



unopar

UNIVERSIDADE PITÁGORAS UNOPAR ANHANGUERA - MARAVILHA  
ENGENHARIA DE SOFTWARE

NATAN OGLIARI - 34466876

LINGUAGEM ORIENTADA A OBJETOS

Maravilha/SC

2023

NATAN OGLIARI - 34466876

## LINGUAGEM ORIENTADA A OBJETOS

Trabalho de portfólio apresentado como requisito parcial  
para a obtenção de pontos para a média semestral.

Orientador: Leonardo Santiago Sidon da Rocha.

Maravilha/SC  
2023

# Sumário

	Páginas
1 Introdução	4
2 Métodos	4
3 Resultados	7
3.1 Método de classe . . . . .	7
4 Conclusões	14
5 Anexos	16

# 1 Introdução

Esta presente aula prática tem por fim a aplicação dos paradigmas da linguagem orientada a objetos com a linguagem de programação Java1.

Para os procedimentos práticos foi sugerido o uso de nome *gerenciaBanco*, o qual será implementado. A finalidade desta aula prática, visa a implementação de um sistema de gerenciamento de banco, aonde o cliente deste banco irá acendero sistema através de uma interface gráfica através da biblioteca *java.swing.\**, o menu deverá ser do tipo *loop* o usuário deverá escolher a opção de finalizar a operação.

As funções para este programa são:

- Incerção das credenciais (Nome, Sobrenome e cpf);
- Consulta de saldo;
- Depósito;
- Saque;
- Finaliza a operação com uma mensagem de despedida;

Segundo freecodecamp (2023), a linguagem orientata a objetos se define em quatro pilares, sendo elas: herança, encapsulamento, abstração e poliformismo. Isso permite definir, neste caso, umas características de **cliente**, e utiliza-lós inúmeras vezes sem a necessidade de rescreve-lá.

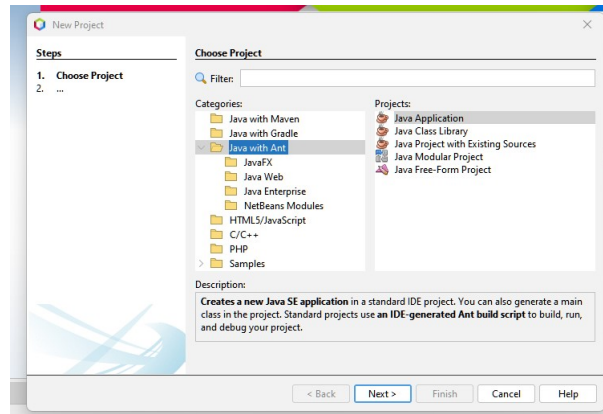
## 2 Métodos

Para a confecção desta aula prática, foi criado um repositório no **Github**, com o objetivo do aplicar os controle de versões e a utilização do sistema *git* em geral, o relatório foi confeccionado em  $\text{\LaTeX}$  com o objetivo da abstração da formatação do texto ficando livre para o aprofundamento na questão textual.

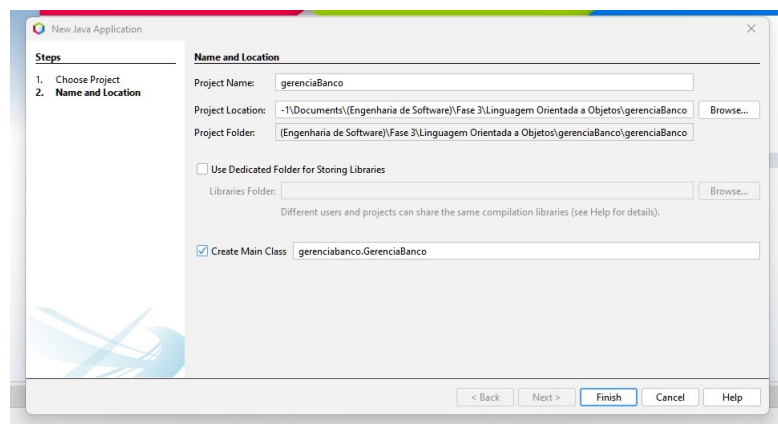
De forma subsequente as instruções do roteiro da aula prática é criado um projeto no NetBeans em java conforme figura 2.



