

UNIVERSIDADE PAULISTA
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGIA – ICET
CURSO DE CIÊNCIA DA COMPUTAÇÃO

**DESENVOLVIMENTO DE UM JOGO COM UTILIZAÇÃO DE
INTERFACE GRÁFICA**

SÃO PAULO
2021

BRUNO GONZALEZ MASSONE – N582JF3
FELIPE DE OLIVEIRA PEREIRA MAURÍCIO – F2322A8
FELIPE NUNES DA SILVA – N4935E0

DESENVOLVIMENTO DE UM JOGO COM UTILIZAÇÃO DE INTERFACE GRÁFICA

Atividades Práticas Supervisionadas – APS -
DESENVOLVIMENTO DE UM JOGO COM
UTILIZAÇÃO DE INTERFACE GRÁFICA –
trabalho apresentado como exigência para a
avaliação do 3º semestre, do curso de **Ciência
da Computação** da Universidade Paulista sob
orientação de professores do semestre.

Orientador: Carlos Baltazar
(Linguagem de Programação Orientada a
Objetos)

SÃO PAULO
2021

LISTA DE FIGURAS

Figura 1 - Tipos de lixo e suas cores	9
Figura 2 - Posicionamento de todas as cartas	10
Figura 3 - Jogada certa	11
Figura 4 - Combinação Inválida.....	11
Figura 5 - Retorna ao estado inicial.....	12
Figura 6 – Caixa de diálogo Game Over	12
Figura 7- Caixa de diálogo Sucesso.....	13
Figura 8 - Imagens utilizadas no jogo.....	15
Figura 9 - Diagrama UML.....	16
Figura 10 - Logo IntelliJ IDEA.....	17
Figura 11 - Gerador de Botões e aleatoriedade	17
Figura 12 - Inner Class TimeListener	18
Figura 13 - Classe que gera Popups.....	19
Figura 14 - Classe Inicial.....	20
Figura 15 - Classe que gera a janela da aplicação	21
Figura 16 - Classe que gera as cartas do jogo.....	22
Figura 17 - Bloco try-catch	23
Figura 18 - Geração de cartas.....	24
Figura 19 - Gerador de aleatoriedade	25
Figura 20 – Caixa de diálogo seleção de dificuldade	38
Figura 21 - Confirmação de dificuldade.....	38
Figura 22 - Tabuleiro inicial	39
Figura 23 - Par combinatório	39
Figura 24 - Par incorreto.....	40
Figura 25 - Cartas voltam ao estado inicial	40
Figura 26 - Todos os pares encontrados.....	41
Figura 27 - Tentativas insuficientes.....	41
Figura 28 – Caixa de diálogo como jogar	42
Figura 29 – Caixa de diálogo sobre.....	42

SUMÁRIO

OBJETIVO.....	5
INTRODUÇÃO	6
1 REGRAS DO JOGO.....	10
2 PLANO DE DESENVOLVIMENTO DO JOGO	14
3 PROJETO DO PROGRAMA	20
4 RELATÓRIO COM AS LINHAS DE CÓDIGO	26
5 APRESENTAÇÃO DO PROGRAMA EM FUNCIONAMENTO EM UM COMPUTADOR	38
BIBLIOGRAFIA	43
FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS	45

OBJETIVO

A Atividade Prática Supervisionada (APS) do 3º semestre tem como principal objetivo a criação de um jogo utilizando os conceitos de linguagem de programação orientada a objetos aprendidos em sala de aula. Tendo como tema a conscientização ambiental e a sustentabilidade nas grandes metrópoles.

Para isso foi desenvolvido um jogo da memória tendo como foco principal a educação sobre o descarte correto do lixo, algo que hoje em dia não é muito praticado nas casas dos brasileiros, nem tão pouco noticiado com frequência.

O jogo tem como objetivo mostrar os tipos de lixos recicláveis que encontramos em nosso cotidiano bem a cor de sua respectiva lixeira, algo que embora pareça muito simples, nem todos sabem ao certo o correto uso ou qual cor pertence a quem.

Usando todos os conceitos de orientação a objetos que foram obtidos através das aulas ao longo do semestre, foi possível desenvolver um simples jogo utilizando uma interface gráfica para facilitar a apresentação de tudo aquilo que foi codificado.

INTRODUÇÃO

Não existem achados concretos que provem a real data de criação do jogo da memória, no entanto alguns historiadores acreditam que o jogo foi criado por volta do século XV na China e era formado por um baralho de cartas duplicadas.

O jogo se popularizou como um clássico entre as crianças, além de entretê-las o jogo também é capaz de desenvolver algumas habilidades como a concentração, a autonomia, a confiança, o raciocínio lógico, a interpretação, a análise por comparação, entre muitas outras. Também é utilizada por professores de escolas infantis para acelerar o aprendizado em sala de aulas.

Segundo (TEIXEIRA, RICARDO, 2014) é por meio da utilização dos jogos que o aluno constrói seu conhecimento de maneira ativa e dinâmica e os sujeitos envolvidos estão geralmente mais propícios a ajuda mútua e a análise de erros e de acertos, ao qual proporciona uma reflexão profunda acerca dos conceitos que estão sendo discutidos.

O jogo em questão tem como objetivo orientar o jogador acerca do destino correto do lixo, a ONU (Organização das Nações Unidas) estima que são produzidos anualmente uma quantidade de 1,4 bilhão de toneladas de resíduos sólidos urbanos, o que dá uma média de aproximadamente 1,2 kg de lixo por dia (per capita), o que assusta ainda mais é que metade desse total é produzido por países desenvolvidos, daqui a dez anos esse número deve dobrar e passar dos 2,2 bilhões de toneladas anuais, se o consumo desenfreado e a falta de conscientização acerca do descarte correto e da reciclagem por parte das pessoas, teremos cerca de 9 bilhões de habitantes e 4 bilhões de toneladas de lixo por ano, se o ritmo se manter, logo teremos mais toneladas de lixo do que pessoas no planeta.

O Brasil ocupa o quarto lugar no ranking mundial de produção de lixo, seguido de Estados Unidos em primeiro, China em segundo e Índia em terceiro. Dos 11 milhões de toneladas de lixo produzidos aqui, apenas 1,28% têm como destino a reciclagem. Muito desse lixo é descartado de forma irregular sem o tratamento adequado, propiciando assim grandes áreas de descarte ou lixões a céu aberto além de contribuir com as enchentes em tempos de chuva já que grande parte também é descartada em córregos e rios.

Segundo (LAYARGUES, PHILIPPE, 2002) apesar da complexidade do tema, muitos programas de educação ambiental são ensinados nas escolas de forma

reducionista, já que em função da reciclagem, apenas se desenvolve a coleta seletiva do lixo deixando de lado o significado ideológico da reciclagem.

O jogo apresenta os principais tipos de lixo recicláveis que são facilmente encontrados em nosso cotidiano, são eles:

- **Metal:** representados pela cor amarela, são materiais que possuem uma durabilidade e resistência mecânica elevada e pode ser utilizado na fabricação de diversos itens como móveis, peças automotivas, utensílios de cozinhas e um dos materiais que são facilmente encontrados em residências no Brasil e no mundo, as famosas latinhas. São classificados como ferrosos (ferro e aço) e não ferroso (alumínio, cobre, chumbo, níquel, zinco). Os resíduos são prensados, classificados e enviados a estações de reciclagem onde são retiradas todas as impurezas, se não descartados de forma correta podem demorar cerca de 500 anos para se decomporem na natureza.
- **Orgânicos:** representado pela cor marrom, os lixos orgânicos são todos aqueles provenientes da vida animal ou vegetal como restos de verduras, frutas, legumes e outros alimentos. Embora não seja tão comum de se fazer, sua reciclagem é feita através da compostagem onde o lixo orgânico é separado e tratado para a produção de adubo sejam em áreas rurais ou em residências onde são armazenados em caixas de compostagem onde minhocas são utilizadas para triturar os alimentos e os transformarem em húmus.
- **Papel:** representado pela cor azul, o papel é fabricado através da extração da celulose de árvores, nesse processo de fabricação são utilizadas grandes quantidades de água e energia, o que torna a reciclagem do papel um grande negócio uma tonelada de aparas pode evitar o corte de 10 a 12 árvores provenientes de reflorestamento além da economia de energia e recursos utilizados na fabricação em especial a água. Se descartados na natureza levam cerca de 6 meses para se decompor.
- **Plástico:** representado pela cor vermelha, o plástico é um polímero derivado do petróleo, uma das maiores invenções do século 20, um material flexível e facilmente moldável. Embora parecem nem todos os

tipos de plásticos são iguais ou compatíveis quimicamente. Dentre os principais podemos destacar:

- Polietileno tereftalato (PET): utilizados na produção e envase de refrigerantes e água.
- Polietileno de alta densidade (PEAD): utilizados por fabricantes de engarrafados de bebidas, autopeças, tambores, baldes e outros produtos.
- Cloreto de polivinila (PVC): muito utilizado em tubos, conexões, em garrafas de água mineral e detergentes líquidos.
- Polietileno de baixa densidade (PEBD): muito utilizado na fabricação de sacos de plástico, nas embalagens de leite, recipientes, garrafas, componentes de computador, peças que necessitem solda entre outros.
- Polipropileno (PP): utilizado nas embalagens de massas, biscoitos, potes de margarina e manteiga, embalagem de salgadinhos entre outros.
- Poliestireno (PS), utilizado na fabricação de eletrodomésticos e copos descartáveis;

Se descartados de forma inadequada podem causar grande impacto no meio ambiente além da demora na decomposição que pode chegar a 100 anos, o plástico pode causar sérios efeitos na vida dos animais bem como a vida marinha aos quais são os maiores prejudicados.

- **Vidro:** representado pela cor verde, o vidro é composto por minerais como areia, barrilha, calcário e feldspato, o que possibilita ser reciclado inúmeras vezes. Assim com o papel a sua reciclagem tem impacto direto nos recursos naturais utilizados e na energia gasta na fabricação já que com um quilo de vidro se faz outro quilo, sendo assim não existem perdas na sua reciclagem. Não pode se determinar quanto tempo é necessário para se decompor na natureza.

Podemos assimilar então as devidas cores das lixeiras com seu respectivo tipo de lixo como na imagem a seguir.

Figura 1 - Tipos de lixos e suas cores



Fonte: <https://tnaplast.com.br/>

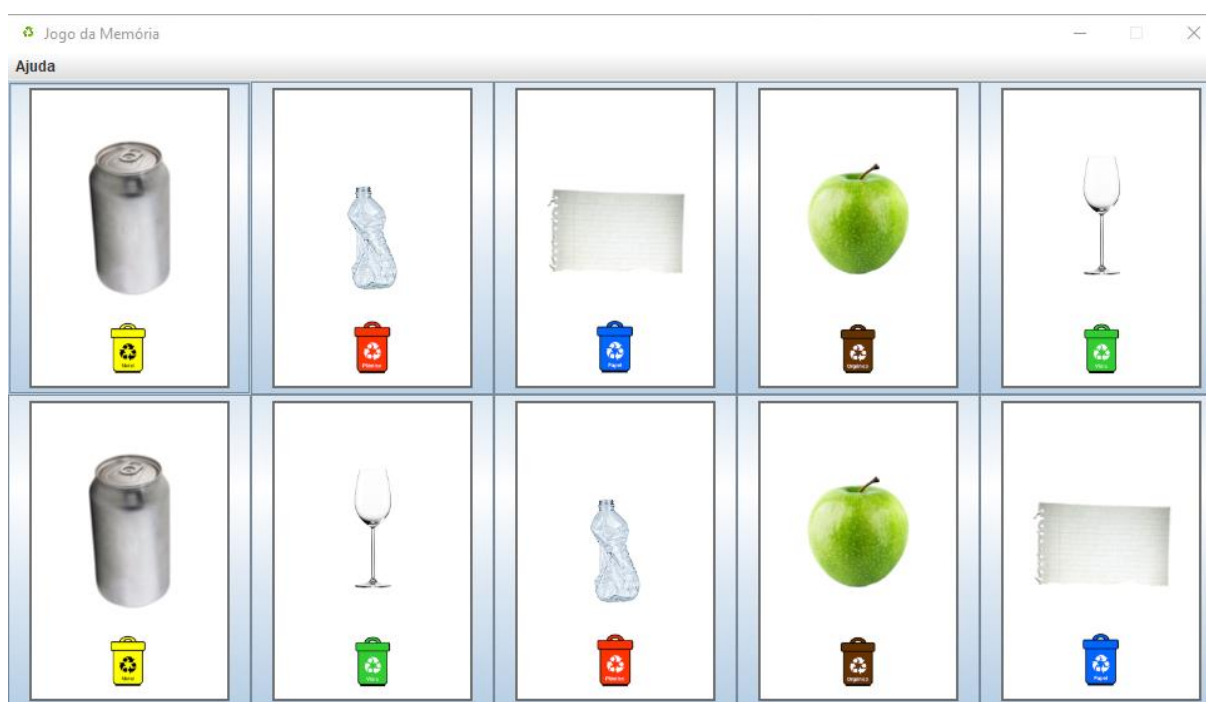
1 REGRAS DO JOGO

Assim como os demais jogos da memória que costumamos jogar, esse, embora digital, segue o mesmo modelo contendo apenas algumas variações de jogabilidade. Para ter êxito e conseguir encontrar todas as cartas durante o jogo, o jogador necessita apenas ter uma boa observação, concentração, uma boa memória e em alguns casos sorte, caso ele opte pelo nível mais difícil. No entanto esses não são requisitos totalmente necessários e sua falta não influencia a mecânica do jogo.

O Jogo da memória em questão tem algumas regras que embora pareçam simples se fazem necessárias de se entender.

A primeira regra do jogo é a mais simples de todas, o jogador tem como objetivo encontrar todas as combinações de cartas utilizando o menor ou o limite de jogadas selecionadas no início do jogo.

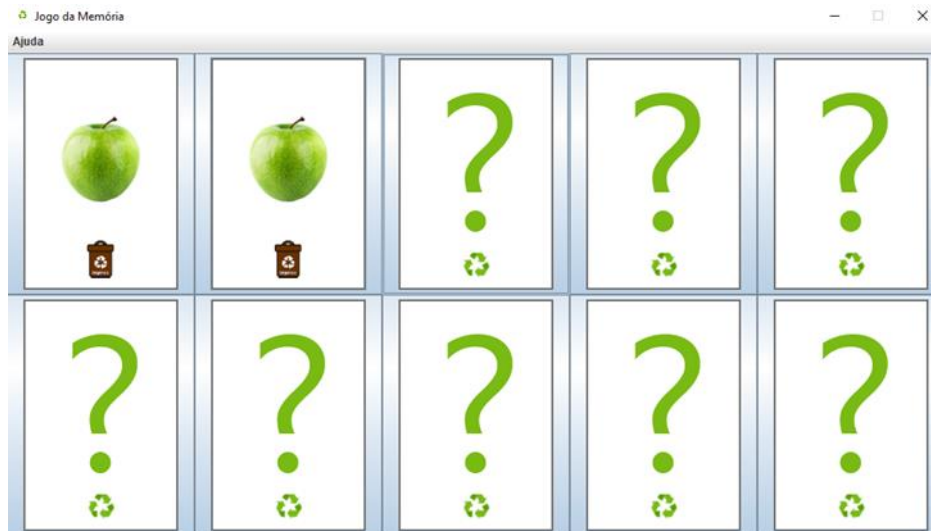
Figura 2 - Posicionamento de todas as cartas



Fonte: Autoria Própria

O jogo considera cada par de cartas virada como uma jogada, sendo assim o jogador só poderá virar duas cartas por vez, o jogo não possibilita virar mais de duas cartas ao mesmo tempo, evitando assim qualquer tipo de trapaça.

