Universidade Federal do Espírito Santo

Uarlley do Nascimento Amorim - Matricula: 2018205346
Breno Souza do Nascimento - Matricula: 2018205338
22 de março de 2021

Linguagens de Programação

Professor : Francisco de Assis Souza dos Santos

Ciência da Computação

Resumo

Neste trabalho, foram realizadas diversas modificações no software disponibilizado pelo professor, modificações que visam modularizar e ampliar a funcionalidade, legibilidade e confiabilidade do mesmo. Para realizar tais modificações foram utilizados, em maior parte, artifícios e ferramentas já conhecidas, não havendo necessidade de pesquisa ou consulta.

Foram obtidos resultados incríveis, sendo os mais importantes a modularização da classe AudioValores, que agora, não necessita da criação de uma nova função sempre que for necessário a ampliação dos números cardinais, a melhoria nas funções de conversão de String para Double para evitar problemas de trucamento, como era o caso, e, por fim, a extensão do limite de valores do software, que antes era limitado de 0 a 999, e agora é de 0 a 999 quadrilhões.

Sumário

1 Lista de Figuras **4**
1.1 Interface . . . 4
1.2 Reprodução de valores . . . 4
1.3 Reprodução de Músicas . . . 5

2 Introdução **5**

3 Resultados **5**
3.1 Funções de Conversão . . . 6
3.1.1 Função converteCentavo . . . 6
3.1.2 Código Fonte: . . . 6
3.1.3 Função converteReal . . . 6
3.1.4 Código Fonte . . . 6
3.1.5 Função converteMoeda . . . 7
3.1.6 Código Fonte: . . . 7
3.2 Utilização da duração do áudio . . . 7
3.2.1 Código Fonte: . . . 7
3.3 Modificações na função de reprodução dos valores numéricos . . . 8
3.3.1 Codigo Fonte: . . . 8
3.3.2 Criação de novos enumeradores . . . 10
3.4 Modificações na reprodução de músicas . . . 10
3.4.1 Classe: ExecutaSom . . . 10
3.4.2 Classe: Apresentação . . . 10

4 Conclusões **11**

5 Referencias Bibliográficas **11**

6 Apêndices **11**
6.1 Apêndice 1: Classe Apresentação . . . 11
6.2 Apêndice 2: Classe AudioValores . . . 20
6.3 Apêndice 3: Classe ExecutaSom . . . 22
6.4 Apêndice 4: Classe VozEletronicaCartoesCredito . . . 23
6.5 Apêndice 5: Classe PainelImagemFundo . . . 23
6.6 Apêndice 6: Classe OutlineLabel . . . 24

1 Lista de Figuras

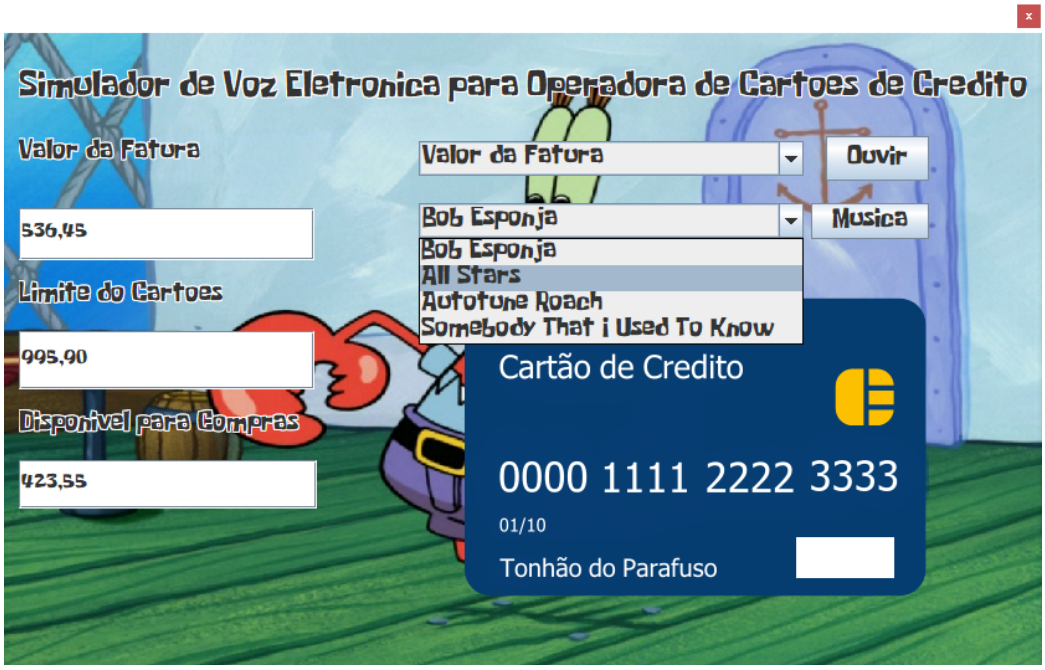
1.1 Interface

Figura(1): Interface principal do programa.

1.2 Reprodução de valores

Figura(2): Módulo de reprodução dos valores monetários presente na interface principal do programa.

1.3 Reprodução de Músicas



Figura(3): Módulo de reprodução de músicas presente na interface principal do programa.

2 Introdução

Desenvolver um software de boa qualidade não é tarefa fácil, pois, quando se pensa em criar um software, é de extrema importância saber definir qual LP é a mais adequada ao projeto, ter a capacidade de aprender uma LP facilmente, ter a capacidade de solucionar problemas, conhecer as ferramentas e funcionalidades de uma LP e, se necessário, saber desenvolver uma LP.

A disciplina de **Linguagens de Programação** tem como objetivo fornecer tal conhecimento ao desenvolvedor, tornando mais efetivo o processo de desenvolvimento de um software, garantindo a manutenibilidade, confiabilidade e eficiência do do mesmo.

O objetivo da atividade é implementar as práticas apresentadas em aula, práticas que garantem propriedades desejadas em um software.

Neste trabalho, diversos métodos e funções foram alteradas e implementadas, visando tornar o código mais organizado, modularizado e confiável, além de expandir a funcionalidade do software.

3 Resultados

Os resultados obtidos foram positivos, as funções modificadas e implementadas apresentaram mais eficiência, confiabilidade e expandiram as funcionalidades do software se compa-

rado com o software inicial.