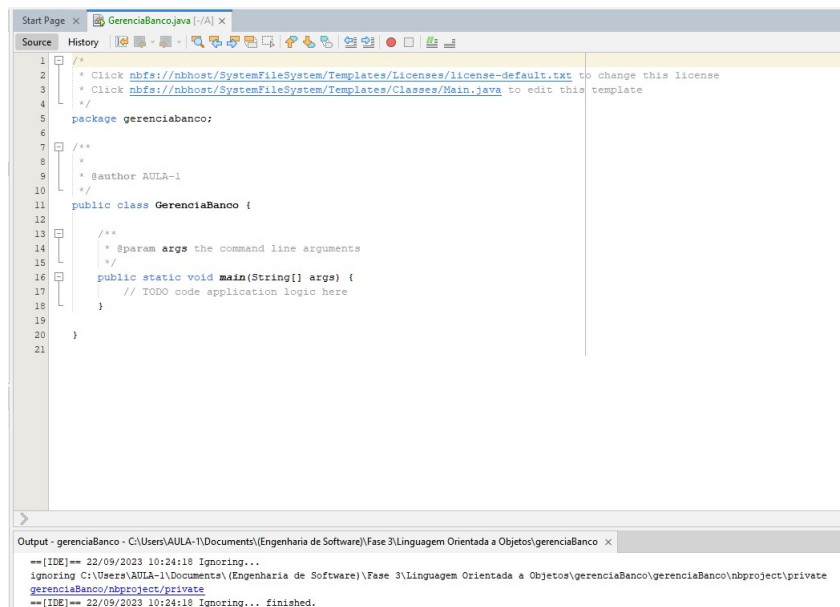
Figura 1. Logo Java, Stackify (2023)



(a) Início projeto.



(b) Nome projeto.

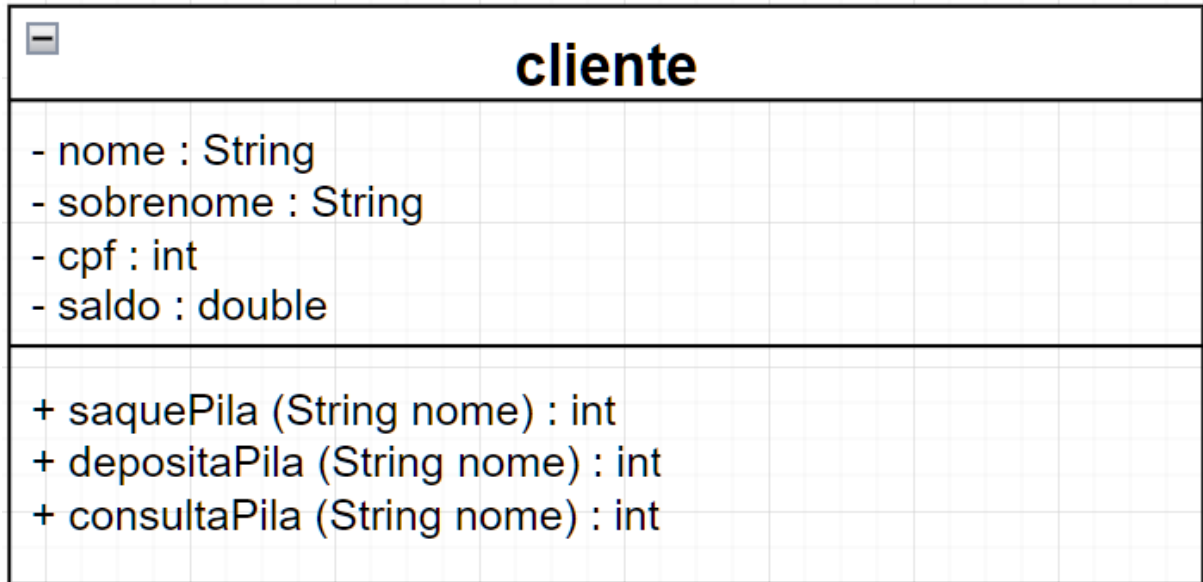


**Figura 2.** Projeto java, O autor (2023)

Na figura 2(a) solicita o tipo de projeto a ser criado, no caso *Java Application*, na figura 2(b) é solicitado o local aonde o projeto será salvo e o nome do mesmo, deste modo o nome foi

definido conforme sugestão do roteiro *gerenciaBanco* e na figura 2(c) demonstra a criação do projeto em **branco** já com a inclusão do pacote *gerenciabanco*, após estes passos foi prosseguido com a programação do que foi solicitado ao discente.

Para organização desta aula prática foi confeccionado um diagrama UML para o programa, conforme demonstra a figura 3



**Figura 3.** Diagrama UML, O autor (2023)

Foram definidos atributos para a classe e o mesmos são:

- name : String
- sobrenome : String
- cpf : String
- saldo : double

Seguindo boas práticas de programação, define-se variáveis de controle de erros, e para tal, a criação de constante de verificação em vez de zeros e uns (0,1), do tipo *private static final int STATUS\_OK = 1*; e *private static final int STATUS\_FAIL = 0*;

## 3 Resultados

Após a criação dos repositórios e do projeto, prossegue com a confecção do código<sup>1</sup>, no decorrer desta seção será incluído figuras dos resultados da presente aula.

