UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO
CIENTÍFICA DEPARTAMENTO DE MATEMÁTICA APLICADA

REBECA LIE YATSUZUKA SILVA

Protótipo de aplicação bancária em Java

Campinas
2021

REBECA LIE YATSUZUKA SILVA

Protótipo de aplicação bancária em Java

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado, sob a orientação do(a) Prof. Laércio Luís Vendite.

Campinas
2021

Resumo

Pesquisas mostram que Java é uma das linguagens mais utilizadas por desenvolvedores e programadores de todo o mundo. Por causa principalmente de sua flexibilidade, Java se tornou a linguagem base e o padrão global para o desenvolvimento de quase todos os jogos, aplicações, conteúdo Web, softwares corporativos, entre outros.

Nesse projeto iremos apresentar um *mvp* (produto mínimo viável) de aplicação bancária em Java. O programa tem duas opções de conta: conta de pessoa física e conta de pessoa jurídica. Esses dois objetos têm senha, nome e saldo como atributos em comum, a diferença é que a pessoa física tem um CPF enquanto a pessoa jurídica tem um CNPJ. Ademais, esta aplicação apresenta as seguintes operações para o usuário: criar uma nova conta, fazer login, sacar e depositar dinheiro, transferir dinheiro de uma conta para outra e, além disso, apresentar o extrato que informará os detalhes principais de suas transações.

Abstract

Research shows that Java is one of the most used languages by developers and programmers around the world. Mainly because of its flexibility, Java has become a base language and the global standard for the development of almost all games, applications, web content, enterprise software, among others.

In this project we will present a mvp (minimum viable product) of a bank application in Java. The program has two account options: individual account and corporate account. These two objects have a password, name and balance as attributes in common, the difference is that the individual has a CPF while the legal entity has a CNPJ. Furthermore, this application has the following operations for the user: create a new account, login, withdraw and deposit money, transfer money from one account to another and, in addition, present the statement that will inform the main details of your transactions.

Sumário

1 Introdução

Uma pesquisa da Tiobe Index, empresa especializada em qualidade de software que analisou o mercado de TI até julho de 2017, apontou que o Java é a linguagem mais utilizada por desenvolvedores e programadores de todo o mundo. Entre os diversos motivos responsáveis por esse resultado, a flexibilidade foi um dos mais apontados.

Este projeto apresentará um *mvp* (produto mínimo viável) de aplicação bancária em Java. O programa terá disponíveis duas opções de conta: conta de pessoa física e conta de pessoa jurídica. Esses dois objetos terão senha, nome e saldo como atributos em comum, a diferença será que a pessoa física terá um CPF enquanto a pessoa jurídica terá um CNPJ. Ademais, esta aplicação apresentará as seguintes operações para o usuário: criar uma nova conta, fazer login, sacar e depositar dinheiro, transferir dinheiro de uma conta para outra e, além disso, apresentar o extrato que informará os detalhes principais de suas transações.

2 Objetivo

O objetivo desse projeto é apresentar um código de modelo *mvp* bancário programado em Java mais focado na parte de *Back-end*. Para a parte de interação com o usuário, por meio de caixas de diálogo simples e objetivas, foi utilizado o pacote *javax.swing.JOptionPane*.

3 Desenvolvimento e resultados

3.1 Classes das contas

Para cada tipo de conta foi criada uma classe, em que os atributos são as informações relacionadas àquele modelo de conta. Como as contas de pessoa jurídica e física possuem atributos em comum, também foi construído uma classe mãe, que recebeu o nome de ***Modelo_Conta***, enquanto que as classes filhas receberam os nomes de ***PessoaFísica***, classe responsável por criar e guardar as informações das contas de pessoa física, e de ***PessoaJurídica***, classe responsável por criar e guardar as informações das contas de pessoa jurídica.

3.1.1 Classe *Modelo_Conta*

Na classe ***Modelo_Conta*** criamos e implementamos os métodos *Get* e *Set* para os atributos referentes às informações base de qualquer conta bancária nossa: senha, nome, saldo, número de transações (representada por *n_transações*) e lista de transações (representada por listaT, explicado no próximo subcapítulo). Além disso, fizemos o método construtor com os atributos senha e nome, já que, quando uma

conta é criada, seu saldo e número de transações será igual a zero e a lista de transações estará vazia.

```java
package banco;

import java.util.*;

public abstract class Modelo_Conta {
    private String senha;
    private String nome;
    private double saldo = 0;
    private ArrayList<Transações> listaT = new ArrayList<>();

    public String getSenha() {
        return senha;
    }

    public void setSenha(String senha) {
        this.senha = senha;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public double getSaldo() {
        return saldo;
    }

    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }

    public ArrayList<Transações> getListaT() {
        return listaT;
    }

    public void setListaT(Transações transação) {
```

```
        this.listaT.add(transação);
    }

    public Modelo_Conta(String senha, String nome) {
        this.senha = senha;
        this.nome = nome;
    }
}
```

3.1.2 Classe *PessoaFísica*

Na classe ***PessoaFísica*** criamos e implementamos os métodos *Get* e *Set* para o atributo referente à informação exclusiva da nossa conta para pessoa física: CPF. Além disso, estendemos a classe ***Modelo_Conta***, para que pudéssemos usar os atributos e métodos dessa classe e fizemos o método construtor com os atributos senha, nome e CPF.

```
package banco;

public class PessoaFísica extends Modelo_Conta{
    private String CPF;

    public String getCPF() {
        return CPF;
    }

    public void setCPF(String CPF) {
        this.CPF = CPF;
    }

    public PessoaFísica(String CPF, String senha, String nome) {
        super(senha, nome);
        this.CPF = CPF;
    }

}
```

