÇA E POLIMORFISMO

Última atualização: 12/05/2023

Exercício 1 – Criando subclasses de banco

Objetivo do exercício – Neste exercício, criaremos duas subclasses da **classe Conta** no projeto Banco: **ContaPoupanca** e **ContaCorrente**.

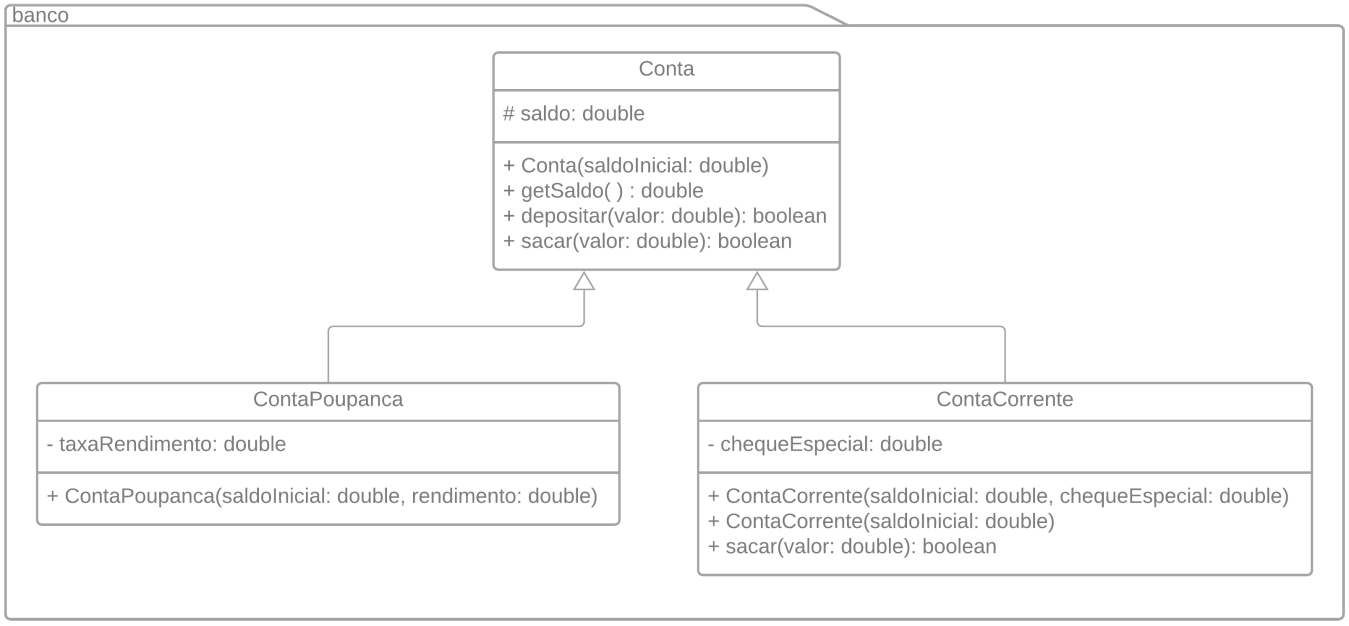


Figura 1: As subclasses da classe `Conta`.

Dica: Duplique a última versão do projeto desenvolvido na lista de exercício anterior para que você possa manter o histórico de evolução do código do projeto Banco durante as listas de exercícios. Outra possibilidade é usar um sistema de controle de versões.

Tarefas:

1. Modifique a **classe `Conta`**

   - Observe que no diagrama UML, a **classe Conta** mudou com relação à ultima versão: o **atributo saldo** é agora **protected** (indicado pelo **caracter #** ao invés do **caracter -**).
   - No programa em Java, mude o modificador de acesso do **atributo `saldo`** para **`protected`**.

2. Adicione ao pacote banco a subclasse **`ContaPoupanca`**, conforme diagrama de classes acima.
   - Observe que no construtor, além do saldo inicial existe um valor que indica a taxa de rendimento mensal da poupança (por exemplo 0.03 para indicar 3%)

3. Adicione ao pacote banco a subclasse **`ContaCorrente`**, conforme diagrama de classes acima.

   - Observe que em um dos construtores, além do saldo inicial existe um valor que indica o cheque especial.