As principais mudanças foram realizadas nas funções de conversão e funções de reprodução de áudio.

3.1 Funções de Conversão

Para maior confiabilidade decidimos utilizar a tabela ASCII, assim evitamos problemas de truncamentos que estavam presentes na versão inicial do projeto.

3.1.1 Função converteCentavo

No método de converteCentavo transformamos o primeiro caractere da string em uma dezena e o segundo caractere em uma unidade ao final retornamos a soma dos resultados obtidos.

3.1.2 Código Fonte:

```
0     long converteCentavo(String moeda){
1         return moeda.charAt(1) - 48 + (moeda.charAt(0) - 48)*10;
2     }
```

3.1.3 Função converteReal

No método converteReal percorremos a string da moeda e incrementamos em **result** a conversão do caractere na posição i para long de acordo com sua cardinalidade, ao final do laço obtemos o resultado da conversão total.

3.1.4 Código Fonte

```
0     long converteReal(String moeda){
1         long result = 0; //Variaval para armazenar o resultado
2         int size = moeda.length(),i,k = 0;//Tamanho da string, variavel utilizada para ocultamento
              de entidade em bloco aninhado.
3         i = size-1;
4         while(i >= 0){
5             if(moeda.charAt(i) != '.'){//Condicao para ignorar os '.' da string
6                 long mult = (long)pow(10,k);//Ocultamento da entidade mult.
7                 k++;
8                 result += (moeda.charAt(i)-48)*mult;
9             }
10            i--;
11        }
12        return result; //Retorno do resultado
13    }
```

3.1.5 Função converteMoeda

No método converteMoeda utilizamos o método split para separar a parte inteira e a parte dos centavos da string moeda. Assim, utilizamos a primeira parte da string separada para chamar a função de conversão para real e armazenar o valor da conversão em valor[0], de maneira semelhante é utilizado a segunda parte da string para fazer a conversão para centavos e armazenar o valor da conversão em valor[1]. Ao final é retornado um array de long, onde a primeira posição é referente a parte dos reais e a segunda posição é referente aos centavos.

3.1.6 Código Fonte:

```
0
1       long[] converteMoeda(String moeda){
2           long valor[] = new long[2]; //Vetor para armazenar o resultado
3           String[] parts = moeda.split(",");//Dividindo a string em reais e centavos
4           valor[0] = converteReal(parts[0]);//Armazendo os reais na primeira posicao
5           valor[1] = converteCentavo(parts[1]);//Armazenando os centavos na segunda posicao
6           return valor; //retorno do vetor
7       }
```

3.2 Utilização da duração do áudio

Para não definir manualmente a pausa entre a reprodução dos áudios, alteramos o retorno da função **executaSom** contida na classe **ExecutaSom**. A mudança garante que o retorno da função seja a duração do audio em milissegundos.

3.2.1 Código Fonte:

```
0
1       public long executaSom(String caminho, boolean modo_continuo, float decibeis) throws Exception
            {
2
3           //URL do som que no caso esta na pasta C:\VozEletronicaCartoesCredito\Sons
4           AudioInputStream audioInputStream = AudioSystem.getAudioInputStream(new File(caminho).
                getAbsoluteFile());
5           oClip = AudioSystem.getClip();
6           oClip.open(audioInputStream);
7           FloatControl gainControl = (FloatControl) oClip.getControl(FloatControl.Type.MASTER_GAIN);
8           gainControl.setValue(decibeis);
9
10          if (modo_continuo) /* Flag usado para verificar se ser  tocado o som de forma repetida ou
                uma  nica  vez  */
11              oClip.loop(Clip.LOOP_CONTINUOUSLY); // Toca continuamente
12          else
13              oClip.loop(0); // Toca apenas uma vez
14
15          return oClip.getMicrosecondLength()/1000;
16      }
```

3.3 Modificações na função de reprodução dos valores numéricos

No método de reprodução dos valores numéricos, foi notada a criação de uma função para cada número cardinal, e uma função para reprodução dos centavos. Nesse contexto, as função de reprodução foram modularizadas, resultando em apenas uma função que reproduz o áudio de qualquer valor, desde que tal valor não passe do limite estipulado (999 quadrilhões). Este limite foi definido pois, como o software trata de valores monetários, não faz sentido para o programa ultrapassar tal valor. Mas, se for necessário expandir o limite, basta manter os valores como Strings. Assim, valores com até 2^{16} dígitos poderão ser lidos.

Neste método, foi adicionado o parâmetro tipo, que indica qual cardinalidade será reproduzida (milhão, bilhão, trilhão...). Dessa forma, é passado uma centena que deve ser reproduzida, seguida da cardinalidade. O processo é repetido recursivamente, até que o valor seja 0. Portanto, dividindo o valor em centenas, basta reproduzir cada centena e sua respectiva cardinalidade.