3.1.3 Classe *PessoaJurídica*

Na classe ***PessoaJurídica*** criamos e implementamos os métodos *Get* e *Set* para o atributo referente à informação exclusiva da nossa conta para pessoa jurídica: CNPJ. Além disso, estendemos a classe ***Modelo_Conta***, para que pudéssemos usar os atributos e métodos dessa classe e fizemos o método construtor com os atributos senha, nome e CNPJ.

```
package banco;

public class PessoaJurídica extends Modelo_Conta{
    private String CNPJ;

    public String getCNPJ() {
        return CNPJ;
    }

    public void setCNPJ(String CNPJ) {
        this.CNPJ = CNPJ;
    }

    public PessoaJurídica(String CNPJ, String senha, String nome) {
        super(senha, nome);
        this.CNPJ = CNPJ;
    }

}
```

3.2 Classe *Transações*

Na classe ***Transações*** criamos e implementamos os métodos *Get* e *Set* para os atributos referentes às informações base de qualquer transferência nossa: operação, valor, remetente, destinatário e saldo final (representado por *saldo_final*). Além disso, fizemos o método construtor com todos os atributos listados anteriormente e reescrever nosso método toString, para que ele retorne algo significativo.

```
package banco;

public class Transações {
    String operação;
```

```
    double valor;
    String remetente;
    String destinatário;
    double saldo_final;

    public Transações(String operação, double valor, String
remetente, String destinatário, double saldo_final) {
        this.operação = operação;
        this.valor = valor;
        this.remetente = remetente;
        this.destinatário = destinatário;
        this.saldo_final = saldo_final;
    }

    public String getOperação() {
        return operação;
    }

    public void setOperação(String operação) {
        this.operação = operação;
    }

    public double getValor() {
        return valor;
    }

    public void setValor(double valor) {
        this.valor = valor;
    }

    public String getRemetente() {
        return remetente;
    }

    public void setRemetente(String remetente) {
        this.remetente = remetente;
    }

    public String getDestinatário() {
        return destinatário;
    }
```

```
    public void setDestinatário(String destinatário) {
        this.destinatário = destinatário;
    }

    public double getSaldo_final() {
        return saldo_final;
    }

    public void setSaldo_final(double saldo_final) {
        this.saldo_final = saldo_final;
    }

    @Override
    public String toString() {
        return this.operação + "| valor: " + this.valor + "|
remetente: " + this.remetente + "| destinatário: " +
this.destinatário + "| saldo final: " + this.saldo_final + "\n";
    }
}
```

3.3 Funções

Para cada operação que deixamos disponível para o usuário, criamos uma função referente a ela.

3.3.1 Funções *newcontaF* e *newcontaJ*

As funções *newcontaF()* e *newcontaJ()* tem como objetivo criar uma nova conta; física ou jurídica, respectivamente; para o usuário com base nas informações que foram recebidas pelo programa por meio de caixas de pergunta. Além disso, a nova conta é adicionada a uma lista de contas (*contasF*, caso seja conta de uma pessoa física ou *contasJ*, caso seja conta de uma pessoa jurídica) e será retornado o nome da pessoa responsável por essa nova conta.

```
package banco;

import java.util.*;
import javax.swing.JOptionPane;

public class Banco {
```

```
    static ArrayList<PessoaFísica> contasF = new ArrayList<>();
    static ArrayList<PessoaJurídica> contasJ = new ArrayList<>();

    public static String newcontaF(){
        String pessoaFísicaNome;
        String pessoaFísicaCPF;
        String pessoaFísicaSenha;

        pessoaFísicaNome = JOptionPane.showInputDialog("Qual é o
nome? ");
        pessoaFísicaCPF = JOptionPane.showInputDialog("Qual é o CPF?
");
        pessoaFísicaSenha = JOptionPane.showInputDialog("Qual é a
senha? ");

        contasF.add(new PessoaFísica(pessoaFísicaCPF,
pessoaFísicaSenha, pessoaFísicaNome));

        return pessoaFísicaNome;
    }

    public static String newcontaJ(){
        String pessoaJurídicaNome;
        String pessoaJurídicaCNPJ;
        String pessoaJurídicaSenha;

        pessoaJurídicaNome = JOptionPane.showInputDialog("Qual é o
nome? ");
        pessoaJurídicaCNPJ = JOptionPane.showInputDialog("Qual é o
CNPJ? ");
        pessoaJurídicaSenha = JOptionPane.showInputDialog("Qual é a
senha? ");

        contasJ.add(new PessoaJurídica(pessoaJurídicaCNPJ,
pessoaJurídicaSenha, pessoaJurídicaNome));

        return pessoaJurídicaNome;
    }
}
```

3.3.2 Funções *login_ContaFísica* e *login_ContaJurídica*

As funções *login_ContaFísica()* e *login_ContaJurídica()* tem como objetivo permitir o usuário acessar sua conta, caso o mesmo ainda não tenha, criar uma nova conta; física ou jurídica, respectivamente. Se o usuário comunicar que já possui uma conta, física ou jurídica, a função irá procurá-la nas listas *contasF* ou *contasJ*, respectivamente, com base nas informações que foram recebidas pelo programa por meio de caixas de pergunta.

```
public class Banco {
    static ArrayList<PessoaFísica> contasF = new ArrayList<>();
    static ArrayList<PessoaJurídica> contasJ = new ArrayList<>();
    static int indice_CLIENTE;

    public static void login_ContaFísica(){
        indice_CLIENTE = -1;
        int login = JOptionPane.showConfirmDialog(null, "Já possui
conta?", "Selecione uma opção:", JOptionPane.YES_NO_OPTION);
            if (login == 0){
                String clienteNovoVelhoNome =
JOptionPane.showInputDialog("Qual o nome? ");
                String clienteNovoVelhoSenha =
JOptionPane.showInputDialog("Qual a senha? ");
                for (int k = 0; k < contasF.size(); k++){
                    if
((contasF.get(k).getNome().equals(clienteNovoVelhoNome)) &&
(contasF.get(k).getSenha().equals(clienteNovoVelhoSenha))){
                        JOptionPane.showMessageDialog(null, "Conta
confirmada!");
                        indice_CLIENTE = k;
                        break;
                    }
                }
                if (indice_CLIENTE == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                    JOptionPane.showMessageDialog(null, "Seja bem
vindo " + newcontaF() + " !");
                    indice_CLIENTE = contasF.size() - 1;
                }
            } else {
                JOptionPane.showMessageDialog(null, "Seja bem vindo "
```

```
+ newcontaF() + " !");
                indice_CLIENTE = contasF.size() - 1;
            }
    }

    public static void login_ContaJurídica(){
        indice_CLIENTE = -1;
        int login = JOptionPane.showConfirmDialog(null, "Já possui
conta?", "Selecione uma opção:", JOptionPane.YES_NO_OPTION);
            if (login == 0){
                String clienteNovoVelhoNome =
JOptionPane.showInputDialog("Qual é o seu nome? ");
                String clienteNovoVelhoSenha =
JOptionPane.showInputDialog("Qual é a sua senha? ");
                for (int k = 0; k < contasJ.size(); k++){
                    if
((contasJ.get(k).getNome().equals(clienteNovoVelhoNome)) &&
(contasJ.get(k).getSenha().equals(clienteNovoVelhoSenha))){
                        JOptionPane.showMessageDialog(null, "Olá " +
contasJ.get(k).getNome() + " !");
                        indice_CLIENTE = k;
                        break;
                    }
                }
                if (indice_CLIENTE == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                    JOptionPane.showMessageDialog(null, "Seja bem
vindo " + newcontaJ() + " !");
                    indice_CLIENTE = contasJ.size() - 1;
                }
            } else {
                JOptionPane.showMessageDialog(null, "Seja bem vindo "
+ newcontaJ() + " !");
                indice_CLIENTE = contasJ.size() - 1;
            }
    }
```