- Observe que a **classe `ContaCorrente`** sobrescreve (**Overriding**) o método **`sacar`**. A implementação do método deve checar o seguinte: se o valor dos atributos **`saldo`** e **`chequeEspecial`** forem suficientes para cobrir o valor a ser sacado; proceda normalmente (caso seja necessário o valor do saldo deve ser ficar negativo, desde que não estoure o limite de cheque especial). Se o valor do saque for maior que a soma do saldo atual mais cheque especial, então toda a transação deve falhar com a saldo atual não sendo afetado.

4. Evolua o programa `TesteBanco` da lista de exercícios anterior para produzir a seguinte saída (observe que a nova versão deverá criar objetos do tipo `ContaPoupanca` ou `ContaCorrente`. Não deve-se criar objetos do tipo `Conta`). Também observe que deverá ser criada uma conta conjunta entre Diego Alves e sua esposa Lorena Lara. Execute o programa `TesteBanco`. A saída gerada deverá ser similar à abaixo:

```
----------------- CRIAÇÃO DE CONTAS BANCÁRIAS ------------------

Criando uma conta poupança para o cliente Bruno Henrique com saldo de R$ 50.000,00 e taxa de rendimentos de 3%.
Criando uma conta corrente para o cliente Everton Ribeiro com saldo de R$ 45.000,00 e cheque especial de R$ 30.000,00
Criando uma conta corrente para o cliente Filipe Luis com saldo de R$ 70.000,00 sem cheque especial.
Criando uma conta poupança para o cliente Gabriel Barbosa com saldo de R$ 220.000,00 e taxa de rendimentos de 3%."
Criando uma conta corrente para o cliente Daniel Alves com saldo de R$ 50.000,00 e sem cheque especial.
Diego Alves autorizou o cadastro de sua conta corrente como sendo conjunta com a cliente Lorena Lara.

----------------- RELATÓRIO DE TRANSAÇÕES ------------------
Recuperando o cliente Bruno Henrique
Sacando R$ 1.200,00: true
Depositando R$ 8.525,00: true
Sacando R$ 12.800,00: true
Sacando R$ 50.000,00: false
Cliente [Bruno, Henrique] tem o saldo de R$ 44525.0

Recuperando o cliente Everton Ribeiro
Saque de R$ 12.500,00: true
Saque de R$ 18.500,00: true
deposito de R$ 3.500,00: true
Saque de R$ 17.000,00: true
Saque de R$ 25.000,00: true
Cliente [Everton, Ribeiro] tem o saldo de R$ -24500.0

Recuperando o cliente Filipe Luis.
Saque de R$ 25500.00: true
deposito de R$ 2000.00: true
Saque de R$ 37200.00: true
Saque de R$ 15000.00: false
Cliente [Filipe, Luis] tem o saldo de R$ 9300.0
```

```
Recuperando o cliente Gabriel Barbosa
Saque de R$ 15500.00: true
deposito de R$ 3000.00: true
Saque de R$ 23400.00: true
Saque de R$ 17000.00: true
Cliente [Gabriel, Barbosa] tem o saldo de R$ 167100.0

Recuperando o cliente Diego Alves
Saque de R$ 28000.00: true
deposito de R$ 3000.00: true
Saque de R$ 17000.00: true
Cliente [Diego, Alves] tem o saldo de R$ 8000.0

Recuperando a cliente Lorena Lara
Saque de R$ 32000.00: false
deposito de R$ 13000.00: true
Saque de R$ 16.600.00: true
Cliente [Lorena, Lara] tem o saldo de R$ 4400.0
```

Figura 6.2: Saída do Exercício 1

As transações na conta poupança de Bruno Henrique ocorrem sem nenhum problema. As transações do cliente Everton Ribeiro em sua conta corrente com cheque especial também procedem, entretanto verifique que no caso dele foi necessário uso do cheque especial. Tendo em vista que Filipe Luis possui uma conta corrente sem cheque especial, nem todas as transações em sua conta são executadas. Todas as transações na conta poupança do cliente Gabriel Barbosa são executadas com sucesso. Diego alves tem uma conta corrente, sem cheque especial, conjunta com Lorena Lara. Embora todas as transações de Diego Alves sejam executadas com sucesso, uma transação de sua esposa Lorena Lara não é executada.

Errata:

- Na sexta linha da saída no console (Figura 6.2), onde está descrito Daniel Alves é na verdade Diego Alves.