### 3.1 Método de classe

Os métodos definem as ações da classe, e o acesso é *minhaClasse.meuMetodo(args)*, foram definidos três métodos, sendo eles:

- *consultaPila()* Responsável pela consulta do saldo do cliente, conforme código 1;
- *depositoPila()* Responsável pelo depósito na conta do cliente, conforme código 2 e
- *saquePila()* Responsável pelo saque na conta do cliente, conforme código 3.

```
1      /**
2      * Realiza a consulta do saldo de uma conta
3      * @param nome Passa o nome da conta a ser consultada e
4      *       informa o saldo ao cliente.
5      */
6      public void consultaPila(String nome) {
7          JOptionPane.showMessageDialog(null, "Seu saldo atual é de:
8          R$" + this.saldo, this.nome + " " + this.sobrenome,
          JOptionPane.INFORMATION_MESSAGE, icon);
7      }
8  }
```

**Listing 1.** *consultaPila()*

---

<sup>1</sup>código completo

```

1      /**
2      * Realiza um deposito em uma conta
3      * @param nome Informa o nome da conta a ser depositado
4      * @return retorna STATUS_OK se a operação ocorreu com sucesso
5      *         e retorna STATUS_FAIL se ocorrer um erro
6      */
7
8      public int depositaPila(String nome){
9
10         try { // verifica se a entrada e do tipo numeral
11             double pilaDeposito = Double.parseDouble(JOptionPane.
12                 showInputDialog(null, "Informe a quantidade em Reais
13                 (R$) a ser depositado na conta do "+this.nome+" "+
14                 this.sobrenome));
15             this.saldo += pilaDeposito;
16             //JOptionPane.showMessageDialog(null, "Seu Saldo é R$
17             :"+this.consultaPilas(this.nome)+" Reais", this.
18             nome, JOptionPane.INFORMATION_MESSAGE, icon);
19             return STATUS_OK;
20         }
21
22         catch (NumberFormatException e) { // imprime o erro na
23             tela e informa o que foi digitado.
24             JOptionPane.showMessageDialog(null, "Entre com valor
25             válido, do tipo numeral.\n Use (.) ponto em vez de
26             (,) virgula\n ERRO: " + e.getMessage() , "ERRO",
27             JOptionPane.ERROR_MESSAGE);
28             return STATUS_FAIL;
29         }
30     }
31 }

```

**Listing 2.** *depositaPila()*

```

1      /**
2      * Realiza o saque de uma conta
3      * @param nome Informa o nome da conta a ser realizado o saque
4      * @return retorna STATUS_OK se a operação ocorreu com sucesso
5      *         e retorna STATUS_FAIL se ocorrer um erro
6      */
7
8      public int saquePila(String nome){
9
10         try { // tratamento de exception
11             if (this.saldo > 0){
12                 return STATUS_FAIL;
13             }
14         }
15     }

```



```

11         }
12         double pilaSaque = Double.parseDouble(JOptionPane
            .showInputDialog(null, "Informe a quantidade em
                Reais (R$) a ser sacada na conta do "+this.
                nome+" "+this.sobrenome));
13         if (this.saldo >= pilaSaque){//verifica se tem
            saldo
14             this.saldo -= pilaSaque;
15             return STATUS_OK;
16         }
17         else{
18             JOptionPane.showMessageDialog(null, "O valor
                informado para o saque é maior que seu
                saldo.\nSeu saldo é de: R$ "+this.saldo, "
                Saldo Insuficiente", JOptionPane.
                ERROR_MESSAGE, icon);
19             return STATUS_FAIL;
20         }
21     }
22     catch (NumberFormatException e){
23         JOptionPane.showMessageDialog(null, "Entre com
            valor válido, do tipo numeral.\n ERRO: " + e.
            getMessage() , "ERRO", JOptionPane.
            ERROR_MESSAGE);
24         return STATUS_FAIL;
25     }
26 }

```

**Listing 3.** *saquePila()*

É solicitado que crie um *loop*, para que o usuário escolha quando deseja sair da operação, portanto o código para tal é o código 4 e a figura 4.

```

1         while(true){ //InterruptedException
2
3             String opcao = JOptionPane.showInputDialog(null, "Opção 1
                - Consulta saldo\n Opção 2 - Realizar um deposito\n
                Opção 3 - Realizar um saque\n Opção 4 - Sair \n", 4); //
                deixa a opçã4 4 como deful
4             //conversão de String para int
5             int control = 0;
6             if ("opcao" == null){
7
8                 control = 0;