3.3.1 Codigo Fonte:

```
0
1   public void audio_Centena(long valor, String Diretorio, String tipo) throws Exception{
2
3       int oqFalar;
4       long time = 0;
5       oqFalar = (calculaTamanho(valor)-1)/3-1;
6       while(oqFalar > 0){
7           long comp = (long)pow(10,(oqFalar+1)*3);
8           if(valor/comp == 1)
9               audio_Centena(valor/comp,Diretorio,"" + mil_quadrilhao.values()[oqFalar]);
10          else audio_Centena(valor/comp,Diretorio, "" + mil_quadrilhaoP.values()[oqFalar]);
11          valor = valor%comp;
12          oqFalar = (calculaTamanho(valor)-1)/3-1;
13      }
14
15      int i_centena, i_dezena, i_unidade, played = 0;
16      ExecutaSom play = new ExecutaSom();//Define objeto play da Classe ExecutaSom
17
18      if(valor == 0){
19              time = play.executaSom(Diretorio+"De.wav",false,0);/* Utiliza o enumerador centena
                    para concatenar o caminho de onde estao os arquivos de som */
20              Thread.sleep(time - 200);//Para estabelecer um intervalo entre os sons
21              time = play.executaSom(Diretorio+ tipo + ".wav",false,0);
22              Thread.sleep(time);
23      }
24
25      if(valor >= 1000){
26          if(valor/1000 == 1){
27              time = play.executaSom(Diretorio+"Mil.wav",false,0);
28              Thread.sleep(time - 200);
29              played++;
30          }
31          else audio_Centena(valor/1000, Diretorio, "Mil");
32          valor = valor%1000;
33          //played++;
```

```
34          }
35
36          i_centena= (int)valor/100;
37          i_dezena= (int)(valor%100)/10;
38          i_unidade= (int)valor%10;
39
40          if(i_centena!=0){
41              if(valor == 100){
42                  time = play.executaSom(Diretorio+"Cem.wav",false,0);/* Utiliza o enumerador
                        centena para concatenar o caminho de onde est o os arquivos de som */
43                  Thread.sleep(time - 200);//Para estabelecer um intervalo entre os sons
44              }
45              else{
46                  time = play.executaSom(Diretorio+centena.values()[i_centena-1]+".wav",false,0);
47                  Thread.sleep(time - 200);//Para estabelecer um intervalo entre os sons
48              }
49              played++;
50
51          }
52          if(valor%100 >= 10 && valor%100 <= 19){
53              if(played>0){
54                  time = play.executaSom(Diretorio+"ETeste.wav",false,-5.0f);
55                  Thread.sleep(time - 200);//Para estabelecer um intervalo para os possiveis
                        proximos sons
56              }
57              time = play.executaSom(Diretorio + dez_dezenove.values()[i_unidade]+".wav",false,0);
58              Thread.sleep(time - 200);//Para estabelecer um intervalo para os possiveis proximos
                    sons
59              played++;
60          }
61          else{
62              if(i_dezena!=0){
63                  if(played>0){
64                      time = play.executaSom(Diretorio+"ETeste.wav",false,-5.0f);
65                      Thread.sleep(time - 300);//Para estabelecer um intervalo para os possiveis
                            proximos sons
66                  }
67                  time = play.executaSom(Diretorio+dezena.values()[i_dezena-1]+".wav",false,0);
68                  Thread.sleep(time - 300);//Para estabelecer um intervalo entre os sons
69                  played++;
70              }
71              if(i_unidade!=0){
72                  if(played>0){
73                      time = play.executaSom(Diretorio+"ETeste.wav",false,-5.0f);
74                      Thread.sleep(time - 200);//Para estabelecer um intervalo para os possiveis
                            proximos sons
75                  }
76                  time = play.executaSom(Diretorio+unidade.values()[i_unidade-1]+".wav",false,0);
77                  Thread.sleep(time - 200);//Para estabelecer um intervalo entre os sons
78                  played++;
79              }
80          }
81          if(played>0){
82              time = play.executaSom(Diretorio+ tipo + ".wav",false,0);
83              Thread.sleep(time);//Para estabelecer um intervalo entre os possiveis proximos sons
84          }
85      }
86  }
```

3.3.2 Criação de novos enumeradores

Além disso, foi criado dois enumeradores para as unidades cardinais e o plural das mesmas para auxiliar na função de reprodução.

```
87      public enum mil_quadrilhao{mil, milhao, bilhao, trilhao, quadrilhao};
88      public enum mil_quadrilhaoP{mil, milhoes, bilhoes, trilhoes, quadrilhoes};
```

3.4 Modificações na reprodução de músicas

3.4.1 Classe: ExecutaSom

Para contornamos o problema de uma música sobrescrevendo outra, criamos um método para terminar um áudio que está tocando na classe ExecutaSom. Assim, esse método apenas verifica se existe um áudio tocando no momento, caso positivo é chamado o método Stop() para terminar a sua execução.

```
89
90      public void StopAudio() {
91          if (oClip.isRunning()) oClip.stop();
92      }
```

3.4.2 Classe: Apresentação

O método abaixo é responsável por selecionar o nome da música através do índice selecionado pelo usuário. Primeiramente é realizado a verificação se play é diferente de null, caso verdadeiro é chamado o método de parar áudio para garantir que não tenha nenhum áudio tocando, em seguida é selecionado o nome da música e chamado o método para executá-la.

```
94
95      ExecutaSom play = null;
96
97      private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
98          if (play != null) play.StopAudio();
99          try {
100             String NomeMusica = ""; //Nome da M sica
101             switch (options1.getSelectedIndex()) {
102                 case 0:
103                     NomeMusica="Spongebob";
104                     break;
105                 case 1:
106                     NomeMusica = "allStar";
107                     break;
108                 case 2:
109                     NomeMusica = "autotuneRoach";
110                     break;
111                 case 3:
112                     NomeMusica = "SomebodyThatIUsedToKnow";
113                     break;
114                 default:
115                     break;
116             }
```

```
117             play = new ExecutaSom();
118             play.executaSom(PathAudios+NomeMusica+".wav",true,-10.0f);
119         } catch (Exception ex) {
120             Logger.getLogger(Apresentacao.class.getName()).log(Level.SEVERE, null, ex);
121         }
122     }
```

4 Conclusões

Neste trabalho implementamos uma nova abordagem para conversão dos valores monetários, assim conseguimos uma maior confiabilidade na conversão. Uma vez melhorado o algoritmo de conversão foi realizado uma modularização na classe que gerencia a reprodução dos valores, para conseguir aumentar facilmente o limite do valor reproduzido, melhorando assim, a reusabilidade dessa classe. Uma das dificuldades encontradas foi a reprodução do áudio, e a definição de pausas durante essa reprodução. Conseguimos superar essa dificuldade modificando a classe ExecutaSom, implementamos um método para parar um áudio que esteja tocando e retornamos a duração do áudio no método executaSom(). Em geral no processo de desenvolvimento do trabalho conseguimos compreender os aspectos técnicos através das soluções criadas para resolver os problemas.

5 Referencias Bibliográficas

Consultas nos materiais da disciplina; Conteúdos aprendidos na disciplina de programação orientada à objetos;

6 Apêndices

6.1 Apêndice 1: Classe Apresentação

```
0
1  package vozeletronicacartoescredito;
2
3  import java.awt.Font;
4  import static java.lang.Math.ceil;
5  import static java.lang.Math.pow;
6  import static java.lang.Math.round;
7  import java.text.DecimalFormat;
8  import java.text.ParseException;
9  import java.util.logging.Level;
10 import java.util.logging.Logger;
11 import javax.swing.ImageIcon;
12 import javax.swing.JLabel;
13 import javax.swing.JOptionPane;
14
15
16 public class Apresentacao extends javax.swing.JFrame {
```

Quadro 1 - Propriedades, amarrações, definições e declarações em Linguagens de Programação.