Exercício 2– Criando Contas de Clientes

Objetivo do exercício – Neste exercício, criaremos uma associação para representar a agregação de **clientes** e **contas**. Ou seja, um dado **cliente** pode possuir várias **contas** de diferentes tipos. A Figura 3 exibe o diagrama de classes para esta versão do projeto `Banco`.

Dica: Duplique o projeto desenvolvido no exercício anterior para que você possa manter o histórico de evolução do código do projeto Banco durante as listas de exercícios. Outra possibilidade é usar um sistema de controle de versões.

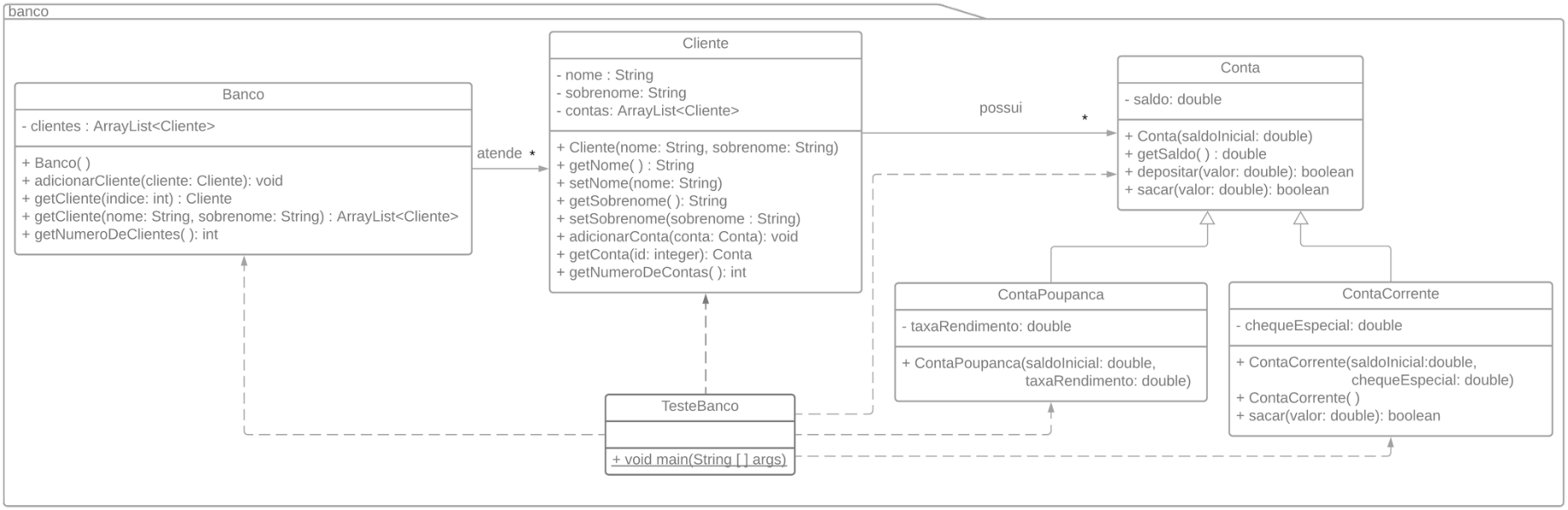


Figura 3: Evolução do projeto bancário para permitir associar um cliente a mais de uma conta independente do tipo.

Errata:

- No diagrama de classes da Figura 3, na classe `Cliente` em vez de termos:
  - `-contas: ArrayList<Cliente>`, devemos ler – `contas: ArrayList<Conta>`

Tarefas:

Modifique a **classe `Cliente`**

1. Modifique a **classe `Cliente`** para permitir a associação de um cliente com várias contas. Ela deve incluir três métodos públicos: **`adicionarConta(Conta)`**, **`getConta(int)`** e **`getNumeroDeContas()`**. Reutilize a classe `ArrayList` pertencente à API padrão da linguagem Java.

Copie o arquivo **`EsqueletoTesteBanco2.java`** fornecido junto com o enunciado deste exercício para o pacote `banco`. Este arquivo cria um conjunto de clientes e contas e gera um relatório de clientes e seus respectivos saldos em conta. Neste arquivo você encontrará blocos de comentários que iniciam e terminam com **/*** ... ***/**. Estes comentários indicam a localização onde você deve inserir código.