```

```

9         }
10        else {
11            control = Integer.parseInt(opcao);
12        }
13
14        //int control = Integer.parseInt(opcao);
15
16        if ("opcao" == null) { //caso o usuario cancele a opção
17            JOptionPane.showMessageDialog(null, "Você cancelou a
18                operação");
19            break;
20        }
21
22        if (control == 0) {
23            JOptionPane.showMessageDialog(null, "Você cancelou a
24                operação");
25            break;
26        }
27
28        if (control == 4) { //para sair da operação
29            JOptionPane.showMessageDialog(null, "Volte sempre " +
30                cliente1.nome + " " + cliente1.sobrenome + "\nTenha um
31                bom dia!", "LOGOUT", JOptionPane.
32                INFORMATION_MESSAGE, icon); //add custom icon
33            break;
34        }
35
36        switch (control) {
37            case 0:
38                JOptionPane.showMessageDialog(null, "Você
39                    cancelou a operação");
40                break;
41
42            case 1: //consulta de saldo        consultaPilas(nome)
43                JOptionPane.showMessageDialog(null, "Consulta
44                    saldo.", cliente1.nome + " " + cliente1.sobrenome,
45                    JOptionPane.INFORMATION_MESSAGE, icon); //add
46                    custom icon
47                cliente1.consultaPila(cliente1.nome);
48                break;