Elementos	Termos (Qnt. de exemplos)	Linhas de código
Propriedades	Legibilidade (3X)	Apêndice 1: Ex. (1): 204 a 205 Ex. (2): 210 a 222 Ex. (3): 241 a 245
	Confiabilidade (2X)	Apêndice 1: Ex. (1): 288 - 316 Ex. (2): 323 - 343
	Facilidade de Aprendizado (2X)	Apêndice 1: Ex. (1): 205 Apêndice 3: Ex. (2): 33
	Modificabilidade (1X)	Apêndice 1: Ex. (1): 18, 22
	Reusabilidade (2X)	Apêndice 3: Ex. (1): 15 a 30 Apêndice 1 Ex. (2): 228 a 233
Amarrações	Identificadores (5X)	Apêndice 1: Ex. (1): 210 Ex. (2): 211 Ex. (3): 212 Ex. (4): 256 Apêndice 2: Ex. (5) 26
	Escopo estático - Ocultamento de entidades em blocos aninhados (1X)	Recurso não suportado pelo Java
	Escopo estático - Blocos Aninhados (3X)	Apêndice 1: Ex. (1): 256 Ex. (2): 100 Apêndice 2: Ex.(3): 30
Definições e Declarações	Definições (3X)	Apêndice 1: Ex. (1): 19 Ex. (2): 292 Apêndice 2: Ex(3): 14
	Declarações (3X)	Apêndice 2: Ex. (1): 26 Apêndice 2: Ex. (2): 291 Ex. (3): 252
	Definições de Tipos (2X)	Apêndice 2: Ex. (1): 7 Ex. (2): 8
	Declaração/Definição com inicialização dinâmica (1X)	Apêndice 1: Ex - 212

```
17
18      final String PathAudios;
19      ExecutaSom play = null;
20
21      public Apresentacao() {
22          this.PathAudios = "C:\\VozEletronicaCartoesCredito\\Sons\\";
23          initComponents();
24      }
25
26      @SuppressWarnings("unchecked")
27      // <editor-fold defaultstate="collapsed" desc="Generated Code">
28      private void initComponents() {
29
30          painelImagemFundo1 = new vozeletronicacartoescredito.PainelImagemFundo();
31          outlineLabel2 = new vozeletronicacartoescredito.OutlineLabel();
32          Imagem = new javax.swing.JLabel();
33          outlineLabel4 = new vozeletronicacartoescredito.OutlineLabel();
34          ValorFatura = new javax.swing.JFormattedTextField();
35          outlineLabel1 = new vozeletronicacartoescredito.OutlineLabel();
36          Limite = new javax.swing.JFormattedTextField();
37          outlineLabel5 = new vozeletronicacartoescredito.OutlineLabel();
38          Disponivel = new javax.swing.JFormattedTextField();
39          options = new javax.swing.JComboBox<>();
40          jButton1 = new javax.swing.JButton();
41          jButton2 = new javax.swing.JButton();
42          options1 = new javax.swing.JComboBox<>();
43
44          setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
45          setFont(new java.awt.Font("Agency FB", 0, 18)); // NOI18N
46          setType(java.awt.Window.Type.UTILITY);
47
48          painelImagemFundo1.setBackground(new java.awt.Color(204, 255, 0));
49          painelImagemFundo1.setImg(new ImageIcon("src/imagens/wallpaper.png"));
50
51          outlineLabel2.setText("Simulador de Voz Eletronica para Operadora de Cartoes de Credito");
52          outlineLabel2.setFont(new java.awt.Font("Spongeboy Me Bob", 0, 24)); // NOI18N
53
54          Imagem.setIcon(new javax.swing.ImageIcon("C:\\VozEletronicaCartoesCredito\\cart o -de-
                cr dito -azul.png")); // NOI18N
55          Imagem.setText("Imagem");
56
57          outlineLabel4.setText("Valor da Fatura");
58          outlineLabel4.setFont(new java.awt.Font("Spongeboy Me Bob", 0, 18)); // NOI18N
59
60          ValorFatura.setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(new javax.
                swing.text.NumberFormatter(new java.text.DecimalFormat("#,###.00"))));
61          ValorFatura.setText("536,45");
62          ValorFatura.setFont(new java.awt.Font("Spongeboy Me Bob", 0, 14)); // NOI18N
63
64          outlineLabel1.setText("Limite do Cartoes");
65          outlineLabel1.setFont(new java.awt.Font("Spongeboy Me Bob", 0, 18)); // NOI18N
66
67          Limite.setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(new javax.swing.
                text.NumberFormatter(new java.text.DecimalFormat("#,###.00"))));
68          Limite.setText("995,90");
69          Limite.setFont(new java.awt.Font("Spongeboy Me Bob", 0, 14)); // NOI18N
70
71          outlineLabel5.setText("Disponivel para Compras");
72          outlineLabel5.setFont(new java.awt.Font("Spongeboy Me Bob", 0, 18)); // NOI18N
73
```

```
74          Disponivel.setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(new javax.
                swing.text.NumberFormatter(new java.text.DecimalFormat("#,###.00"))));
75          Disponivel.setText("423,55");
76          Disponivel.setFont(new java.awt.Font("Spongeboy Me Bob", 0, 14)); // NOI18N
77          Disponivel.addActionListener(new java.awt.event.ActionListener() {
78              public void actionPerformed(java.awt.event.ActionEvent evt) {
79                  DisponivelActionPerformed(evt);
80              }
81          });
82
83          options.setFont(new java.awt.Font("Spongeboy Me Bob", 0, 18)); // NOI18N
84          options.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Valor da Fatura",
                "Limite do Cartao", "Disponivel para Compras" }));
85          options.addActionListener(new java.awt.event.ActionListener() {
86              public void actionPerformed(java.awt.event.ActionEvent evt) {
87                  optionsActionPerformed(evt);
88              }
89          });
90
91          jButton1.setFont(new java.awt.Font("Spongeboy Me Bob", 0, 18)); // NOI18N
92          jButton1.setText("Ouvir");
93          jButton1.addActionListener(new java.awt.event.ActionListener() {
94              public void actionPerformed(java.awt.event.ActionEvent evt) {
95                  jButton1ActionPerformed(evt);
96              }
97          });
98
99          jButton2.setFont(new java.awt.Font("Spongeboy Me Bob", 0, 18)); // NOI18N
100         jButton2.setText("Musica");
101         jButton2.addActionListener(new java.awt.event.ActionListener() {
102             public void actionPerformed(java.awt.event.ActionEvent evt) {
103                 jButton2ActionPerformed(evt);
104             }
105         });
106
107         options1.setFont(new java.awt.Font("Spongeboy Me Bob", 0, 18)); // NOI18N
108         options1.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Bob Esponja", "
                All Stars", "Autotune Roach", "Somebody That I Used To Know" }));
109         options1.addActionListener(new java.awt.event.ActionListener() {
110             public void actionPerformed(java.awt.event.ActionEvent evt) {
111                 options1ActionPerformed(evt);
112             }
113         });
114
115         javax.swing.GroupLayout painelImagemFundo1Layout = new javax.swing.GroupLayout(
                painelImagemFundo1);
116         painelImagemFundo1.setLayout(painelImagemFundo1Layout);
117         painelImagemFundo1Layout.setHorizontalGroup(
118             painelImagemFundo1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
                    )
119             .addGroup(painelImagemFundo1Layout.createSequentialGroup()
120                 .addContainerGap()
121                 .addGroup(painelImagemFundo1Layout.createParallelGroup(javax.swing.GroupLayout.
                        Alignment.LEADING)
122                     .addGroup(painelImagemFundo1Layout.createSequentialGroup()
123                         .addComponent(outlineLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax
                                .swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                                PREFERRED_SIZE)
124                         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
125                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, painelImagemFundo1Layout
                            .createSequentialGroup()
```

```
126                         .addGroup(painelImagemFundo1Layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.LEADING)
127                             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                                    painelImagemFundo1Layout.createParallelGroup(javax.swing.
                                    GroupLayout.Alignment.LEADING)
128                                 .addComponent(outlineLabel1, javax.swing.GroupLayout.
                                        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.
                                        swing.GroupLayout.PREFERRED_SIZE)
129                                 .addComponent(outlineLabel5, javax.swing.GroupLayout.
                                        PREFERRED_SIZE, 249, javax.swing.GroupLayout.PREFERRED_SIZE)
130                                 .addGroup(painelImagemFundo1Layout.createParallelGroup(javax.swing
                                        .GroupLayout.Alignment.TRAILING, false)
131                                     .addComponent(Limite, javax.swing.GroupLayout.Alignment.
                                            LEADING)
132                                     .addComponent(ValorFatura, javax.swing.GroupLayout.Alignment.
                                            LEADING, javax.swing.GroupLayout.PREFERRED_SIZE, 238,
                                            javax.swing.GroupLayout.PREFERRED_SIZE))
133                                 .addComponent(Disponivel, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        240, javax.swing.GroupLayout.PREFERRED_SIZE))
134                             .addComponent(outlineLabel4, javax.swing.GroupLayout.PREFERRED_SIZE,
                                    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                                    PREFERRED_SIZE))
135                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 21,
                                Short.MAX_VALUE)
136                         .addGroup(painelImagemFundo1Layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.LEADING)
137                             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                                    painelImagemFundo1Layout.createSequentialGroup()
138                                 .addComponent(options, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        310, javax.swing.GroupLayout.PREFERRED_SIZE)
139                                 .addGap(18, 18, 18)
140                                 .addComponent(jButton1))
141                             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                                    painelImagemFundo1Layout.createSequentialGroup()
142                                 .addComponent(options1, javax.swing.GroupLayout.PREFERRED_SIZE,
                                        310, javax.swing.GroupLayout.PREFERRED_SIZE)
143                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                                        RELATED)
144                                 .addComponent(jButton2))
145                             .addComponent(Imagem, javax.swing.GroupLayout.Alignment.TRAILING,
                                    javax.swing.GroupLayout.PREFERRED_SIZE, 379, javax.swing.
                                    GroupLayout.PREFERRED_SIZE))
146                         .addGap(91, 91, 91))))
147         );
148         painelImagemFundo1Layout.setVerticalGroup(
149             painelImagemFundo1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
                    )
150             .addGroup(painelImagemFundo1Layout.createSequentialGroup()
151                 .addGap(23, 23, 23)
152                 .addComponent(outlineLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 45, javax.
                        swing.GroupLayout.PREFERRED_SIZE)
153                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
154                 .addGroup(painelImagemFundo1Layout.createParallelGroup(javax.swing.GroupLayout.
                        Alignment.LEADING)
155                     .addGroup(painelImagemFundo1Layout.createSequentialGroup()
156                         .addGap(9, 9, 9)