2. Use o **operador `instanceof`** para testar o tipo de conta e atribua ao atributo **`tipoConta`** um valor apropriado, tal como **"Conta Poupanca"** ou **"Conta Corrente"**
3. Imprima o **tipo de conta** e o **saldo**.
4. Compile e execute este programa. Você deverá ver a seguinte saída:

```
            RELATÓRIO DE CLIENTES
            ================

Cliente: Bruno Henrique
O saldo da Conta Poupança é de R$ 50000.0
O saldo da Conta Corrente é de R$ 220000.0

Cliente: Everton Ribeiro
O saldo da Conta Corrente é de R$ 45000.0

Cliente: Filipe Luis
O saldo da Conta Poupança é de R$ 150000.0
O saldo da Conta Corrente é de R$ 70000.0

Cliente: Gabriel Barbosa
O saldo da Conta Corrente é de R$ 70000.0
O saldo da Conta Poupança é de R$ 220000.0
```

Exercício 3– Pesquisando clientes pelo nome e sobrenome

Objetivo do exercício – Neste exercício, criaremos um método na classe `Banco` que torna mais fácil a perquisa por clientes (usando `nome` e `sobrenome`).

Dica: Duplique o projeto desenvolvido no exercício anterior para que você possa manter o histórico de evolução do código do projeto `Banco` durante as listas de exercícios. Outra possibilidade é usar um sistema de controle de versões.

Tarefas:

Implemente em Java o método `getCliente` da classe `Banco` presente no Diagrama de classes do exercício anterior. A assinatura do método em Java é fornecida abaixo:

```
public ArrayList<Cliente> getCliente(String nome, String sobrenome)
```

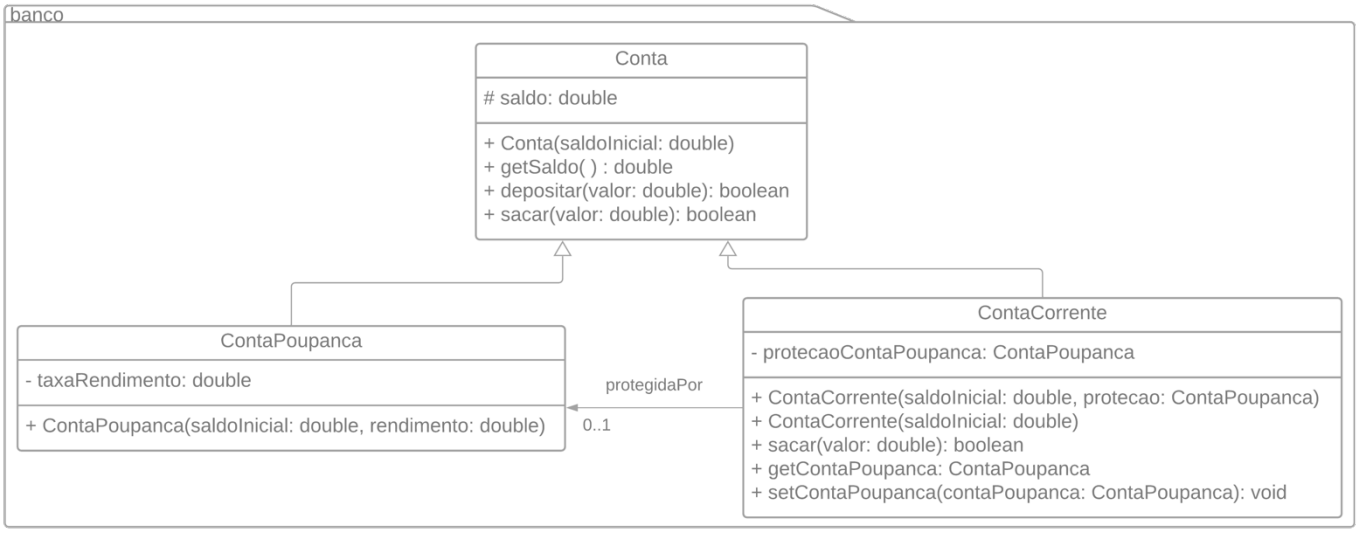
1. Observe que o método `getCliente` retorna uma lista de clientes (`ArrayList<Cliente>`), tendo em vista que mais de um cliente pode ter o mesmo nome e sobrenome.
2. Copie o arquivo **`EsqueletoTesteBanco3.java`** fornecido junto com o enunciado deste exercício para o pacote `banco`. Neste arquivo você encontrará blocos de comentários que iniciam e terminam com /*** **...** ***/. Estes comentários indicam a localização onde você deve inserir código.

```
                    RESULTADO DA BUSCA
================================================================
Nome fornecido para busca...: Bruno Henrique
Temos 2 cliente(s) com o nome fornecido

Cliente 1 ...: Bruno Henrique
Tipo de conta: Conta Poupanca com saldo de R$ 50000.0
Tipo de conta: Corrente com saldo de R$ 220000.0

Cliente 2 ...: Bruno Henrique
Tipo de conta: Conta Poupanca com saldo de R$ 2000.0

================================================================
Nome fornecido para busca...: Gabriel Barbosa
Temos 1 cliente(s) com o nome fornecido

Cliente 1 ...: Gabriel Barbosa
Tipo de conta: Conta Poupanca com saldo de R$ 220000.0

================================================================
Nome fornecido para busca...: Giorgian De Arrascaeta
Temos 0 cliente(s) com o nome fornecido

================================================================
```