3.3.3 Função *Saque*

A função *Saque(double valor_saldo)* tem como objetivo retirar uma quantia de dinheiro, especificada pelo usuário por meio de caixas de pergunta, do saldo de sua conta. A operação só será realizada se essa quantia for menor ou igual ao saldo que o usuário possuir na conta, caso contrário, aparecerá uma mensagem informando que o saldo é insuficiente.

```
public class Banco {
    static ArrayList<PessoaFísica> contasF = new ArrayList<>();
    static ArrayList<PessoaJurídica> contasJ = new ArrayList<>();
    static double valor_saque;

    public static double Saque(double valor_saldo){
        valor_saque =
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
saque?"));
        if (valor_saldo > valor_saque){
            valor_saldo -= valor_saque;
            JOptionPane.showMessageDialog(null, "Saque realizado.");
            return valor_saldo;
        } else {
            JOptionPane.showMessageDialog(null, "O saque não pode ser
realizado. Saldo insuficiente.");
            int Saque_inesperado =
JOptionPane.showConfirmDialog(null, "Deseja trocar o valor do
saque?", "Selecione uma opção:", JOptionPane.YES_NO_OPTION);
            if (Saque_inesperado == -1){
                valor_saldo = Saque(valor_saldo);
                return valor_saldo;
            } else {
                return valor_saldo;
            }
        }
    }
}
```

3.3.4 Função *Depósito*

A função *Depósito(double valor_saldo)* tem como objetivo depositar uma quantia de dinheiro, especificada pelo usuário por meio de caixas de pergunta, ao saldo de sua conta.

```
public class Banco {
    static ArrayList<PessoaFísica> contasF = new ArrayList<>();
    static ArrayList<PessoaJurídica> contasJ = new ArrayList<>();
    static double valor_depositado;

    public static double Depósito(double valor_saldo){
        valor_depositado =
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
depósito?"));
        valor_saldo += valor_depositado;
        JOptionPane.showMessageDialog(null, "Depósito realizado.");
        return valor_saldo;
    }
}
```

3.3.5 Função *Transferências_ContaFísica*

A função *Transferências_ContaFísica()* tem como objetivo executar uma transferência da conta de pessoa física do usuário para uma outra conta, física (função *TransferênciaCF_CF*) ou jurídica (função *TransferênciaCJ_CJ*), cujas informações básicas para encontrá-la são informadas por este remetente.

```
    public static void Transferências_ContaFísica(){
        indice_pagamento = -1;
        Object[] transferências = {"Transferência para Cliente
Físico", "Transferência para Cliente Jurídico"};
            Object selectedTransfers =
JOptionPane.showInputDialog(null, "Que tipo de transferência deseja
realizar?", "transferência", JOptionPane.INFORMATION_MESSAGE, null,
transferências, transferências[0]);
            if (selectedTransfers == transferências[0]){
                String clientePagoNome =
JOptionPane.showInputDialog("Qual o nome do destinatário? ");
                String clientePagoCPF =
```

```
JOptionPane.showInputDialog("Qual o CPF do destinatário? ");
                for (int m = 0; m < contasF.size(); m++){
                    if
((contasF.get(m).getNome().equals(clientePagoNome)) &&
(contasF.get(m).getCPF().equals(clientePagoCPF))){
                        JOptionPane.showMessageDialog(null, "Conta
encontrada!");
                        indice_pagamento = m;

TransferênciaCF_CF(contasF.get(indice_CLIENTE),
contasF.get(indice_pagamento));
                        break;
                    }
                }
                if (indice_pagamento == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                }
            } else {
                String clientePagoNome =
JOptionPane.showInputDialog("Qual o nome do destinatário? ");
                String clientePagoCNPJ =
JOptionPane.showInputDialog("Qual o CNPJ do destinatário? ");
                for (int m = 0; m < 10; m++){
                    if
((contasJ.get(m).getNome().equals(clientePagoNome)) &&
(contasJ.get(m).getCNPJ().equals(clientePagoCNPJ))){
                        JOptionPane.showMessageDialog(null, "Conta
encontrada!");
                        indice_pagamento = m;

TransferênciaCF_CJ(contasF.get(indice_CLIENTE),
contasJ.get(indice_pagamento));
                        break;
                    }
                }
                if (indice_pagamento == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                }
            }
    }
```

- Função *TransferênciaCF_CF*:

A função *TransferênciaCF_CF(PessoaFísica cliente_perde, PessoaFísica cliente_ganha)* tem como objetivo transferir uma quantia de dinheiro, especificada pelo usuário por meio de caixas de pergunta, da conta do usuário para outra, sendo ambas contas de pessoa física. Além disso, esta função acrescenta os dados dessa transação à lista de transações de cada conta envolvida nesse processo.

```
    public static void TransferênciaCF_CF(PessoaFísica cliente_perde,
PessoaFísica cliente_ganha){
        valor_troca =
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
pagamento?"));
        cliente_perde.setSaldo(cliente_perde.getSaldo() -
valor_troca);
        cliente_ganha.setSaldo(cliente_ganha.getSaldo() +
valor_troca);
        cliente_perde.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_perde.getSaldo()));
        cliente_ganha.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_ganha.getSaldo()));
        JOptionPane.showMessageDialog(null, "Transferência
realizada.");
    }
```

- Função *TransferênciaCF_CJ*:

A função *TransferênciaCF_CJ(PessoaFísica cliente_perde, PessoaJurídica cliente_ganha)* tem como objetivo transferir uma quantia de dinheiro, especificada pelo usuário por meio de caixas de pergunta, da conta do usuário para outra, sendo que as contas são de pessoa física e jurídica, respectivamente. Além disso, esta função acrescenta os dados dessa transação à lista de transações de cada conta envolvida nesse processo.

```
    public static void TransferênciaCF_CJ(PessoaFísica cliente_perde,
PessoaJurídica cliente_ganha){
```

```
        valor_troca =
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
pagamento?"));
        cliente_perde.setSaldo(cliente_perde.getSaldo() -
valor_troca);
        cliente_ganha.setSaldo(cliente_ganha.getSaldo() +
valor_troca);
        cliente_perde.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_perde.getSaldo()));
        cliente_ganha.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_ganha.getSaldo()));
        JOptionPane.showMessageDialog(null, "Transferência
realizada.");
    }
```

3.3.6 Função *Transferências_ContaJurídica*

A função *Transferências_ContaJurídica()* tem como objetivo executar uma transferência da conta de pessoa jurídica do usuário para uma outra conta, física (função *TransferênciaCJ_CF*) ou jurídica (função *TransferênciaCJ_CJ*), cujas informações básicas para encontrá-la são informadas por este remetente.

```
    public static void Transferências_ContaJurídica(){
        indice_pagamento = -1;
        Object[] transferências = {"Transferência para Cliente
Físico", "Transferência para Cliente Jurídico"};
            Object selectedTransfers =
JOptionPane.showInputDialog(null, "Que tipo de transferência deseja
realizar?", "transferência", JOptionPane.INFORMATION_MESSAGE, null,
transferências, transferências[0]);
            if (selectedTransfers == transferências[0]){
                String clientePagoNome =
JOptionPane.showInputDialog("Qual o nome do destinatário? ");
                String clientePagoCPF =
JOptionPane.showInputDialog("Qual o CPF do destinatário? ");
                for (int m = 0; m < contasF.size(); m++){
                    if
((contasF.get(m).getNome().equals(clientePagoNome)) &&
```

```
(contasF.get(m).getCPF().equals(clientePagoCPF))){
                        JOptionPane.showMessageDialog(null, "Conta
encontrada!");
                        indice_pagamento = m;

TransferênciaCJ_CF(contasJ.get(indice_CLIENTE),
contasF.get(indice_pagamento));
                        break;
                    }
                }
                if (indice_pagamento == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                }
            } else {
                String clientePagoNome =
JOptionPane.showInputDialog("Qual o nome do destinatário? ");
                String clientePagoCNPJ =
JOptionPane.showInputDialog("Qual o CNPJ do destinatário? ");
                for (int m = 0; m < 10; m++){
                    if
((contasJ.get(m).getNome().equals(clientePagoNome)) &&
(contasJ.get(m).getCNPJ().equals(clientePagoCNPJ))){
                        JOptionPane.showMessageDialog(null, "Conta
encontrada!");
                        indice_pagamento = m;

TransferênciaCJ_CJ(contasJ.get(indice_CLIENTE),
contasJ.get(indice_pagamento));
                        break;
                    }
                }
                if (indice_pagamento == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                }
            }
    }
```

- Função *TransferênciaCJ_CF*:

A função *TransferênciaCJ_CF(PessoaJurídica cliente_perde, PessoaFísica cliente_ganha)* tem como objetivo transferir uma quantia de dinheiro, especificada pelo usuário por meio de caixas de pergunta, da conta do usuário para outra, sendo que as contas são de pessoa jurídica e física, respectivamente. Além disso, esta função acrescenta os dados dessa transação à lista de transações de cada conta envolvida nesse processo.

```
    public static void TransferênciaCJ_CF(PessoaJurídica
cliente_perde, PessoaFísica cliente_ganha){
        valor_troca =
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
pagamento?"));
        cliente_perde.setSaldo(cliente_perde.getSaldo() -
valor_troca);
        cliente_ganha.setSaldo(cliente_ganha.getSaldo() +
valor_troca);
        cliente_perde.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_perde.getSaldo()));
        cliente_ganha.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_ganha.getSaldo()));
        JOptionPane.showMessageDialog(null, "Transferência
realizada.");
    }
```

- Função *TransferênciaCJ_CJ*:

A função *TransferênciaCJ_CJ(PessoaJurídica cliente_perde, PessoaJurídica cliente_ganha)* tem como objetivo transferir uma quantia de dinheiro, especificada pelo usuário por meio de caixas de pergunta, da conta do usuário para outra, sendo ambas contas de pessoa jurídica. Além disso, esta função acrescenta os dados dessa transação à lista de transações de cada conta envolvida nesse processo.

```
    public static void TransferênciaCJ_CJ(PessoaJurídica
cliente_perde, PessoaJurídica cliente_ganha){
        valor_troca =
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
```

```
pagamento?"));
        cliente_perde.setSaldo(cliente_perde.getSaldo() -
valor_troca);
        cliente_ganha.setSaldo(cliente_ganha.getSaldo() +
valor_troca);
        cliente_perde.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_perde.getSaldo()));
        cliente_ganha.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_ganha.getSaldo()));
        JOptionPane.showMessageDialog(null, "Transferência
realizada.");
    }
```