157                         .addGroup(painelImagemFundo1Layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.BASELINE)
158                             .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 36,
                                    javax.swing.GroupLayout.PREFERRED_SIZE)
159                             .addComponent(options))
```

```
160                         .addGap(18, 18, 18)
161                         .addGroup(painelImagemFundo1Layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.BASELINE)
162                             .addComponent(jButton2)
163                             .addComponent(options1))
164                         .addGap(42, 42, 42)
165                         .addComponent(Imagem)
166                         .addGap(56, 56, 56))
167                     .addGroup(painelImagemFundo1Layout.createSequentialGroup()
168                         .addComponent(outlineLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 45,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
169                         .addGap(22, 22, 22)
170                         .addComponent(ValorFatura, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
171                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
172                         .addComponent(outlineLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 45,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
173                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
174                         .addComponent(Limite, javax.swing.GroupLayout.PREFERRED_SIZE, 47, javax.
                                swing.GroupLayout.PREFERRED_SIZE)
175                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
176                         .addComponent(outlineLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 45,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
177                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
178                         .addComponent(Disponivel, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
179                         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
180         );
181 
182         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
183         getContentPane().setLayout(layout);
184         layout.setHorizontalGroup(
185             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
186             .addComponent(painelImagemFundo1, javax.swing.GroupLayout.Alignment.TRAILING, javax.
                    swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.
                    MAX_VALUE)
187         );
188         layout.setVerticalGroup(
189             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
190             .addComponent(painelImagemFundo1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.
                    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
191         );
192 
193         pack();
194         setLocationRelativeTo(null);
195     }// </editor-fold>
196 
197     public JLabel mensagensDeAviso(String mensagem){
198         JLabel label = new JLabel(mensagem);
199         label.setFont(new Font("Spongeboy Me Bob", 1, 13));
200         return label;
201     }
202 
203     //Fun   o que converte a parte fracion ria da string em long
204     long converteCentavo(String moeda){
205         return moeda.charAt(1) - 48 + (moeda.charAt(0) - 48)*10;
206     }
207 
208     ////////////////////////////////////////////////////////
209     //Fun   o que converte a real da string em long
210     long converteReal(String moeda){
```

```
211         long result = 0; //Variaval para armazenar o resultado
212         int size = moeda.length(),i,k = 0;//Tamanho da string, variavel usada para ocultamento de
                entidade em bloco aninhado
213         i = size - 1;
214         while (i >= 0) {
215             if(moeda.charAt(i) != '.'){//Condi  o para ignorar os '.' da string
216                 long mult = (long)pow(10,k); // Ocultamento da entidade size
217                 k++;
218                 result += (moeda.charAt(i)-48)*mult;
219             }
220             i--;
221         }
222         return result; //Retorno do resultado
223     }
224 
225     ////////////////////////////////////////////////////////
226     /*Fun  o que converte a string e moeda, criando um vetor
227     e armazenando os reais em uma parte, e os centavos em outra*/
228     long[] converteMoeda(String moeda){
229         long valor[] = new long[2]; //Vetor para armazenar o resultado
230         String[] parts = moeda.split(",");//Dividindo a string em reais e centavos
231         valor[0] = converteReal(parts[0]);//Armazendo os reais na primeira posicao
232         valor[1] = converteCentavo(parts[1]);//Armazenando os centavos na segunda posicao
233         return valor; //retorno do vetor
234     }
235 
236     ////////////////////////////////////////////////////////
237     int ObtemInteiro(double num){
238         return (int)(num);
239     }
240     ////////////////////////////////////////////////////////////////
241     int ObtemDecimais(double num){
242         float parte_decimal = (float)(num - ObtemInteiro(num));
243         float parte_centavos = parte_decimal*100-parte_decimal;
244         System.out.println(parte_centavos);
245         return (int)(ceil(parte_centavos));
246     }
247 
248     ////////////////////////////////////////////////////////////////
249     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
250         /* Local onde devem estar os arquivos de a dio .wav */
251 
252         long[] moeda;
253         /*Converte para long os reais e os centavos a
254         string contidade no campo selecionado*/
255         if(options.getSelectedIndex()==0) {
256             String text = ValorFatura.getText();
257             if(text.length() > 26){
258                 JOptionPane.showMessageDialog(null,mensagensDeAviso("Dados invalidos!."),"Dados
                        n o suportados",JOptionPane.ERROR_MESSAGE);
259                 ValorFatura.setText("");
260                 return;
261             }
262             System.out.print("Chamou");
263             moeda = converteMoeda(text);
264         }
265 
266         else if(options.getSelectedIndex() == 1) {
267             String text = Limite.getText();
268             if(text.length() > 26){
```

```
269                 JOptionPane.showMessageDialog(null,mensagensDeAviso("Dados invalidos!."),"Dados
                        n o suportados",JOptionPane.ERROR_MESSAGE);
270                 Limite.setText("");
271                 return;
272             }
273             moeda = converteMoeda(Limite.getText());
274 
275         }
276 
277         else {
278             String text = Disponivel.getText();
279             if(text.length() > 26){
280                 JOptionPane.showMessageDialog(null,mensagensDeAviso("Dados invalidos!."),"Dados
                        n o suportados",JOptionPane.ERROR_MESSAGE);
281                 Disponivel.setText("");
282                 return;
283             }
284             moeda = converteMoeda(Disponivel.getText());
285         }
286 
287 
288         try {
289             /* As cinco linhas a seguir tem o prop sito de obter:
290             a parte inteira, decimal e tamanho da vari vel valor do tipo double */
291             long time;
292             int played = 0;
293             ExecutaSom toca = new ExecutaSom();
294             AudioValores play = new AudioValores(); // Define o objeto da Classe AudioValores
295 
296             //Toca os audios se o valor em reais for maior que 0
297             if(moeda[0] > 0){
298                 //Condi  es de utiliza  o do plural
299                 if(moeda[0] == 1) play.audio_Centena(moeda[0],PathAudios,"Real");
300                 else play.audio_Centena(moeda[0],PathAudios,"Reais");
301                 played++;
302             }
303 
304             if (moeda[1]>0){
305                 //Condi  es de utiliza  o da conjun  o "e"
306                 if(played>0) {
307                     time = toca.executaSom(PathAudios + "ETeste.wav",false,-5.0f);
308                     Thread.sleep(time - 200);
309                 }
310                 //Condi  es de utiliza  o do plural
311                 if(moeda[1] == 1) play.audio_Centena(moeda[1],PathAudios,"Centavo");
312                 else play.audio_Centena(moeda[1],PathAudios,"Centavos");
313             }
314 
315 
316         } catch (Exception ex) {
317             Logger.getLogger(Apresentacao.class.getName()).log(Level.SEVERE, null, ex);
318         }
319     }
320 
321     private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
322         if (play != null) play.StopAudio();
323         try {
324             String NomeMusica = ""; //Nome da M sica
325             switch (options1.getSelectedIndex()) {
326                 case 0:
327                     NomeMusica="Spongebob";
```

```
                    break;
                case 1:
                    NomeMusica = "allStar";
                    break;
                case 2:
                    NomeMusica = "autotuneRoach";
                    break;
                case 3:
                    NomeMusica = "SomebodyThatIUsedToKnow";
                    break;
                default:
                    break;
            }
            play = new ExecutaSom();
            play.executaSom(PathAudios+NomeMusica+".wav",true,-10.0f);
        } catch (Exception ex) {
            Logger.getLogger(Apresentacao.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void optionsActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

    }

    private void DisponivelActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void options1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
            feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.
            html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.
                getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Apresentacao.class.getName()).log(java.util.logging
                .Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Apresentacao.class.getName()).log(java.util.logging
                .Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Apresentacao.class.getName()).log(java.util.logging
                .Level.SEVERE, null, ex);
```

```
383         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
384             java.util.logging.Logger.getLogger(Apresentacao.class.getName()).log(java.util.logging
                    .Level.SEVERE, null, ex);
385         }
386         //</editor-fold>
387 
388         /* Create and display the form */
389         java.awt.EventQueue.invokeLater(new Runnable() {
390             public void run() {
391                 new Apresentacao().setVisible(true);
392             }
393         });
394     }
395 
396     // Variables declaration - do not modify
397     private javax.swing.JFormattedTextField Disponivel;
398     private javax.swing.JLabel Imagem;
399     private javax.swing.JFormattedTextField Limite;
400     private javax.swing.JFormattedTextField ValorFatura;
401     private javax.swing.JButton jButton1;
402     private javax.swing.JButton jButton2;
403     private javax.swing.JComboBox<String> options;
404     private javax.swing.JComboBox<String> options1;
405     private vozeletronicacartoescredito.OutlineLabel outlineLabel1;
406     private vozeletronicacartoescredito.OutlineLabel outlineLabel2;
407     private vozeletronicacartoescredito.OutlineLabel outlineLabel4;
408     private vozeletronicacartoescredito.OutlineLabel outlineLabel5;
409     private vozeletronicacartoescredito.PainelImagemFundo painelImagemFundo1;
410     // End of variables declaration
411 }
```