Exercício 4– Criando Subclasses de Conta

Objetivo do exercício – Neste exercício, criaremos duas subclasses da **classe Conta** no projeto **Banco**: **`ContaPoupanca`** e **`ContaCorrente`**.

Nota – Esta é uma versão alternativa ao **exercício 1**. Ela incorpora um modelo mais complexo do mecanismo de proteção de saques. Ela utiliza a conta poupança do cliente para executar tal proteção.



Dica: Duplique o projeto desenvolvido no na última vesão da LISTA ANTERIOR para que você possa manter o histórico de evolução do código do projeto Banco durante as listas de exercícios. Outra possibilidade é usar um sistema de controle de versões. Este exercício é uma alternativa ao exercício 1 desta lista.

Tarefas:

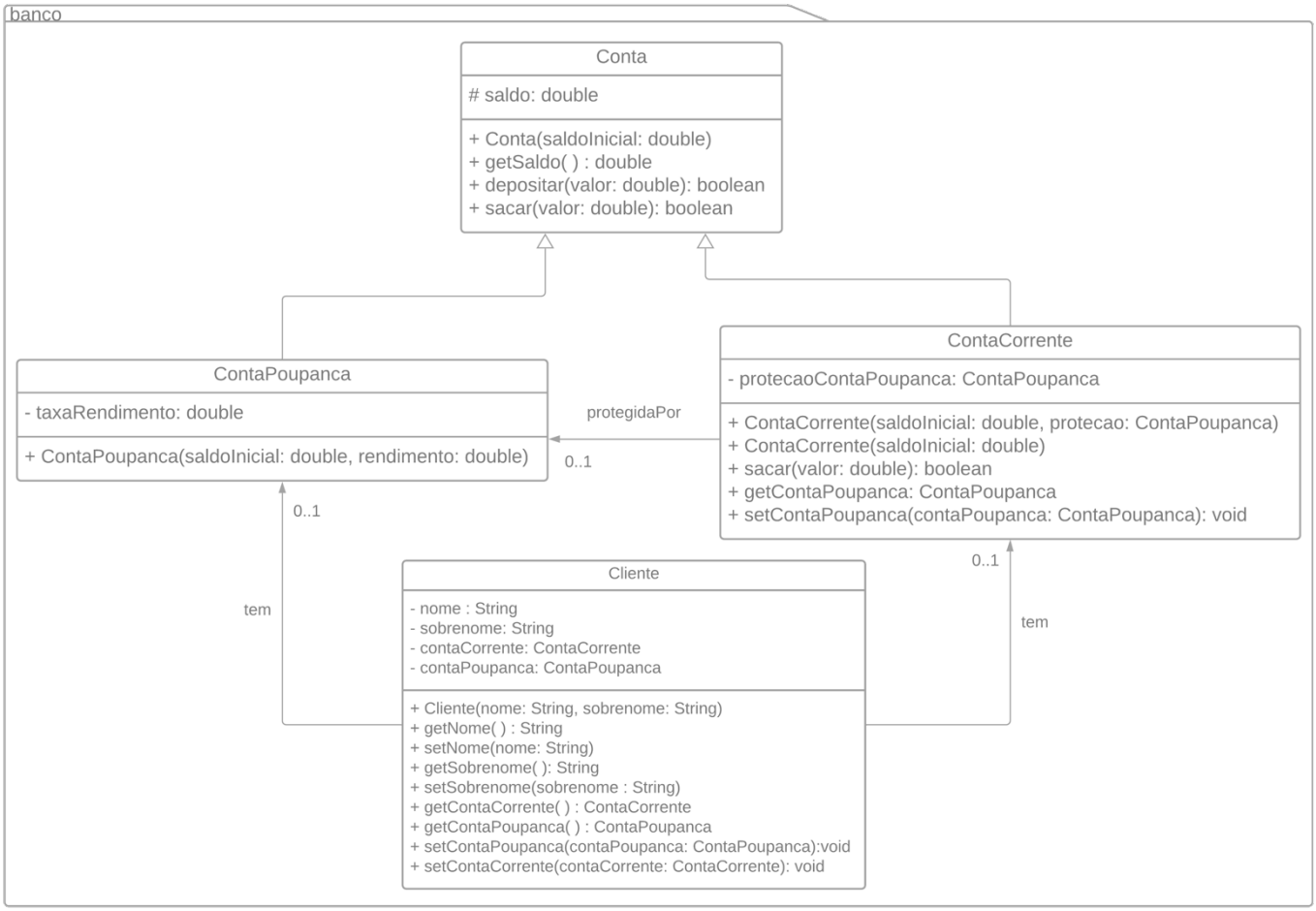
Adicione ao pacote banco as subclasses **`ContaPoupanca`** e **`ContaCorrente`**, conforme diagrama acima.