3.3.7 Função *Extrato_Transações*

A função *Extrato_Transações(ArrayList<Transações> Extrato)* fará com que apareça no espaço de *output* do sistema uma matriz em que os elementos terão as informações de cada transação realizada que envolva o usuário.

```
    public static void Extrato_Transações(ArrayList<Transações>
Extrato){
        System.out.println(Extrato);
    }
```

Observe que o retorno dessa função não é o padrão, isso aconteceu por termos reescrito o método *toString()* na classe ***Transações***.

```
    @Override
    public String toString() {
        return this.operação + "| valor: " + this.valor + "|
remetente: " + this.remetente + "| destinatário: " +
this.destinatário + "| saldo final: " + this.saldo_final + "\n";
    }
```

3.3.8 Funções *Operações_CF* e *Operações_CJ*

As funções *Operações_CF()* e *Operações_CJ()* perguntam e executam as funções relacionadas à operação que o usuário deseja realizar, tendo saque, depósito, transferência e visualizar extrato como opções. Além disso, acrescenta os dados das transações de saque e depósito à lista de transações da conta do usuário, seja ela de pessoa física ou pública, respectivamente.

```
    public static void Operações_CF(){
        Object[] operations = {"Saque", "Depósito", "Transferência",
"Extrato"};
            Object selectedOperation =
JOptionPane.showInputDialog(null, "Que operação deseja realizar?",
"operação", JOptionPane.INFORMATION_MESSAGE, null, operations,
operations[0]);
            double novo_saldo;
            Transações transação;
        if (selectedOperation == operations[0]){
            novo_saldo =
Saque(contasF.get(indice_CLIENTE).getSaldo());
            contasF.get(indice_CLIENTE).setSaldo(novo_saldo);
            transação = new Transações ("Saque", valor_saque,
contasF.get(indice_CLIENTE).getNome(),
contasF.get(indice_CLIENTE).getNome(),
contasF.get(indice_CLIENTE).getSaldo());
            contasF.get(indice_CLIENTE).setListaT(transação);
        } else if (selectedOperation == operations[1]) {
            novo_saldo =
Depósito(contasF.get(indice_CLIENTE).getSaldo());
            contasF.get(indice_CLIENTE).setSaldo(novo_saldo);
            transação = new Transações ("Depósito", valor_depositado,
contasF.get(indice_CLIENTE).getNome(),
contasF.get(indice_CLIENTE).getNome(),
contasF.get(indice_CLIENTE).getSaldo());
            contasF.get(indice_CLIENTE).setListaT(transação);
        } else if (selectedOperation == operations[2]){
            Transferências_ContaFísica();
        } else {

Extrato_Transações(contasF.get(indice_CLIENTE).getListaT());
        }
    }
```

```
    public static void Operações_CJ(){
        Object[] operations = {"Saque", "Depósito", "Transferência",
"Extrato"};
            Object selectedOperation =
JOptionPane.showInputDialog(null, "Que operação deseja realizar?",
"operação", JOptionPane.INFORMATION_MESSAGE, null, operations,
operations[0]);
            double novo_saldo;
            Transações transação;
        if (selectedOperation == operations[0]){
            novo_saldo =
Saque(contasJ.get(indice_CLIENTE).getSaldo());
            contasJ.get(indice_CLIENTE).setSaldo(novo_saldo);
            transação = new Transações ("Saque", valor_saque,
contasJ.get(indice_CLIENTE).getNome(),
contasJ.get(indice_CLIENTE).getNome(),
contasJ.get(indice_CLIENTE).getSaldo());
            contasJ.get(indice_CLIENTE).setListaT(transação);
        } else if (selectedOperation == operations[1]) {
            novo_saldo =
Depósito(contasJ.get(indice_CLIENTE).getSaldo());
            contasJ.get(indice_CLIENTE).setSaldo(novo_saldo);
            transação = new Transações ("Depósito", valor_depositado,
contasJ.get(indice_CLIENTE).getNome(),
contasJ.get(indice_CLIENTE).getNome(),
contasJ.get(indice_CLIENTE).getSaldo());
            contasJ.get(indice_CLIENTE).setListaT(transação);
        } else if (selectedOperation == operations[2]){
            Transferências_ContaJurídica();
        } else {

Extrato_Transações(contasJ.get(indice_CLIENTE).getListaT());
        }
    }
```

3.3.9 Função *Atendimento_Cliente*

A função *Atendimento_Cliente()* pergunta que tipo de “cliente” o usuário é, pessoa física ou jurídica, executa a função *login* e em seguida a função *operações*. Após esse procedimento padrão, questiona se o usuário deseja realizar mais alguma

operação, caso a resposta seja afirmativa, executa a função *operações* novamente, caso contrário, se despede do usuário.

```
    public static void Atendimento_Cliente() {
        Object[] pessoas = {"Cliente físico", "Cliente jurídico"};
            Object selectedPerson = JOptionPane.showInputDialog(null,
"Quem é você?", "pessoa", JOptionPane.INFORMATION_MESSAGE, null,
pessoas, pessoas[0]);

        if (selectedPerson == pessoas[0]){
            login_ContaFísica();
            do {
                Operações_CF();
                Continuar = JOptionPane.showConfirmDialog(null,
"Deseja realizar mais alguma operação?", "Selecione uma opção:",
JOptionPane.YES_NO_OPTION);
            } while (Continuar == 0);
            JOptionPane.showMessageDialog(null, "Obrigado, tenha um
bom dia " + contasF.get(indice_CLIENTE).getNome());
        } else {
            login_ContaJurídica();
            do {
                Operações_CJ();
                Continuar = JOptionPane.showConfirmDialog(null,
"Deseja realizar mais alguma operação?", "Selecione uma opção:",
JOptionPane.YES_NO_OPTION);
            } while (Continuar == 0);
            JOptionPane.showMessageDialog(null, "Obrigado, tenha um
bom dia " + contasJ.get(indice_CLIENTE).getNome());
        }
    }
```

3.4 Função *main*

A função *main(String[] args)* executa a função *Atendimento_Cliente()* até que se complete a quantidade de dez atendimentos (dez foi um número aleatório escolhido, poderia ser qualquer outro número inteiro).

```
public static void main(String[] args) {
        do {
```

```
            Atendimento_Cliente();
            atendimentos++;
            if (atendimentos == 10){
                atendimento = "off";
            }
        } while (atendimento.equals("on"));
    }
```