```

```

41         case 2://realiza deposito
42             JOptionPane.showMessageDialog(null,"Realizar um
                deposito.", clientel.nome+" "+clientel.
                sobrenome, JOptionPane.INFORMATION_MESSAGE,
                icon);
43             int verifica = clientel.depositaPila(clientel.
                nome);
44             if (verifica == 1){
45                 JOptionPane.showMessageDialog(null,"Depósito
                    realizado com sucesso.", clientel.nome+" "
                    +clientel.sobrenome, JOptionPane.
                    INFORMATION_MESSAGE, icon);
46             }else{
47                 JOptionPane.showMessageDialog(null,"Falha no
                    deposito.\n Tente novamente.", clientel.
                    nome+" "+clientel.sobrenome, JOptionPane.
                    ERROR_MESSAGE, icon);
48             }
49             break;
50
51         case 3:// realiza saque
52             JOptionPane.showMessageDialog(null,"Realiza um
                saque.", clientel.nome+" "+clientel.sobrenome,
                JOptionPane.INFORMATION_MESSAGE, icon);
53             verifica = clientel.saquePila(clientel.nome);
54
55             if (verifica == 1){
56                 JOptionPane.showMessageDialog(null,"Saque
                    realizado com sucesso.", clientel.nome+" "
                    +clientel.sobrenome, JOptionPane.
                    INFORMATION_MESSAGE, icon);
57             }else{
58                 JOptionPane.showMessageDialog(null,"Falha no
                    saque.\n Tente novamente.", clientel.nome+
                    " "+clientel.sobrenome, JOptionPane.
                    ERROR_MESSAGE, icon);
59             }
60             break;
61
62         default:
63             JOptionPane.showMessageDialog(null,"Opção

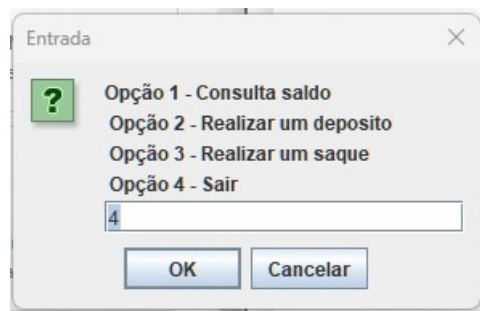
```

```

        inválida.", cliente1.nome+" "+cliente1.
        sobrenome, JOptionPane.INFORMATION_MESSAGE,
        icon);
64         break;
65     }
66 }

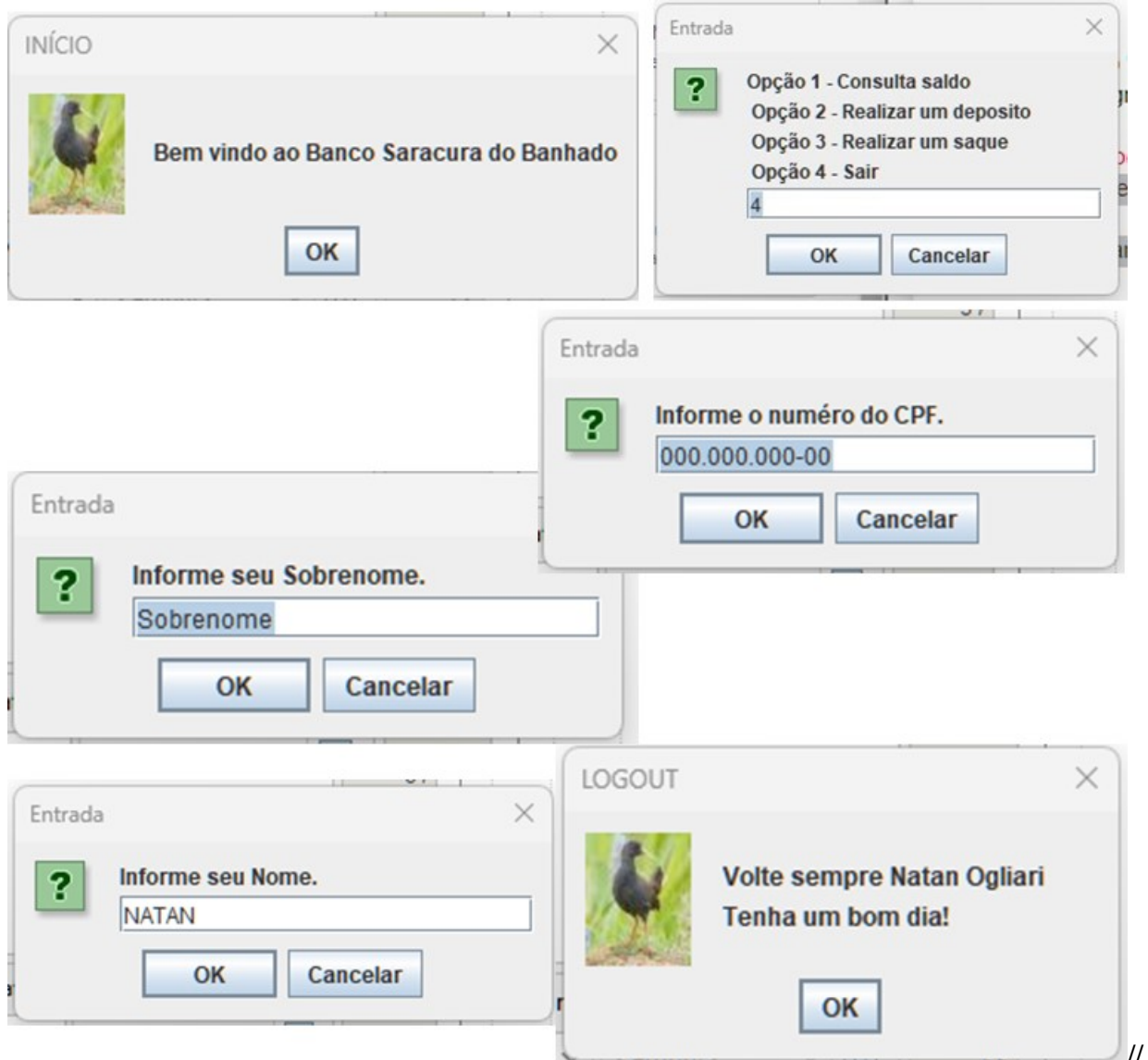
```