1. A classe `ContaCorrente` deve sobreescrever (**overriding**) o método **`sacar`**. Ela deve checar o seguinte: se o valor do atributo **`saldo`** é suficiente para cobrir o total a ser sacado; proceda normalmente. Se não e se existe uma proteção através da conta poupança, então tente cobrir a diferença (**`saldo - total`**) pelo saldo da conta poupança. Neste caso, proceda, fazendo com que o saldo em conta corente fique igual a 0.00 e o saldo em conta poupança atualizado. Se a última transação falhar (total a a ser sacado for maior que o saldo da conta corrente mais o saldo da poupança), então toda transação deve falhar com a saldo atual da conta corrente não sendo afetado.

Modifique a **classe `Cliente`** para guardar duas contas: uma **conta corrente** e uma **conta poupança**; ambas opcionais conforme diagrama UML abaixo.

2. Na lista de exercícios anterior, a **classe `Cliente`** tinha um atributo de associação chamado **`conta`** que mantinha um objeto **`Conta`**. Reescreva esta classe para conter dois atributos de associação: **`contaPoupanca`** e **`contaCorrente`** com valores **default `null`**.
3. Inclua dois métodos de acesso: **`getContaCorrente()`** e **`getContaPoupanca()`**, que retornam as **contas corrente** e **poupanca** respectivamente.

4. Inclua dois métodos: **`setContaPoupanca`** e **`setContaCorrente`** , que iniciam a conta poupança e corrente, respectivamente.



Teste o código:

5. Copie arquivo **`TesteBancoExercicio4.java`** fornecido junto com o enunciado deste exercício para o pacote `banco`. Não deixe de analisar o código para verificar seu correto entendimento.
6. Analise o código de **`TesteBancoExercicio4.java`** para verificar seu correto entendimento.
7. Compile e execute o programa **`TesteBancoExercicio4`**. A saída deve ser:

```
Criando uma conta conta poupança com saldo de R$ 50.000,00 para o cliente Bruno Henrique.

Criando uma conta corrente com saldo de R$ 220.000,00 para o cliente Bruno Henrique.

Criando uma conta corrente com saldo de R$ 50.000,00 para o cliente Diego Alves, sem conta poupança de proteção

Contas criadas com sucesso

------------------ RELATÓRIO DE TRANSAÇÕES -------------------
Recuperando o cliente Bruno Henrique
Sacando R$ 230.000,00: true
Depositando R$ 8.525,00: true
Sacando R$ 12.800,00: true
Sacando R$ 40.000,00: false
Cliente [Bruno, Henrique] tem uma conta corrente com saldo R$ 0.0
e uma proteção em conta poupança com saldo R$ 35725.0
Recuperando o cliente Diego Alves
Saque de R$ 12.500,00: true
Saque de R$ 18.500,00: true
deposito de R$ 3.500,00: true
Saque de R$ 17.000,00: true
Saque de R$ 25.000,00: false
Cliente [Diego, Alves] tem uma conta corrente com saldo R$ 5500.0
```