6.2 Apêndice 2: Classe AudioValores

```
0  package vozeletronicacartoescredito;
1 
2  import static java.lang.Math.pow;
3 
4  public class AudioValores {
5      /*os elementos dos enumeradores devem ter o mesmo nome de cada arquivo de voz .wav
           correspondente */
6      public enum unidade {um, dois, tres, quatro, cinco, seis, sete, oito, nove};
7      public enum dez_dezenove {dez, onze, doze, treze, quatorze, quinze, dezeseis, dezesete,
           dezoito, dezenove};
8      public enum dezena {dez, vinte, trinta, quarenta, cinquenta, sessenta, setenta, oitenta,
           noventa};
9      public enum centena {cento, duzentos, trezentos, quatrocentos, quinhentos, seiscentos,
           setecentos, oitocentos, novecentos};
10     public enum mil_quadrilhao{mil, milhao, bilhao, trilhao, quadrilhao};
11     public enum mil_quadrilhaoP{mil, milhoes, bilhoes, trilhoes, quadrilhoes};
12 
13     public int calculaTamanho(long valor) {
14         long num = valor;
15         int tam=0;
16         while(num !=0) {
17            num = num/10;
18            tam++;
19         }
20         return tam;
21     }
```

```
22
23
24      public void audio_Centena(long valor, String Diretorio, String tipo) throws Exception{
25
26          int oqFalar;
27          long time = 0;
28          oqFalar = (calculaTamanho(valor)-1)/3-1;
29          while(oqFalar > 0){
30              long comp = (long)pow(10,(oqFalar+1)*3);
31              if(valor/comp == 1)
32                  audio_Centena(valor/comp,Diretorio,"" + mil_quadrilhao.values()[oqFalar]);
33              else audio_Centena(valor/comp,Diretorio, "" + mil_quadrilhaoP.values()[oqFalar]);
34              valor = valor%comp;
35              oqFalar = (calculaTamanho(valor)-1)/3-1;
36          }
37
38          int i_centena, i_dezena, i_unidade, played = 0;
39          ExecutaSom play = new ExecutaSom();//Define objeto play da Classe ExecutaSom
40
41          if(valor == 0){
42                  time = play.executaSom(Diretorio+"De.wav",false,0);/* Utiliza o enumerador centena
                        para concatenar o caminho de onde estao os arquivos de som */
43                  Thread.sleep(time - 200);//Para estabelecer um intervalo entre os sons
44                  time = play.executaSom(Diretorio+ tipo + ".wav",false,0);
45                  Thread.sleep(time);
46          }
47
48          if(valor >= 1000){
49              if(valor/1000 == 1){
50                  time = play.executaSom(Diretorio+"Mil.wav",false,0);
51                  Thread.sleep(time - 200);
52                  played++;
53              }
54              else audio_Centena(valor/1000, Diretorio, "Mil");
55              valor = valor%1000;
56              //played++;
57          }
58
59          i_centena= (int)valor/100;
60          i_dezena= (int)(valor%100)/10;
61          i_unidade= (int)valor%10;
62
63          /*Concatena o diretorio onde est o os arquivos de audio,
64          nome do arquivo localizado no enumerado e extens o .wav*/
65          if(i_centena!=0){
66              if(valor == 100){
67                  time = play.executaSom(Diretorio+"Cem.wav",false,0);/* Utiliza o enumerador
                        centena para concatenar o caminho de onde est o os arquivos de som */
68                  Thread.sleep(time - 200);//Para estabelecer um intervalo entre os sons
69              }
70              else{
71                  time = play.executaSom(Diretorio+centena.values()[i_centena-1]+".wav",false,0);/*
                        Utiliza o enumerador centena para concatenar o caminho de onde est o os
                        arquivos de som */
72                  Thread.sleep(time - 200);//Para estabelecer um intervalo entre os sons
73              }
74              played++;
75
76          }
77          if(valor%100 >= 10 && valor%100 <= 19){
78              if(played>0){
```

```
79                  time = play.executaSom(Diretorio+"ETeste.wav",false,-5.0f);
80                  Thread.sleep(time - 200);//Para estabelecer um intervalo para os possiveis
                        proximos sons
81              }
82              time = play.executaSom(Diretorio + dez_dezenove.values()[i_unidade]+".wav",false,0);
83              Thread.sleep(time - 200);//Para estabelecer um intervalo para os possiveis proximos
                    sons
84              played++;
85          }
86          else{
87              if(i_dezena!=0){
88                  if(played>0){
89                      time = play.executaSom(Diretorio+"ETeste.wav",false,-5.0f);
90                      Thread.sleep(time - 300);//Para estabelecer um intervalo para os possiveis
                            proximos sons
91                  }
92                  time = play.executaSom(Diretorio+dezena.values()[i_dezena-1]+".wav",false,0);
93                  Thread.sleep(time - 300);//Para estabelecer um intervalo entre os sons
94                  played++;
95              }
96              if(i_unidade!=0){
97                  if(played>0){
98                      time = play.executaSom(Diretorio+"ETeste.wav",false,-5.0f);
99                      Thread.sleep(time - 200);//Para estabelecer um intervalo para os possiveis
                            proximos sons
100                 }
101                 time = play.executaSom(Diretorio+unidade.values()[i_unidade-1]+".wav",false,0);
102                 Thread.sleep(time - 200);//Para estabelecer um intervalo entre os sons
103                 played++;
104             }
105         }
106         if(played>0){
107             time = play.executaSom(Diretorio+ tipo + ".wav",false,0);
108             Thread.sleep(time);//Para estabelecer um intervalo entre os possiveis proximos sons
109         }
110     }
111 }
```