Figura 3 - Jogada certa

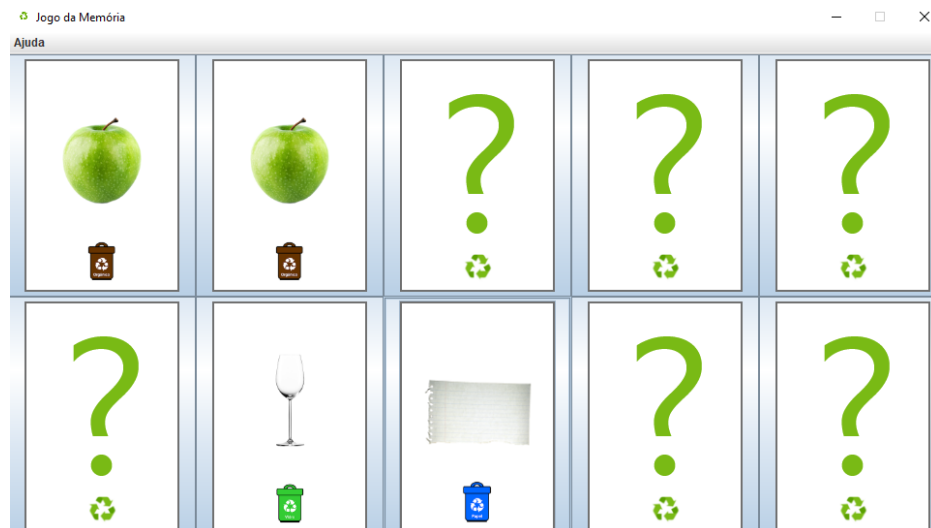


Fonte: Autoria Própria

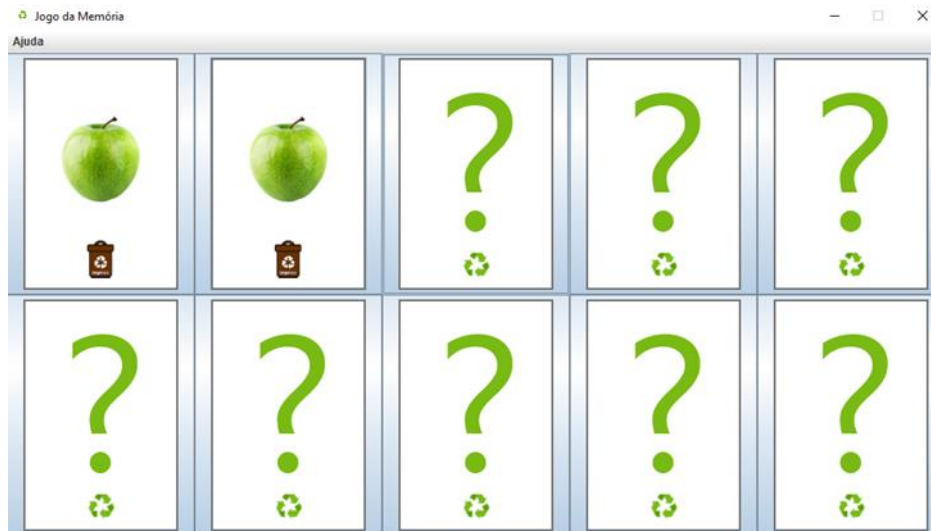
Se o jogador acertar a combinação de cartas elas permanecerão viradas para cima demonstrando que já foram descobertas, a cada acerto o contador de cartas encontradas é incrementado.

Se o jogador errar a combinação certa de cartas elas se voltaram para baixo e o erro será registrado, para mais tarde ser exibido nas estatísticas do jogo.

Figura 4 - Combinação Inválida

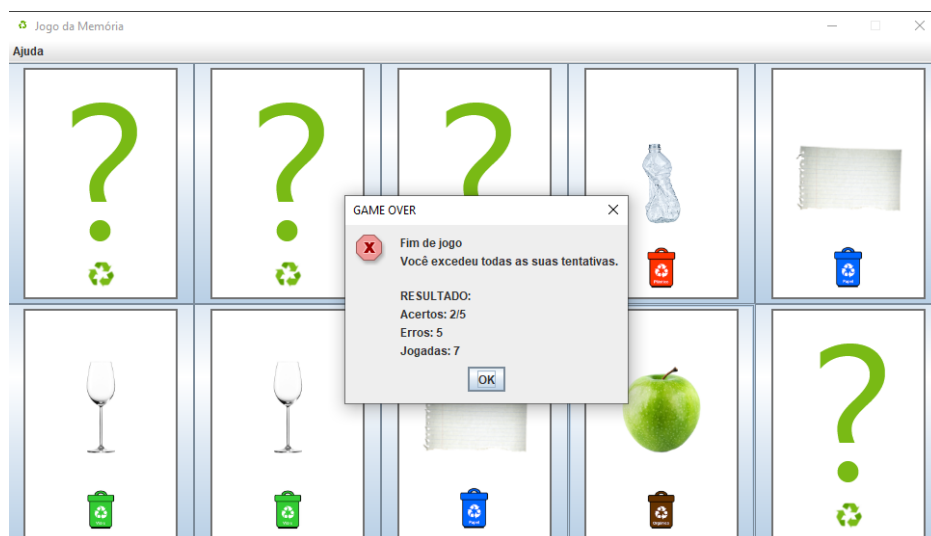


Fonte: Autoria Própria

Figura 5 - Retorna ao estado inicial

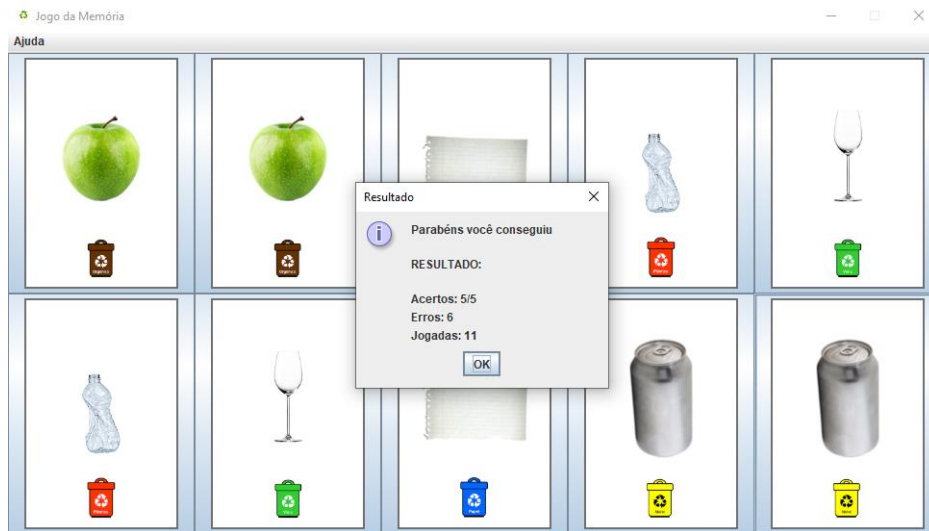
Fonte: Autoria Própria

Ao atingir a quantidade máxima de tentativas que o nível escolhido propõe, uma mensagem de “Game Over” será exibida na tela seguida dos resultados do jogador.

Figura 6 – Caixa de diálogo Game Over

Fonte: Autoria Própria

Quando acertar todas as combinações de cartas possíveis o programa exibirá uma tela com os resultados do jogo, informando o número de acertos, erros e jogadas feitas.

Figura 7- Caixa de diálogo Sucesso

Fonte: Autoria Própria

2 PLANO DE DESENVOLVIMENTO DO JOGO

Na fase de desenvolvimento inicial do jogo, foi discutido qual o estilo de jogo se adequaria melhor as necessidades, foi escolhido falar sobre os tipos de lixos recicláveis. Algo pouco se falado nos dias de hoje, mas que se praticado de forma consciente pode mudar a forma como vivemos, bem como o nosso meio-ambiente ao qual já se encontra em declínio, devido ao alto consumo de recursos naturais praticado pelo homem nos dias de atuais se a reciclagem não se tornar parte de nossa rotina, podemos transformar o planeta em um lixão a céu aberto.