4 Código completo da classe main

A seguir está apresentado o código completo da classe main:

```
package banco;

import java.util.*;
import javax.swing.JOptionPane;

public class Banco {
    static ArrayList<PessoaFísica> contasF = new ArrayList<>();
    static ArrayList<PessoaJurídica> contasJ = new ArrayList<>();
    static double valor_saque;
    static double valor_depositado;
    static double valor_troca;
    static int indice_CLIENTE;
    static int indice_pagamento;
    static int Continuar = 0;
    static int atendimentos = 0;
    static String atendimento = "on";

    public static String newcontaF(){
        String pessoaFísicaNome;
        String pessoaFísicaCPF;
        String pessoaFísicaSenha;

        pessoaFísicaNome = JOptionPane.showInputDialog("Qual é o
nome? ");
        pessoaFísicaCPF = JOptionPane.showInputDialog("Qual é o CPF?
");
        pessoaFísicaSenha = JOptionPane.showInputDialog("Qual é a
senha? ");
```

```
        contasF.add(new PessoaFísica(pessoaFísicaCPF,
pessoaFísicaSenha, pessoaFísicaNome));

        return pessoaFísicaNome;
    }

    public static String newcontaJ(){
        String pessoaJurídicaNome;
        String pessoaJurídicaCNPJ;
        String pessoaJurídicaSenha;

        pessoaJurídicaNome = JOptionPane.showInputDialog("Qual é o
nome? ");
        pessoaJurídicaCNPJ = JOptionPane.showInputDialog("Qual é o
CNPJ? ");
        pessoaJurídicaSenha = JOptionPane.showInputDialog("Qual é a
senha? ");

        contasJ.add(new PessoaJurídica(pessoaJurídicaCNPJ,
pessoaJurídicaSenha, pessoaJurídicaNome));

        return pessoaJurídicaNome;
    }

    public static void login_ContaFísica(){
        indice_CLIENTE = -1;
        int login = JOptionPane.showConfirmDialog(null, "Já possui
conta?", "Selecione uma opção:", JOptionPane.YES_NO_OPTION);
            if (login == 0){
                String clienteNovoVelhoNome =
JOptionPane.showInputDialog("Qual o nome? ");
                String clienteNovoVelhoSenha =
JOptionPane.showInputDialog("Qual a senha? ");
                for (int k = 0; k < contasF.size(); k++){
                    if
((contasF.get(k).getNome().equals(clienteNovoVelhoNome)) &&
(contasF.get(k).getSenha().equals(clienteNovoVelhoSenha))){
                        JOptionPane.showMessageDialog(null, "Conta
confirmada!");
                        indice_CLIENTE = k;
                        break;
```

```
                    }
                }
                if (indice_CLIENTE == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                    JOptionPane.showMessageDialog(null, "Seja bem
vindo " + newcontaF() + " !");
                    indice_CLIENTE = contasF.size() - 1;
                }
            } else {
                JOptionPane.showMessageDialog(null, "Seja bem vindo "
+ newcontaF() + " !");
                indice_CLIENTE = contasF.size() - 1;
            }
    }

    public static void login_ContaJurídica(){
        indice_CLIENTE = -1;
        int login = JOptionPane.showConfirmDialog(null, "Já possui
conta?", "Selecione uma opção:", JOptionPane.YES_NO_OPTION);
            if (login == 0){
                String clienteNovoVelhoNome =
JOptionPane.showInputDialog("Qual é o seu nome? ");
                String clienteNovoVelhoSenha =
JOptionPane.showInputDialog("Qual é a sua senha? ");
                for (int k = 0; k < contasJ.size(); k++){
                    if
((contasJ.get(k).getNome().equals(clienteNovoVelhoNome)) &&
(contasJ.get(k).getSenha().equals(clienteNovoVelhoSenha))){
                        JOptionPane.showMessageDialog(null, "Olá " +
contasJ.get(k).getNome() + " !");
                        indice_CLIENTE = k;
                        break;
                    }
                }
                if (indice_CLIENTE == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                    JOptionPane.showMessageDialog(null, "Seja bem
vindo " + newcontaJ() + " !");
                    indice_CLIENTE = contasJ.size() - 1;
                }
```

```
            } else {
                JOptionPane.showMessageDialog(null, "Seja bem vindo "
+ newcontaJ() + " !");
                indice_CLIENTE = contasJ.size() - 1;
            }
    }

    public static double Saque(double valor_saldo){
        valor_saque =
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
saque?"));
        if (valor_saldo > valor_saque){
            valor_saldo -= valor_saque;
            JOptionPane.showMessageDialog(null, "Saque realizado.");
            return valor_saldo;
        } else {
            JOptionPane.showMessageDialog(null, "O saque não pode ser
realizado. Saldo insuficiente.");
            int Saque_inesperado =
JOptionPane.showConfirmDialog(null, "Deseja trocar o valor do
saque?", "Selecione uma opção:", JOptionPane.YES_NO_OPTION);
            if (Saque_inesperado == 0){
                valor_saldo = Saque(valor_saldo);
                return valor_saldo;
            } else {
                return valor_saldo;
            }
        }
    }

    public static double Depósito(double valor_saldo){
        valor_depositado =
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
depósito?"));
        valor_saldo += valor_depositado;
        JOptionPane.showMessageDialog(null, "Depósito realizado.");
        return valor_saldo;
    }

    public static void TransferênciaCF_CJ(PessoaFísica cliente_perde,
PessoaJurídica cliente_ganha){
        valor_troca =
```

```
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
pagamento?"));
        cliente_perde.setSaldo(cliente_perde.getSaldo() -
valor_troca);
        cliente_ganha.setSaldo(cliente_ganha.getSaldo() +
valor_troca);
        cliente_perde.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_perde.getSaldo()));
        cliente_ganha.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_ganha.getSaldo()));
        JOptionPane.showMessageDialog(null, "Transferência
realizada.");
    }

    public static void TransferênciaCJ_CF(PessoaJurídica
cliente_perde, PessoaFísica cliente_ganha){
        valor_troca =
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
pagamento?"));
        cliente_perde.setSaldo(cliente_perde.getSaldo() -
valor_troca);
        cliente_ganha.setSaldo(cliente_ganha.getSaldo() +
valor_troca);
        cliente_perde.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_perde.getSaldo()));
        cliente_ganha.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_ganha.getSaldo()));
        JOptionPane.showMessageDialog(null, "Transferência
realizada.");
    }

    public static void TransferênciaCF_CF(PessoaFísica cliente_perde,
PessoaFísica cliente_ganha){
        valor_troca =
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
pagamento?"));
        cliente_perde.setSaldo(cliente_perde.getSaldo() -
valor_troca);
```

```
        cliente_ganha.setSaldo(cliente_ganha.getSaldo() +
valor_troca);
        cliente_perde.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_perde.getSaldo()));
        cliente_ganha.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_ganha.getSaldo()));
        JOptionPane.showMessageDialog(null, "Transferência
realizada.");
    }

    public static void TransferênciaCJ_CJ(PessoaJurídica
cliente_perde, PessoaJurídica cliente_ganha){
        valor_troca =
Double.parseDouble(JOptionPane.showInputDialog("Qual será o valor do
pagamento?"));
        cliente_perde.setSaldo(cliente_perde.getSaldo() -
valor_troca);
        cliente_ganha.setSaldo(cliente_ganha.getSaldo() +
valor_troca);
        cliente_perde.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_perde.getSaldo()));
        cliente_ganha.setListaT(new Transações ("Transferência",
valor_troca, cliente_perde.getNome(), cliente_ganha.getNome(),
cliente_ganha.getSaldo()));
        JOptionPane.showMessageDialog(null, "Transferência
realizada.");
    }

    public static void Transferências_ContaFísica(){
        indice_pagamento = -1;
        Object[] transferências = {"Transferência para Cliente
Físico", "Transferência para Cliente Jurídico"};
            Object selectedTransfers =
JOptionPane.showInputDialog(null, "Que tipo de transferência deseja
realizar?", "transferência", JOptionPane.INFORMATION_MESSAGE, null,
transferências, transferências[0]);