**Listing 4.** *loop*

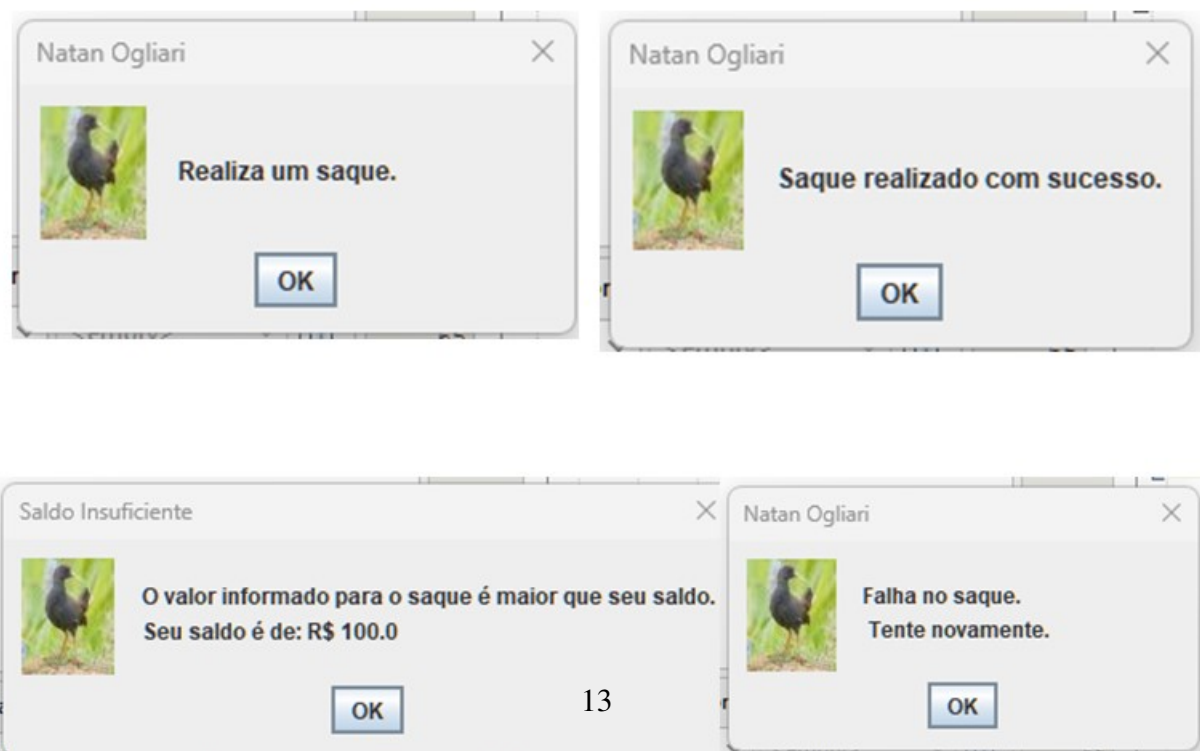


**Figura 4.** *Interface gráfica do loop, O autor(2023)*

Foi confeccionado uma interface gráfica para o usuário, conforme demonstra as figuras 5



(a) Menu total.



Para este projeto foi utilizado os comentários no formato do *javadoc*, e conforme a figura ??, demonstra a documentação deste projeto gerada automaticamente pelo NetBeans.

**Constructor Detail**

cliente

public cliente()

**Method Detail**

depositaPila

public int depositaPila(java.lang.String nome)

Realiza um deposito em uma conta

Parameters:

nome - Informa o nome da conta a ser depositado

Returns:

retorna STATUS\_OK se a operação ocorreu com sucesso e retorna STATUS\_FAIL se ocorrer um erro

saquePila

public int saquePila(java.lang.String nome)

Realiza o saque de uma conta

Parameters:

nome - Informa o nome da conta a ser realizado o saque

Returns:

retorna STATUS\_OK se a operação ocorreu com sucesso e retorna STATUS\_FAIL se ocorrer um erro

consultaPila

public void consultaPila(java.lang.String nome)

Realiza a consulta do saldo de uma conta

Parameters:

nome - Passa o nome da conta a ser consultada e informa o saldo ao cliente.

**Figura 6.** Documentação do projeto pelo *javadoc*, O autor (2023)

## 4 Conclusões

Esta aula prática teve como fim a aplicação de conhecimentos adquiridos em sala de aula virtuais da disciplina de linguagem orientada a objetos do curso bacharel em engenharia

de software da faculdade UNOPAR, o código completo<sup>2</sup> pode ser visto nos anexos 5 deste documento.

Com a implementação desta aula prática ficou evidente que é importante o tratamento de erros, por mais que nesta utilizei apenas uma instância e a complexidade fica evidente, de forma generalizada a importância do entendimento da documentação, nos tratamentos de erros observa que nos escopos *try*, deve-se utilizar apenas os códigos passíveis de erros, pois esta parte demanda um consumo de memória.

---

<sup>2</sup>código completo

## Referências

FREECODECAMP. **Programação orientada a objetos e programação estruturada**. 2023. Acessado em: 06 nov. 2023. Disponível em: <<https://www.freecodecamp.org/portuguese/news/os-quatro-pilares-da-programacao-orientada-a-objetos-com-javascript/>>.

STACKIFY. **Application Logging e Exception Handling**. 2023. Acessado em: 09 set. 2023. Disponível em: <<https://stackify.com/java-logging-best-practices/>>.