Foi realizado uma reunião, ao qual através de um brainstorm todos os integrantes deixaram suas ideias expostas de modo que várias ideias, dos mais variados gêneros surgiram. Tinha que ser algo que de certa forma não demandasse um conhecimento muito avançado por parte da programação, visto que todos possuíam um conhecimento um tanto quanto limitado acerca de orientação a objetos e ao uso eficaz da linguagem de programação Java.

Depois de organizar as ideias e alinhá-las de modo que todos concordaram com o que foi proposto, o grupo então optou por fazer algo simples, mas que de certa forma atenderia os requisitos propostos na Atividade Prática, foi decidido então criar um clássico jogo da memória que teria como tema a correta reciclagem do lixo. Embora à primeira vista pareça algo muito simples de se implementar, o jogo da memória possui algumas mecânicas internas um tanto quanto complicadas de se compreender à primeira vista, já que um jogo feito em linguagem de programação tende a ser muito mais complicado de se fazer do que um jogo clássico de cartas.

Decidido o modelo de jogo e o tema que seria aplicado, o próximo passo foi a tentar imaginar como seria a interface gráfica, ou seja, como apresentaríamos o jogo ao jogador, em *fullscreen* (Tela Cheia) ou janela, as cores que iriam compor as cartas, os tamanhos, as imagens que iriam compor o jogo, tipos de fontes e se usaríamos algum tipo de framework. No final, foi decidido que seria utilizado nenhuma *engine* de criação ou gráfica, bem como nenhum *framework* afim de evitar incompatibilidade entre as plataformas bem como facilitar a manutenção posteriormente.

Foi decidido que o jogo seria em modo janela, já que a quantidade de cartas seria limitada a apenas dez, sobre as cores escolhidas no jogo foi pensado em algo mais minimalista possível, verde e branco foram as cores bases do jogo. Foram usadas cinco imagens ao qual representam os cinco tipos de lixo que podemos

reciclar de forma ecológica. Todas as imagens foram retiradas de um repositório de licença livre a fim de evitar problemas de copyright.

Figura 8 - Imagens utilizadas no jogo



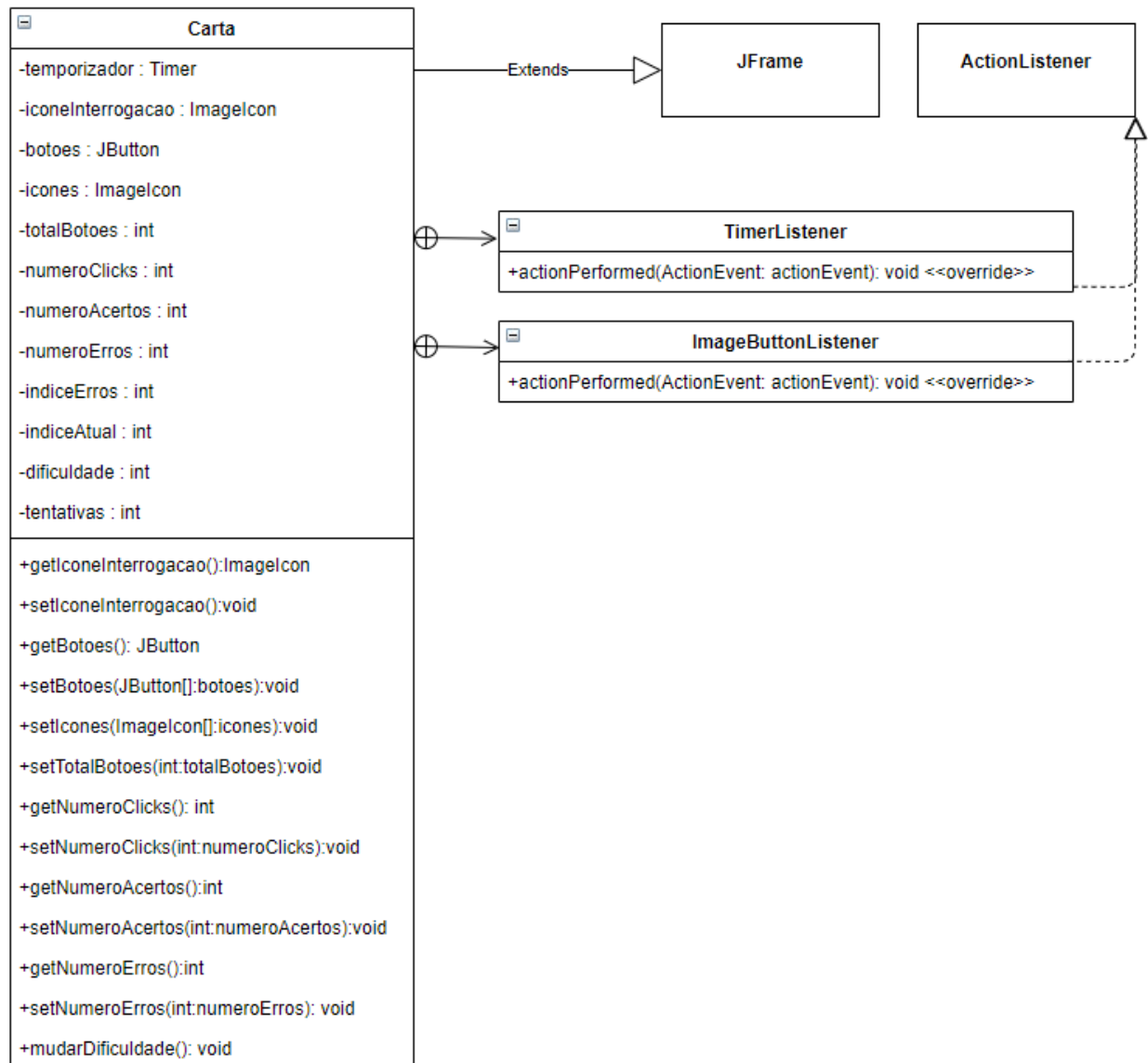
Fonte: <https://www.gratispng.com>

No entanto houve a necessidade de se manipular as imagens pois elas se encontravam em tamanhos variados, o que prejudicaria a jogabilidade já que algumas cartas eram maiores que as outras. Foi necessário então a utilização de um programa de manipulação de imagens, o programa Figma foi utilizado por ser fácil de usar, ser gratuito e funcionar em navegadores. As imagens então foram redimensionadas para tamanhos semelhantes umas das outras, cerca de 160 x 240 pixels , foram criados também algumas imagens como o caractere de interrogação usado para cobrir as cartas, o símbolo de reciclagem abaixo dele e as latinhas de lixo coloridas que aparecem logo abaixo do objeto.

Depois de decidida a parte gráfica, o próximo passo foi definir as classes e como o programa se comportaria, depois de muita pesquisa e discussões foi decidido criar um rascunho de como seria a classe de cartas do jogo. Classe essa responsável pela principal mecânica do jogo, pelos temporizadores das cartas e pela lógica de acerto e erro, além e claro da posição aleatória das cartas a cada jogo.

Após definida a classe e montado o rascunho de como a classe se comportaria, o rascunho foi convertido para um modelo de UML (*Unified Modeling Language*) que nada mais é do que uma linguagem de notação com a finalidade de ilustrar através de diagramas os modelos de classe e as relações que possuem entre si, além de facilitar o processo de desenvolvimento. Para desenvolver o UML da classe “Cartas” foi utilizado a ferramenta para criação de diagramas UML encontrada no site do Draw.IO (<https://app.diagrams.net/>), o resultado foi a imagem representada logo abaixo.

Figura 9 - Diagrama UML



Fonte: Autoria Própria

As funcionalidades internas do jogo foram de longe as mais árduas de se implementar, como já dito, embora pareça um jogo simples sua implementação pode se tornar complicada a partir do momento que funcionalidades extras ou efeitos são adicionadas.

Feito o diagrama da classe principal do programa, foi necessário escolher uma IDE (*Integrated Development Environment*) dentre as mais variadas disponíveis, o IntelliJ IDEA, criado e mantido pela empresa de software Jet Brains, foi a escolhida

pois apresentava uma interface mais amigável e um sistema de depuração mais preciso.