6.3 Apêndice 3: Classe ExecutaSom

```
0
1  package vozeletronicacartoescredito;
2
3  import java.io.File;
4  /* Sons amostrados  */
5  import javax.sound.sampled.AudioInputStream; /* Amostragem de entrada de audio  */
6  import javax.sound.sampled.AudioSystem; /* Carrega os arquivos de audio para a memoria em formato
       legivel pelo sistema operacional*/
7  import javax.sound.sampled.Clip; /* Executa os sons no sistema operacional */
8  import javax.sound.sampled.FloatControl;
9
10
11 public class ExecutaSom {
12
13     private Clip oClip;
14
15     public long executaSom(String caminho, boolean modo_continuo, float decibeis) throws Exception
           {
16
```

```
17          //URL do som que no caso esta na pasta C:\VozEletronicaCartoesCredito\Sons
18          AudioInputStream audioInputStream = AudioSystem.getAudioInputStream(new File(caminho).
                getAbsoluteFile());
19          oClip = AudioSystem.getClip();
20          oClip.open(audioInputStream);
21          FloatControl gainControl = (FloatControl) oClip.getControl(FloatControl.Type.MASTER_GAIN);
22          gainControl.setValue(decibeis);
23
24          if (modo_continuo) /* Flag usado para verificar se ser  tocado o som de forma repetida ou
                uma  nica  vez  */
25              oClip.loop(Clip.LOOP_CONTINUOUSLY); // Toca continuamente
26          else
27              oClip.loop(0); // Toca apenas uma vez
28
29          return oClip.getMicrosecondLength()/1000;
30      }
31
32      public void StopAudio() {
33          if (oClip.isRunning()) oClip.stop();
34      }
35  }
```