## 5 Anexos

```
1 package gerenciabanco;
2
3 import java.util.*;
4 //import javax.swing.JOptionPane;
5 import javax.swing.*;
6 import java.awt.*;
7 import java.lang.Integer;
8 import java.lang.Exception;
9 import java.lang.Error;
10 import java.lang.reflect.Method;
11 /**
12  * @author Natan Ogliari
13  * @version 0.1
14  */
15 public class GerenciaBanco {
16
17     public static class cliente{
18         String nome;
19         String sobrenome;
20         String cpf;
21         double saldo;
22
23         private static final int STATUS_OK = 1;///
```



```

27
28     /**
29     * Realiza um deposito em uma conta
30     * @param nome Informa o nome da conta a ser depositado
31     * @return retorna STATUS_OK se a operação ocorreu com sucesso
32     *         e retorna STATUS_FAIL se ocorrer um erro
33     */
34     public int depositaPila(String nome){
35
36         try { // verifica se a entrada e do tipo numeral
37             double pilaDeposito = Double.parseDouble(JOptionPane.
38                 showInputDialog(null, "Informe a quantidade em Reais
39                 (R$) a ser depositado na conta do "+this.nome+" "+
40                 this.sobrenome));
41             this.saldo += pilaDeposito;
42             //JOptionPane.showMessageDialog(null, "Seu Saldo é R$
43             :"+this.consultaPilas(this.nome)+" Reais", this.
44             nome, JOptionPane.INFORMATION_MESSAGE, icon);
45             return STATUS_OK;
46         }
47
48         catch (NumberFormatException e) { // imprime o erro na
49             tela e informa o que foi digitado.
50             JOptionPane.showMessageDialog(null, "Entre com valor
51             válido, do tipo numeral.\n Use (.) ponto em vez de
52             (,) virgula\n ERRO: " + e.getMessage() , "ERRO",
53             JOptionPane.ERROR_MESSAGE);
54             return STATUS_FAIL;
55         }
56     }
57
58     /**
59     * Realiza o saque de uma conta
60     * @param nome Informa o nome da conta a ser realizado o saque
61     * @return retorna STATUS_OK se a operação ocorreu com sucesso
62     *         e retorna STATUS_FAIL se ocorrer um erro
63     */
64     public int saquePila(String nome){
65
66         try { // tratamento de exception
67             if (this.saldo > 0){

```

```

57         return STATUS_FAIL;
58     }
59     double pilaSaque = Double.parseDouble(JOptionPane
        .showInputDialog(null, "Informe a quantidade em
        Reais (R$) a ser sacada na conta do "+this.
        nome+" "+this.sobrenome));
60     if (this.saldo >= pilaSaque){//verifica se tem
        saldo
61         this.saldo -= pilaSaque;
62         return STATUS_OK;
63     }
64     else{
65         JOptionPane.showMessageDialog(null, "O valor
        informado para o saque é maior que seu
        saldo.\nSeu saldo é de: R$ "+this.saldo, "
        Saldo Insuficiente", JOptionPane.
        ERROR_MESSAGE, icon);
66         return STATUS_FAIL;
67     }
68 }
69 catch (NumberFormatException e){
70     JOptionPane.showMessageDialog(null, "Entre com
        valor válido, do tipo numeral.\n ERRO: " + e.
        getMessage() , "ERRO", JOptionPane.
        ERROR_MESSAGE);
71     return STATUS_FAIL;
72 }
73 }
74
75 /**
76  * Realiza a consulta do saldo de uma conta
77  * @param nome Passa o nome da conta a ser consultada e
        informa o saldo ao cliente.
78  */
79 public void consultaPila(String nome){
80     JOptionPane.showMessageDialog(null, "Seu saldo atual é de:
        R$" +this.saldo, this.nome+" "+this.sobrenome,
        JOptionPane.INFORMATION_MESSAGE, icon);
81 }
82 }
83

```

```

84     public static void main(String[] args) {
85         /**Importa o logo do banco. */
86         ImageIcon icon = new ImageIcon("C:\\Users\\AULA-1\\Documents
            \\(Engenharia de Software)\\Fase 3\\Linguagem Orientada a
            Objetos\\gerenciaBanco\\gerenciaBanco\\src\\gerenciabanco
            \\saracura.jpg");
87
88         /** Instância a classe */
89         cliente clientel = new cliente();
90         JOptionPane.showMessageDialog(null, "Bem vindo ao Banco
            Saracura do Banhado\\n", "INÍCIO", JOptionPane.
            INFORMATION_MESSAGE, icon); //add custom icon
91         try { /** customer's name and checks for possible errors */
92             clientel.nome = JOptionPane.showInputDialog(null, "
                Informe seu Nome.", "Nome");
93             if (clientel.nome == null) { /** caso o usuário cancele a
                opção */
94                 JOptionPane.showMessageDialog(null, "Você cancelou a
                    operação");
95             }
96         }
97         catch (NullPointerException e) {
98             JOptionPane.showMessageDialog(null, "Entre com um Nome
                válido.", "Erro" , JOptionPane.ERROR_MESSAGE);
99         }
100
101         try { /** Solicita o sobrenome do cliente e verifica possíveis
            erros */
102             clientel.sobrenome = JOptionPane.showInputDialog(null, "
                Informe seu Sobrenome.", "Sobrenome");
103         }
104         catch (NullPointerException e) {
105             JOptionPane.showMessageDialog(null, "Entre com um
                sobrenome válido.", "Erro" , JOptionPane.ERROR_MESSAGE)
                ;
106         }
107
108         try { /** Solicita o cpf do cliente e verifica possíveis erros
            */
109             clientel.cpf = JOptionPane.showInputDialog(null, "Informe
                o número do CPF.", "000.000.000-00");