            if (selectedTransfers == transferências[0]){
                String clientePagoNome =
JOptionPane.showInputDialog("Qual o nome do destinatário? ");
```

```
                String clientePagoCPF =
JOptionPane.showInputDialog("Qual o CPF do destinatário? ");
                for (int m = 0; m < contasF.size(); m++){
                    if
((contasF.get(m).getNome().equals(clientePagoNome)) &&
(contasF.get(m).getCPF().equals(clientePagoCPF))){
                        JOptionPane.showMessageDialog(null, "Conta
encontrada!");
                        indice_pagamento = m;

TransferênciaCF_CF(contasF.get(indice_CLIENTE),
contasF.get(indice_pagamento));
                        break;
                    }
                }
                if (indice_pagamento == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                }
            } else {
                String clientePagoNome =
JOptionPane.showInputDialog("Qual o nome do destinatário? ");
                String clientePagoCNPJ =
JOptionPane.showInputDialog("Qual o CNPJ do destinatário? ");
                for (int m = 0; m < 10; m++){
                    if
((contasJ.get(m).getNome().equals(clientePagoNome)) &&
(contasJ.get(m).getCNPJ().equals(clientePagoCNPJ))){
                        JOptionPane.showMessageDialog(null, "Conta
encontrada!");
                        indice_pagamento = m;

TransferênciaCF_CJ(contasF.get(indice_CLIENTE),
contasJ.get(indice_pagamento));
                        break;
                    }
                }
                if (indice_pagamento == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                }
            }
```

```
    }

    public static void Transferências_ContaJurídica(){
        indice_pagamento = -1;
        Object[] transferências = {"Transferência para Cliente
Físico", "Transferência para Cliente Jurídico"};
            Object selectedTransfers =
JOptionPane.showInputDialog(null, "Que tipo de transferência deseja
realizar?", "transferência", JOptionPane.INFORMATION_MESSAGE, null,
transferências, transferências[0]);
            if (selectedTransfers == transferências[0]){
                String clientePagoNome =
JOptionPane.showInputDialog("Qual o nome do destinatário? ");
                String clientePagoCPF =
JOptionPane.showInputDialog("Qual o CPF do destinatário? ");
                for (int m = 0; m < contasF.size(); m++){
                    if
((contasF.get(m).getNome().equals(clientePagoNome)) &&
(contasF.get(m).getCPF().equals(clientePagoCPF))){
                        JOptionPane.showMessageDialog(null, "Conta
encontrada!");
                        indice_pagamento = m;

TransferênciaCJ_CF(contasJ.get(indice_CLIENTE),
contasF.get(indice_pagamento));
                        break;
                    }
                }
                if (indice_pagamento == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                }
            } else {
                String clientePagoNome =
JOptionPane.showInputDialog("Qual o nome do destinatário? ");
                String clientePagoCNPJ =
JOptionPane.showInputDialog("Qual o CNPJ do destinatário? ");
                for (int m = 0; m < 10; m++){
                    if
((contasJ.get(m).getNome().equals(clientePagoNome)) &&
(contasJ.get(m).getCNPJ().equals(clientePagoCNPJ))){
                        JOptionPane.showMessageDialog(null, "Conta
```

```
encontrada!");
                        indice_pagamento = m;

TransferênciaCJ_CJ(contasJ.get(indice_CLIENTE),
contasJ.get(indice_pagamento));
                        break;
                    }
                }
                if (indice_pagamento == -1){
                    JOptionPane.showMessageDialog(null, "Essa conta
não existe!");
                }
            }
    }

    public static void Extrato_Transações(ArrayList<Transações>
Extrato){
        System.out.println(Extrato);
    }

    public static void Operações_CF(){
        Object[] operations = {"Saque", "Depósito", "Transferência",
"Extrato"};
            Object selectedOperation =
JOptionPane.showInputDialog(null, "Que operação deseja realizar?",
"operação", JOptionPane.INFORMATION_MESSAGE, null, operations,
operations[0]);
            double novo_saldo;
            Transações transação;
        if (selectedOperation == operations[0]){
            novo_saldo =
Saque(contasF.get(indice_CLIENTE).getSaldo());
            contasF.get(indice_CLIENTE).setSaldo(novo_saldo);
            transação = new Transações ("Saque", valor_saque,
contasF.get(indice_CLIENTE).getNome(),
contasF.get(indice_CLIENTE).getNome(),
contasF.get(indice_CLIENTE).getSaldo());
            contasF.get(indice_CLIENTE).setListaT(transação);
        } else if (selectedOperation == operations[1]) {
            novo_saldo =
Depósito(contasF.get(indice_CLIENTE).getSaldo());
            contasF.get(indice_CLIENTE).setSaldo(novo_saldo);
```

```
            transação = new Transações ("Depósito", valor_depositado,
contasF.get(indice_CLIENTE).getNome(),
contasF.get(indice_CLIENTE).getNome(),
contasF.get(indice_CLIENTE).getSaldo());
            contasF.get(indice_CLIENTE).setListaT(transação);
        } else if (selectedOperation == operations[2]){
            Transferências_ContaFísica();
        } else {

Extrato_Transações(contasF.get(indice_CLIENTE).getListaT());
        }
    }

    public static void Operações_CJ(){
        Object[] operations = {"Saque", "Depósito", "Transferência",
"Extrato"};
            Object selectedOperation =
JOptionPane.showInputDialog(null, "Que operação deseja realizar?",
"operação", JOptionPane.INFORMATION_MESSAGE, null, operations,
operations[0]);
            double novo_saldo;
            Transações transação;
        if (selectedOperation == operations[0]){
            novo_saldo =
Saque(contasJ.get(indice_CLIENTE).getSaldo());
            contasJ.get(indice_CLIENTE).setSaldo(novo_saldo);
            transação = new Transações ("Saque", valor_saque,
contasJ.get(indice_CLIENTE).getNome(),
contasJ.get(indice_CLIENTE).getNome(),
contasJ.get(indice_CLIENTE).getSaldo());
            contasJ.get(indice_CLIENTE).setListaT(transação);
        } else if (selectedOperation == operations[1]) {
            novo_saldo =
Depósito(contasJ.get(indice_CLIENTE).getSaldo());
            contasJ.get(indice_CLIENTE).setSaldo(novo_saldo);
            transação = new Transações ("Depósito", valor_depositado,
contasJ.get(indice_CLIENTE).getNome(),
contasJ.get(indice_CLIENTE).getNome(),
contasJ.get(indice_CLIENTE).getSaldo());
            contasJ.get(indice_CLIENTE).setListaT(transação);
        } else if (selectedOperation == operations[2]){
            Transferências_ContaJurídica();
```

```
        } else {

Extrato_Transações(contasJ.get(indice_CLIENTE).getListaT());
        }
    }

    public static void Atendimento_Cliente() {
        Object[] pessoas = {"Cliente físico", "Cliente jurídico"};
            Object selectedPerson = JOptionPane.showInputDialog(null,
"Quem é você?", "pessoa", JOptionPane.INFORMATION_MESSAGE, null,
pessoas, pessoas[0]);

        if (selectedPerson == pessoas[0]){
            login_ContaFísica();
            do {
                Operações_CF();
                Continuar = JOptionPane.showConfirmDialog(null,
"Deseja realizar mais alguma operação?", "Selecione uma opção:",
JOptionPane.YES_NO_OPTION);
            } while (Continuar == 0);
            JOptionPane.showMessageDialog(null, "Obrigado, tenha um
bom dia " + contasF.get(indice_CLIENTE).getNome());
        } else {
            login_ContaJurídica();
            do {
                Operações_CJ();
                Continuar = JOptionPane.showConfirmDialog(null,
"Deseja realizar mais alguma operação?", "Selecione uma opção:",
JOptionPane.YES_NO_OPTION);
            } while (Continuar == 0);
            JOptionPane.showMessageDialog(null, "Obrigado, tenha um
bom dia " + contasJ.get(indice_CLIENTE).getNome());
        }
    }

    public static void main(String[] args) {
        do {
            Atendimento_Cliente();
            atendimentos++;
            if (atendimentos == 1){
                atendimento = "off";
            }
```

```
        } while (atendimento.equals("on"));
    }
}
```