Figura 10 - Logo IntelliJ IDEA



Fonte: <https://www.jetbrains.com/pt-br/idea/>

Depois de escolhida a IDE, deu-se início a codificação do jogo, a parte que mais demandou tempo foi a de geração das cartas, foi criada uma função que adicionava carta por carta ao tabuleiro, no entanto, ao se adicionar mais imagens do que o programado o programa tinha como tendência falhar e apresentar mensagens de erro. Depois de muita pesquisa e da implementação de outro pseudocódigo, chegou-se a um laço contador que automaticamente adicionava todos os botões a janela independente de quantas imagens o são inseridas no *array*, como ilustrado abaixo.

Figura 11 - Gerador de Botões e aleatoriedade

```

62
63
64 // Loop que ira gerar os botões e adicionar as imagens e listeners.
65 for (int i = 0, j = 0; i < filenames.length; i++) {
66     this.icones[j] = new ImageIcon(Objects.requireNonNull(getClass().getResource(filenames[i])));
67     this.botoes[j] = new JButton( text: "");
68     this.botoes[j].addActionListener(new ImageButtonListener());
69     this.botoes[j].setIcon(this.getIcôneInterrogacao());
70     add(this.botoes[j++]);
71
72     this.icones[j] = this.icones[j - 1];
73     botoes[j] = new JButton( text: "");
74     botoes[j].addActionListener(new ImageButtonListener());
75     botoes[j].setIcon(this.getIcôneInterrogacao());
76     add(botoes[j++]);
77 }
78
79 // Gera posições aleatorias das cartas
80 Random gen = new Random();
81 for (int i = 0; i < getTotalBotoes(); i++) {
82     int rand = gen.nextInt(getTotalBotoes());
83     ImageIcon temp = icones[i];
84     icones[i] = icones[rand];
85     icones[rand] = temp;
86 }

```

Fonte: Autoria Própria

Outra importante funcionalidade do jogo foi o temporizador das cartas. Caso o jogador não tenha êxito na combinação de cartas, elas deveriam voltar ao estado

inicial logo em seguida, algo que de certa forma não estava funcionando, quando não havia uma combinação era mostrado uma mensagem de que o jogador havia errado e ao clicar em “OK” a função “bloquearCarta” era chamada. No entanto a função apresentava algumas falhas como o travamento e encerramento inesperado, ao pesquisar mais a fundo foi descoberto a utilização de eventos, ao qual através da implantação, encontramos a classe Timer que tem por objetivo manipular e registrar o tempo de execução de determinado evento. A partir daí, houve êxito na criação do efeito de virar a carta de cabeça para baixo novamente sem a necessidade de confirmar caixas de diálogo.

Figura 12 - Inner Class TimerListener

```

89
90      // Cria um temporizador que ira desvirar as cartas se as mesmas forem diferentes
91      temporizador = new Timer(delay: 1000, new TimerListener());
92
93  } catch (NullPointerException e) { ... }
94
95  }
96
97  class TimerListener implements ActionListener {
98
99      @Override
100      public void actionPerformed(ActionEvent actionEvent) {
101          // O temporizador depois de ativado, define o ícone das cartas como uma interrogação
102          botoes[indiceAtual].setIcon(iconeInterrogacao);
103          botoes[indiceErros].setIcon(iconeInterrogacao);
104          temporizador.stop();
105      }
106  }
107
108  }
109
110  }

```

Fonte: Autoria Própria

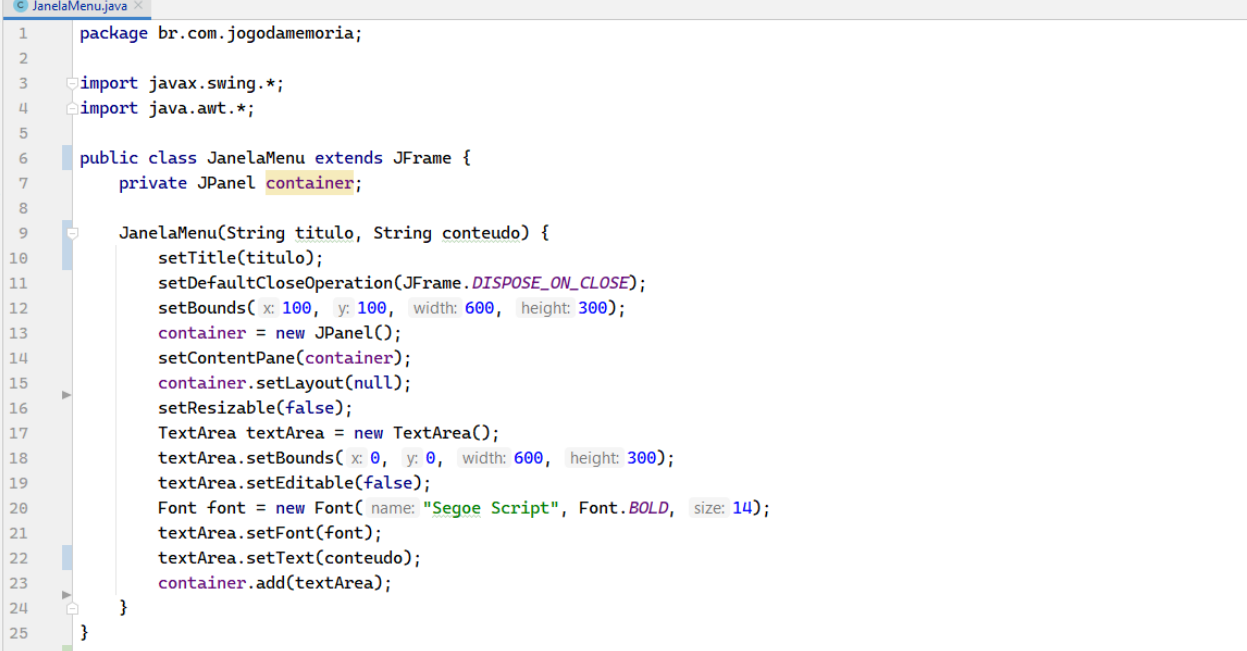
Houve também a ideia da criação de janelas externas que estariam acessíveis através do menu principal do programa com o título de Ajuda, esse menu possui os seguintes subitens:

- **Como Jogar?** : A tela de ajuda tem como objetivo apresentar ao jogador uma breve introdução do jogo, seguida de um tutorial em texto do correto funcionamento e regras do jogo.
- **Sobre o Jogo:** Como todo programa ou jogo criado, a janela *about* (sobre) sempre esteve presente nos programas. Sua finalidade é apresentar ao usuário ou jogador a versão atual do programa ou jogo ao qual possui, bem como todo o pessoal que participou do processo de desenvolvimento.

A classe utilizada para a criação dessa janela possui apenas um construtor que recebera duas *strings*, o título do subitem e o conteúdo. Logo depois é criado um outro

JFrame contendo um JPanel que exibira o texto inserido no construtor. A seguir é possível encontrar mais detalhes sobre a classe.

Figura 13 - Classe que gera Popups



```
1 package br.com.jogodamemoria;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class JanelaMenu extends JFrame {
7     private JPanel container;
8
9     JanelaMenu(String titulo, String conteudo) {
10         setTitle(titulo);
11         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
12         setBounds(x: 100, y: 100, width: 600, height: 300);
13         container = new JPanel();
14         setContentPane(container);
15         container.setLayout(null);
16         setResizable(false);
17         JTextArea textArea = new JTextArea();
18         textArea.setBounds(x: 0, y: 0, width: 600, height: 300);
19         textArea.setEditable(false);
20         Font font = new Font(name: "Segoe Script", Font.BOLD, size: 14);
21         textArea.setFont(font);
22         textArea.setText(conteudo);
23         container.add(textArea);
24     }
25 }
```

Fonte: Autoria Própria

3 PROJETO DO PROGRAMA

Foi utilizada a linguagem Java para o desenvolvimento do jogo, uma linguagem orientada a objetos consolidada ainda hoje no mercado, sua portabilidade permite que não somente esse jogo, mas qualquer outro tipo de aplicação escrito na linguagem possa rodar em quaisquer outros sistemas operacionais que suportem o Java.