6.4 Apêndice 4: Classe VozEletronicaCartoesCredito

```
0
1
2   package vozeletronicacartoescredito;
3
4   public class VozEletronicaCartoesCredito {
5
6       public static void main(String[] args) {
7           Apresentacao Tela = new Apresentacao();
8           Tela.setVisible(true);
9
10      }
11
12  }
```

6.5 Apêndice 5: Classe PainelImagemFundo

```
0
1
2   package vozeletronicacartoescredito;
3
4   import java.awt.Graphics;
5   import javax.swing.ImageIcon;
6
7   public class PainelImagemFundo extends javax.swing.JPanel{
8
9       private ImageIcon img;
10
11      public PainelImagemFundo(){
12          img = new ImageIcon();
13      }
14
15      @Override
```

```
16      public void paintComponent(Graphics g){
17          super.paintComponent(g);
18          g.drawImage(img.getImage(), 0, 0, this.getWidth(), this.getHeight(), this);
19
20      }
21
22      public void setImg(ImageIcon img){
23          this.img = img;
24      }
25
26      public ImageIcon getImg(){
27          return this.img;
28      }
29
30  }
```

6.6 Apêndice 6: Classe OutlineLabel

```
0
1   /*
2    * To change this license header, choose License Headers in Project Properties.
3    * To change this template file, choose Tools | Templates
4    * and open the template in the editor.
5    */
6   package vozeletronicacartoescredito;
7
8   import java.awt.Color;
9   import java.awt.Graphics;
10  import javax.swing.JLabel;
11
12  public class OutlineLabel extends JLabel {
13
14      private Color outlineColor = Color.WHITE;
15      private boolean isPaintingOutline = false;
16      private boolean forceTransparent = false;
17
18      public OutlineLabel() {
19          super();
20      }
21
22      public OutlineLabel(String text) {
23          super(text);
24      }
25
26      public OutlineLabel(String text, int horizontalAlignment) {
27          super(text, horizontalAlignment);
28      }
29
30      public Color getOutlineColor() {
31          return outlineColor;
32      }
33
34      public void setOutlineColor(Color outlineColor) {
35          this.outlineColor = outlineColor;
36          this.invalidate();
37      }
38
39      @Override
40      public Color getForeground() {
```

```
41          if ( isPaintingOutline ) {
42              return outlineColor;
43          } else {
44              return super.getForeground();
45          }
46      }
47
48      @Override
49      public boolean isOpaque() {
50          if ( forceTransparent ) {
51              return false;
52          } else {
53              return super.isOpaque();
54          }
55      }
56
57      public void paint(Graphics g) {
58
59          String text = getText();
60          if ( text == null || text.length() == 0 ) {
61              super.paint(g);
62              return;
63          }
64
65          // 1 2 3
66          // 8 9 4
67          // 7 6 5
68
69          if ( isOpaque() )
70              super.paint(g);
71
72          forceTransparent = true;
73          isPaintingOutline = true;
74          g.translate(-1, -1); super.paint(g); // 1
75          g.translate( 1,  0); super.paint(g); // 2
76          g.translate( 1,  0); super.paint(g); // 3
77          g.translate( 0,  1); super.paint(g); // 4
78          g.translate( 0,  1); super.paint(g); // 5
79          g.translate(-1,  0); super.paint(g); // 6
80          g.translate(-1,  0); super.paint(g); // 7
81          g.translate( 0, -1); super.paint(g); // 8
82          g.translate( 1,  0); // 9
83          isPaintingOutline = false;
84
85          super.paint(g);
86          forceTransparent = false;
87
88      }
89  }
```