5 Conclusão

Por meio desse projeto, podemos concluir que com um conhecimento de nível intermediário em linguagem Java é possível montar um programa *Backend* e *Frontend* (pacote *javax.swing.JOptionPane*) simples de um caixa bancário. Com mais conhecimento a respeito dessa linguagem pode-se desenvolver aplicações mais complexas, não só na parte de *Backend* quanto na de *Frontend*.

Essas aplicações mais complexas poderiam ser, por exemplo, as funções de empréstimo, limite de cartão de crédito baseado nas transações, pagamento de despesas por meio de boleto e até opções para investir. Com opções mais avançadas de *Frontend*, pode-se criar telas com estéticas mais complexas.

6 Bibliografia

Os seguintes sites utilizados nesse projeto não sofreram alteração durante o semestre:

https://www.lncc.br/~rogerio/lingprog/JOptionPane.pdf

https://www.alura.com.br/conteudo/java-collections?gclid=Cj0KCQiAqbyNBhC2ARIsALDwAsCo4vBhoJLrMmv1-20Q0hLuFvSRbY28maOilUZS_KouILrZxtdFII8aAp18EALw_wcB

https://www.devmedia.com.br/explorando-a-classe-arraylist-no-java/24298

https://www.devmedia.com.br/java-declaracao-e-utilizacao-de-classes/38374

https://www.alura.com.br/conteudo/java-heranca-interfaces-polimorfismo?gclid=Cj0KCQiAqbyNBhC2ARIsALDwAsBvZfNMds_LJQ1TMur1YUHSsvspRVtmJN3yN7RwyR5ap2zXqurRjCUaAqA-EALw_wcB