Figura 14 - Classe Inicial



```
1 package br.com.jogodamemoria;
2
3 import javax.swing.JOptionPane;
4
5 public class Main {
6
7     // Execução do programa
8     public static void main(String[] args) {
9         // TODO Auto-generated method stub
10        try {
11            new JanelaDoJogo().setVisible(true);
12        } catch (RuntimeException e) {
13            JOptionPane.showMessageDialog( parentComponent: null,
14                message: "Ocorreu um erro ao iniciar o programa!\nDetalhes: " + e, title: "Erro",
15                JOptionPane.ERROR_MESSAGE);
16
17            System.exit( status: -1);
18        }
19    }
20 }
21 }
```

Fonte: Autoria Própria

O programa se inicia no arquivo chamado “Main.java”, ele é responsável pela execução do jogo. Na primeira linha é declarado o *package* do programa, ou seja, o nome do pacote que contém todas as classes do jogo, na linha 3 importamos o *javax.swing.JOptionPane* ao qual nos disponibiliza variados tipos de caixas de diálogo.

No entanto existem outras classes que são importadas no programa como:

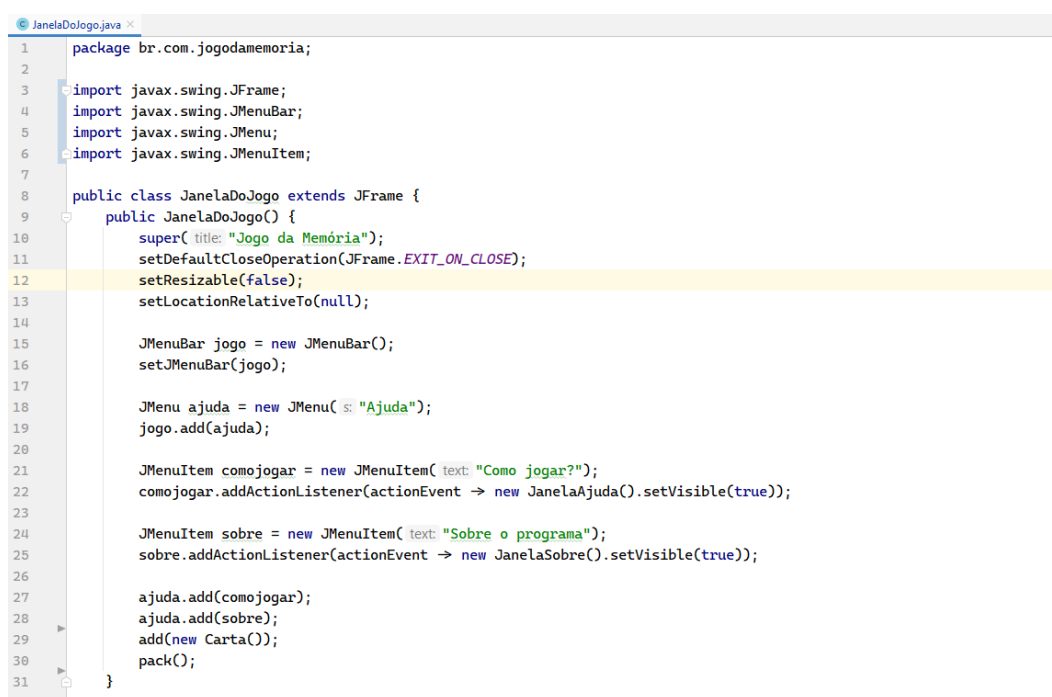
- **javax.swing.*:** contém uma coleção de classes responsáveis pela criação de janelas, pela implementação de componentes como botões e caixas de texto, além do gerenciamento de eventos.
- **java.awt.*:** AWT ou (*Abstract Window Toolkit*) assim como a classe swing implementa componentes gráficos na tela, além de gerenciar suas posições através de layouts pré-definidos.
- **java.awt.event.*:** gerencia todos os eventos gerados pela aplicação, também define gatilhos que serão disparados caso certo evento ocorra.

java.util.Objects: classe responsável por definir alguns métodos de manipulação de objetos, pode por exemplo verificar se um objeto é do *null* ou não, a fim de evitar o famoso *NullPointerException*.

- **javax.swing.Timer:** implementa a classe timer responsável pelos contadores internos do programa, como o timer utilizado no jogo para virar as cartas depois de certo tempo.
- **java.util.Random:** classe responsável pela geração de números aleatórios.

O programa se inicia na linha 8 com a declaração da função *main*, se tudo ocorrer como previsto o programa irá chamar a classe “JanelaDoJogo” e a configurar como *visible* (visível), note que declaramos caso ocorra algum erro uma exceção é lançada com a mensagem de erro e o sistema é encerrado.

Figura 15 - Classe que gera a janela da aplicação



```

1 package br.com.jogodamemoria;
2
3 import javax.swing.JFrame;
4 import javax.swing.JMenuBar;
5 import javax.swing.JMenu;
6 import javax.swing.JMenuItem;
7
8 public class JanelaDoJogo extends JFrame {
9     public JanelaDoJogo() {
10         super( title: "Jogo da Memória");
11         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12         setResizable(false);
13         setLocationRelativeTo(null);
14
15         JMenuBar jogo = new JMenuBar();
16         setJMenuBar(jogo);
17
18         JMenu ajuda = new JMenu( s: "Ajuda");
19         jogo.add(ajuda);
20
21         JMenuItem comojogar = new JMenuItem( text: "Como jogar?");
22         comojogar.addActionListener(actionEvent -> new JanelaAjuda().setVisible(true));
23
24         JMenuItem sobre = new JMenuItem( text: "Sobre o programa");
25         sobre.addActionListener(actionEvent -> new JanelaSobre().setVisible(true));
26
27         ajuda.add(comojogar);
28         ajuda.add(sobre);
29         add(new Carta());
30         pack();
31     }
32 }

```

Fonte: Autoria Própria

A classe “JanelaDoJogo” contém apenas o construtor que inicializara a classe, aqui é possível definir o título da aplicação, o que irá ocorrer caso o usuário clique no botão de fechar a janela, se a janela poderá ser redimensionada e sua localização padrão dentro do desktop. Das linhas 15 a 28 são declarados os menus de ajuda do

jogo, os menus possuem também um evento atrelado ao qual é disparado caso a opção seja selecionada.

Figura 16 - Classe que gera as cartas do jogo

```

1  package br.com.jogodamemoria;
2
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  import java.awt.GridLayout;
6  import javax.swing.JPanel;
7  import javax.swing.Timer;
8  import javax.swing.ImageIcon;
9  import javax.swing.JButton;
10 import javax.swing.JOptionPane;
11 import java.util.Objects;
12 import java.util.Random;
13
14
15 public class Carta extends JPanel {
16
17     static Timer temporizador;
18
19     private ImageIcon iconeInterrogacao;
20     private JButton[] botoes;
21     private ImageIcon[] icones;
22
23     private int totalBotoes;
24     private int numeroClicks;
25     private int numeroAcertos;
26     private int numeroErros;
27
28     private int indiceErros;
29     private int indiceAtual;
30
31     private int dificuldade;
32     private int tentativas;

```

Fonte: Autoria Própria

A classe “Carta” é responsável por gerar as cartas que serão utilizadas no jogo, da linhas 17 até 32 são declaradas algumas variáveis que serão utilizados durante o jogo, a variável *temporizador* é utilizada para conter um objeto do tipo *Timer*, *ImageIcon* e *JButtons* são utilizados para conter imagens e botões que serão gerados. As demais variáveis são utilizadas para conter informações e estatísticas do jogador, para mais tarde, caso o jogador vença, exibi-las na tela.