```

```

110     }
111     catch (NullPointerException e){
112         JOptionPane.showMessageDialog(null, "Entre com um CPF
113             válido.", "Erro" , JOptionPane.ERROR_MESSAGE);
114     }
115     //cliente1.saldo = 445;//remover
116     while(true){ //InterruptedException
117
118         String opcao = JOptionPane.showInputDialog(null, "Opção 1
119             - Consulta saldo\n Opção 2 - Realizar um depósito\n
120             Opção 3 - Realizar um saque\n Opção 4 - Sair \n", 4); //
121             deixa a opção 4 como deful
122         //conversão de String para int
123         int control = 0;
124         if ("opcao" == null){
125
126             control = 0;
127         }
128         else {
129             control = Integer.parseInt(opcao);
130         }
131
132         //int control = Integer.parseInt(opcao);
133
134         if ("opcao" == null){ //caso o usuario cancele a opção
135             JOptionPane.showMessageDialog(null, "Você cancelou a
136                 operação");
137             break;
138         }
139
140         if (control == 0){
141             JOptionPane.showMessageDialog(null, "Você cancelou a
142                 operação");
143             break;
144         }
145
146         if (control == 4){ //para sair da operação
147             JOptionPane.showMessageDialog(null, "Volte sempre "+
148                 cliente1.nome+" "+cliente1.sobrenome+"\nTenha um
149                 bom dia!", "LOGOUT", JOptionPane.
150                 INFORMATION_MESSAGE, icon); //add custom icon

```

```

142         break;
143     }
144
145     switch (control){
146         case 0:
147             JOptionPane.showMessageDialog(null, "Você
148                 cancelou a operação");
149             break;
150
151         case 1://consulta de saldo     consultaPilas(nome)
152             JOptionPane.showMessageDialog(null,"Consulta
153                 saldo.", clientel.nome+" "+clientel.sobrenome,
154                 JOptionPane.INFORMATION_MESSAGE, icon);//add
155                 custon icon
156             clientel.consultaPila(clientel.nome);
157             break;
158
159         case 2://realiza deposito
160             JOptionPane.showMessageDialog(null,"Realizar um
161                 deposito.", clientel.nome+" "+clientel.
162                 sobrenome, JOptionPane.INFORMATION_MESSAGE,
163                 icon);
164             int verifica = clientel.depositaPila(clientel.
165                 nome);
166             if (verifica == 1){
167                 JOptionPane.showMessageDialog(null,"Depósito
168                     realizado com sucesso.", clientel.nome+" "
169                     +clientel.sobrenome, JOptionPane.
170                     INFORMATION_MESSAGE, icon);
171             }else{
172                 JOptionPane.showMessageDialog(null,"Falha no
173                     deposito.\n Tente novamente.", clientel.
174                     nome+" "+clientel.sobrenome, JOptionPane.
175                     ERROR_MESSAGE, icon);
176             }
177             break;
178
179         case 3:// realiza saque
180             JOptionPane.showMessageDialog(null,"Realiza um
181                 saque.", clientel.nome+" "+clientel.sobrenome,
182                 JOptionPane.INFORMATION_MESSAGE, icon);

```

```

167         verifica = clientel.saquePila(clientel.nome);
168
169         if (verifica == 1){
170             JOptionPane.showMessageDialog(null, "Saque
                realizado com sucesso.", clientel.nome+ " "
                +clientel.sobrenome, JOptionPane.
                INFORMATION_MESSAGE, icon);
171         }else{
172             JOptionPane.showMessageDialog(null, "Falha no
                saque.\n Tente novamente.", clientel.nome+
                " "+clientel.sobrenome, JOptionPane.
                ERROR_MESSAGE, icon);
173         }
174         break;
175
176     default:
177         JOptionPane.showMessageDialog(null, "Opção
                inválida.", clientel.nome+ " "+clientel.
                sobrenome, JOptionPane.INFORMATION_MESSAGE,
                icon);
178         break;
179     }
180 }
181 }
182 }

```

**Listing 5.** *gerenciaBanco.java*