Figura 17 - Bloco try-catch

```

28 private int tentativas;
29
30 public Carta() {
31
32     //Excessão principal da classe, se não for possível carregar os arquivos necessario o programa é encerrado.
33     try {
34         // Excessão que tenta lidar com a entrada invalida do usuario.
35         try {
36             mudarDificuldade();
37             JOptionPane.showMessageDialog( parentComponent: null, message: "Você possui " + this.dificuldade +
38                 " tentativas!\nBoa sorte!", title: "Escolha a dificuldade", JOptionPane.INFORMATION_MESSAGE);
39         } catch (NumberFormatException e) {
40             JOptionPane.showMessageDialog( parentComponent: null, message: "Erro: o campo não pode estar vazio!",
41                 title: "Erro", JOptionPane.ERROR_MESSAGE);
42             System.exit( status: 1);
43         }
44     }
45
46     // Carrega o icone de interrogacao das cartas.
47     this.setIcôneInterrogacao(new ImageIcon(Objects.requireNonNull(getClass().
48         getResource( name: "res/interrogacao.png"))));
49
50     // A String filenames contém o caminho de todas as imagens usadas no jogo.
51     String[] filenames = {"res/metais.png", "res/organicos.png", "res/papel.png", "res/plastico.png",
52         "res/vidros.png"};

```

Fonte: Autoria Própria

Aqui é declarado o construtor principal da classe, ele começa com duas exceções, a primeira se certificara que não há qualquer objeto nulo, caso haja o programa é encerrado. A segunda é responsável pela escolha da dificuldade, caso o jogador digite uma opção invalida ou caractere não reconhecido a aplicação também é encerrada.

Logo em seguida através de um do setter “setIcôneInterrogacao” definimos o ícone padrão das cartas viradas para baixo e um array de *string* contendo os nomes das imagens utilizadas nas cartas.

Figura 18 - Geração de cartas

```

52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77

    int totalBtns = filenames.length * 2;

    this.setTotalBotoes(totalBtns);
    this.setBotoes(new JButton[this.totalBotoes]);
    this.setIcones(new ImageIcon[this.totalBotoes]);

    this.setNumeroAcertos(0);
    this.setNumeroErros(0);
    this.setNumeroClicks(0);

    // Loop que ira gerar os botões e adicionar as imagens e listeners.
    for (int i = 0, j = 0; i < filenames.length; i++) {
        this.icones[j] = new ImageIcon(Objects.requireNonNull(getClass().getResource(filenames[i])));
        this.botoes[j] = new JButton( text: "");
        this.botoes[j].addActionListener(new ImageButtonListener());
        this.botoes[j].setIcon(this.getIcôneInterrogacao());
        add(this.botoes[j++]);

        this.icones[j] = this.icones[j - 1];
        botoes[j] = new JButton( text: "");
        botoes[j].addActionListener(new ImageButtonListener());
        botoes[j].setIcon(this.getIcôneInterrogacao());
        add(botoes[j++]);
    }

```

Fonte: Autoria Própria

Aqui é inicializado algumas variáveis através de seus respectivos *setters* e na linha 64 é definido o loop responsável por atribuir a imagem ao botão, são utilizadas as variáveis *j* e *i*, *i* armazenara o índice das imagens e *j* o índice dos botões e ícones. O loop também adiciona um evento a cada botão gerado bem como o posiciona na janela.

Figura 19 - Gerador de aleatoriedade

```

77
78
79 // Gera posições aleatorias das cartas
80 Random gen = new Random();
81 for (int i = 0; i < getTotalBotoes(); i++) {
82     int rand = gen.nextInt(getTotalBotoes());
83     ImageIcon temp = icones[i];
84     icones[i] = icones[rand];
85     icones[rand] = temp;
86 }
87
88 setLayout(new GridLayout( rows: 2, cols: 7));
89
90 // Cria um temporizador que ira desvirar as cartas se as mesmas forem diferentes
91 temporizador = new Timer( delay: 1000, new TimerListener());
92
93 } catch (NullPointerException e) {
94     JOptionPane.showMessageDialog( parentComponent: null,
95         message: "Ocorreu um erro grave e o programa será encerrado!\n " +
96             e.getMessage(), title: "Erro", JOptionPane.ERROR_MESSAGE);
97     System.exit( status: -1);
98 }
99
100

```

Fonte: Autoria Própria

Criamos também um objeto do tipo *Random* que será responsável por gerar números aleatórios que serão usados dentro do contador nas linhas 82 a 85. Esse contador será responsável por gerar uma aleatoriedade na posição das cartas na janela.

4 RELATÓRIO COM AS LINHAS DE CÓDIGO

Logo abaixo se encontra os trechos de código da aplicação criada, assim como comentários e outras funcionalidades não mostradas acima. Também é possível obter o código através do repositório no GitHub (<https://github.com/nunees/Jogo-da-Memoria-APS-UNIP>).

- **main.java**

```
package br.com.jogodamemoria;
import javax.swing.*.*;
public class Main {
    // Execução do programa
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try {
            new JanelaDoJogo().setVisible(true);

        } catch (RuntimeException e) {
            JOptionPane.showMessageDialog(null,
                "Ocorreu um erro ao iniciar o programa!\nDetalhes: " + e, "Erro",
                JOptionPane.ERROR_MESSAGE);

            System.exit(-1);
        }
    }
}
```

- **JanelaDoJogo.java**

```
package br.com.jogodamemoria;

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

public class JanelaDoJogo extends JFrame {
```

```

public JanelaDoJogo() {
    super("Jogo da Memória");
    ImageIcon logo = new
    ImageIcon(Objects.requireNonNull(getClass().getResource("res/logo.png")));
    setIconImage(logo.getImage());
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setResizable(false);
    setLocationRelativeTo(null);

    JMenuBar jogo = new JMenuBar();
    setJMenuBar(jogo);

    JMenu ajuda = new JMenu("Ajuda");
    jogo.add(ajuda);

    JMenuItem comojogar = new JMenuItem("Como jogar?");
    comojogar.addActionListener(actionEvent -> new JanelaMenu("Como
    Jogar?",
        "Como Jogar?\n\n\n" +
            "O jogo da memória é um jogo clássico composto por imagens
    em um lado \n" +
            "e no outro apenas uma simples interrogação ou imagem
    aleatória.\n\n" +
            "O Jogo começa quando as imagens são viradas para baixo
    \n" +
            "para que o jogador não saiba onde se encontra seu outro
    par.\n\n" +
            "Deve-se virar apenas duas peças por vez, se as imagens não
    forem iguais \n" +
            "elas são viradas para baixo novamente e o jogador deve
    escolher outras \n" +
            "cartas até se formar um par de cartas iguais que deverão ficar
    viradas\n" +
            "para cima.\n\n" +

```

```

        "O Jogo termina quando todos os pares de cartas \n" +
        "forem encontrados ou as tentativas se
acabarem.\n").setVisible(true));

```

```

JMenuItem sobre = new JMenuItem("Sobre o jogo");
sobre.addActionListener(actionEvent -> new JanelaMenu("Sobre",
    "Atividade Prática Supervisionada\n" +
    "\n" +
    "Criado por:\n" +
    "\n" +
    "BRUNO GONZALEZ MASSONE – N582JF3\n" +
    "\n" +
    "FELIPE NUNES DA SILVA – N4935E0 \n" +
    "\n" +
    "FELIPE DE OLIVEIRA PEREIRA MAURÍCIO –
F2322A8").setVisible(true));

```

```

        ajuda.add(comojogar);
        ajuda.add(sobre);
        add(new Carta());
        pack();
    }

```

```

    }

```

- **Carta.java**

```

package br.com.jogodamemoria;

import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Objects;
import java.util.Random;

```

```

public class Carta extends JPanel {

    private static Timer temporizador;
    private ImageIcon iconeInterrogacao;
    private JButton[] botoes;
    private ImageIcon[] icones;
    private int totalBotoes;
    private int numeroClicks;
    private int numeroAcertos;
    private int numeroErros;
    private int indiceErros;
    private int indiceAtual;
    private int dificuldade;
    private int tentativas;

    public Carta() {
        //Exceção principal da classe, se não for possível carregar os arquivos
        //necessário o programa é encerrado.
        try {
            // Exceção que tenta lidar com a entrada invalida do usuário.
            try {
                mudarDificuldade();
                JOptionPane.showMessageDialog(null, "Você possui " +
this.dificuldade +
                " tentativas!\nBoa sorte!", "Escolha de dificuldade",
JOptionPane.INFORMATION_MESSAGE);
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(null, "Erro: o campo não pode
estar vazio!",
                "Erro", JOptionPane.ERROR_MESSAGE);
                System.exit(1);
            }
            // Carrega o ícone de interrogação das cartas.

```

```

        this.setIcôneInterrogacao(new
        ImageIcon(Objects.requireNonNull(getClass().
            getResource("res/interrogacao.png"))));

        // A String filenames contém o caminho de todas as imagens usadas
        no jogo.
        String[] filenames = {"res/metais.png", "res/organicos.png",
        "res/papel.png", "res/plastico.png",
        "res/vidros.png"};

        int totalBtns = filenames.length * 2;
        this.setTotalBotoes(totalBtns);
        this.setBotoes(new JButton[this.totalBotoes]);
        this.setIcônes(new ImageIcon[this.totalBotoes]);
        this.setNumeroAcertos(0);
        this.setNumeroErros(0);
        this.setNumeroClicks(0);

        // Loop que irá gerar os botões e adicionar as imagens e listeners.
        for (int i = 0, j = 0; i < filenames.length; i++) {
            this.icônes[j] = new
        ImageIcon(Objects.requireNonNull(getClass().getResource(filenames[i])));
            this.botoes[j] = new JButton("");
            this.botoes[j].addActionListener(new ImageButtonListener());
            this.botoes[j].setIcon(this.getIcôneInterrogacao());
            add(this.botoes[j++]);

            this.icônes[j] = this.icônes[j - 1];
            botoes[j] = new JButton("");
            botoes[j].addActionListener(new ImageButtonListener());
            botoes[j].setIcon(this.getIcôneInterrogacao());
            add(botoes[j++]);
        }

```

```

        // Gera posições aleatórias das cartas
        Random gen = new Random();
        for (int i = 0; i < getTotalBotoes(); i++) {
            int rand = gen.nextInt(getTotalBotoes());
            ImagemIcon temp = icones[i];
            icones[i] = icones[rand];
            icones[rand] = temp;
        }

        setLayout(new GridLayout(2, 7));

        // Cria um temporizador que irá desvirar as cartas se as mesmas
        forem diferentes
        temporizador = new Timer(1000, new TimerListener());

    } catch (NullPointerException e) {
        JOptionPane.showMessageDialog(null,
            "Ocorreu um erro grave e o programa será encerrado!\n " +
            e.getMessage(), "Erro", JOptionPane.ERROR_MESSAGE);
        System.exit(-1);
    }
}

// Getters e Setters
public ImagemIcon getIconeInterrogacao() {
    return iconeInterrogacao;
}

public void setIconeInterrogacao(ImagemIcon lockedCard) {
    this.iconeInterrogacao = lockedCard;
}

public int getTotalBotoes() {
    return totalBotoes;
}

public void setTotalBotoes(int totalBotoes) {

```

```

        this.totalBotoes = totalBotoes;
    }
    public void setBotoes(JButton[] botoes) {
        this.botoes = botoes;
    }
    public void setIcones(ImagemIcon[] icones) {
        this.icones = icones;
    }
    public int getNumeroClicks() {
        return this.numeroClicks;
    }
    public void setNumeroClicks(int numeroClicks) {
        this.numeroClicks += numeroClicks;
    }
    public int getNumeroAcertos() { return this.numeroAcertos; }
    public void setNumeroAcertos(int numeroAcertos) {
        this.numeroAcertos += numeroAcertos;
    }
    public int getNumeroErros() {
        return this.numeroErros;
    }
    public void setNumeroErros(int numeroErros) {
        this.numeroErros += numeroErros;
    }

    // Altera a dificuldade do jogo
    public void mudarDificuldade() {
        String input = JOptionPane.showInputDialog(null,"Escolha uma
dificuldade: \n\n 0 - Fácil (20 tentativas)\n" +
                "1 - Média (10 tentativas)\n2 - Difícil (5 tentativas)\n3 - Vidente(1
tentativa)" +
                "\n\n\nDigite a opção desejada","SELECIONE A
DIFICULDADE",JOptionPane.INFORMATION_MESSAGE);
        if (input == null)

```



```

        System.exit(1);
    int escolha = Integer.parseInt(input);
    if (escolha == 0) {
        this.dificuldade = 20;
    } else if (escolha == 1) {
        this.dificuldade = 10;
    } else if (escolha == 2) {
        this.dificuldade = 5;
    } else if (escolha == 3) {
        this.dificuldade = 1;
    } else {
        JOptionPane.showMessageDialog(null,
            "Escolha inválida, por padrão a dificuldade será configurada
como fácil!",
            "Erro", JOptionPane.ERROR_MESSAGE);
        this.dificuldade = 20;
    }
}

// Inner Class que gerencia o temporizador das cartas.
class TimerListener implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        // O temporizador depois de ativado, define o ícone das cartas como
uma interrogação
        botoes[indiceAtual].setIcon(iconoInterrogacao);
        botoes[indiceErros].setIcon(iconoInterrogacao);
        temporizador.stop();
    }
}

/*

```

* Inner classes criadas para organizar todas as classes que pertencem ao escopo da classe cartas.

* Facilitando assim a manutenção e a organização

*

*/

```
class ImageButtonListener implements ActionListener {
```

```
    @Override
```

```
    public void actionPerformed(ActionEvent actionEvent) {
```

```
        if (temporizador.isRunning())
```

```
            return;
```

```
        setNumeroClicks(1);
```

```
        // Checa qual botão o usuario clicou e incrementa o indice
```

```
        for (int i = 0; i < getTotalBotoes(); i++) {
```

```
            if (actionEvent.getSource() == botoes[i]) {
```

```
                botoes[i].setIcon(icones[i]);
```

```
                indiceAtual = i;
```

```
            }
```

```
        }
```

```
        if (getNumeroClicks() % 2 == 0) {
```

```
            if (indiceAtual == indiceErros) {
```

```
                numeroClicks--;
```

```
                return;
```

```
            }
```

/* Se as cartas não forem iguais, primeiro verifica se o jogador excedeu suas tentativas, logo em seguida

* ativa o temporizador.

* Se as cartas forem iguais, verifica se o jogar possui tentativas restantes, se também acertou todas as

```

        * combinações e logo em seguida apresenta uma mensagem com
os resultados;
        */
        if (icones[indiceAtual] != icones[indiceErros]) {
            // Mostra as imagens por cerca de 1 seg, antes de esconder
novamente
            numeroErros++;
            if ((tentativas + 1) >= dificuldade) {
                JOptionPane.showMessageDialog(null, "Fim de jogo\nVocê
excedeu todas as suas tentativas.\n\nRESULTADO: \nAcertos: " +
                    getNumeroAcertos() + "/" + 5 + "\nErros: " +
getNumeroErros() +
                    "\nJogadas: " + getNumeroClicks() / 2, "GAME
OVER",
                    JOptionPane.ERROR_MESSAGE);
                System.exit(1);
            } else {
                tentativas++;
            }

            temporizador.start();
        } else {
            if (getNumeroAcertos() == 4) {
                /*
                * Como o contador se de acertos se inicia em 0, se faz
necessário incrementar caso todas as
                * cartas sejam encontradas, para então exibir corretamente o
número de acertos na caixa de
                * diálogo.
                */
                setNumeroAcertos(1);
                JOptionPane.showMessageDialog(null, "Parabéns você
conseguiu\n\nRESULTADO: \n\nAcertos: " +

```

```

        getNumeroAcertos() + "/5" + "\nErros: " +
getNumeroErros() +
        "\nJogadas: " + getNumeroClicks() / 2, "Resultado",
JOptionPane.INFORMATION_MESSAGE);
        System.exit(0);
    } else {
        setNumeroAcertos(1);
    }

    }
} else {
    // Grava as jogadas erradas
    indiceErros = indiceAtual;
}
}
}
}
}

```

- **JanelaMenu.java**

```

package br.com.jogodamemoria;

import javax.swing.*.*;
import java.awt.*.*;

public class JanelaMenu extends JFrame {
    private JPanel container;

    JanelaMenu(String titulo, String conteudo) {
        setTitle(titulo);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setBounds(100, 100, 600, 300);
        container = new JPanel();
        setContentPane(container);
        container.setLayout(null);
        setResizable(false);
    }
}

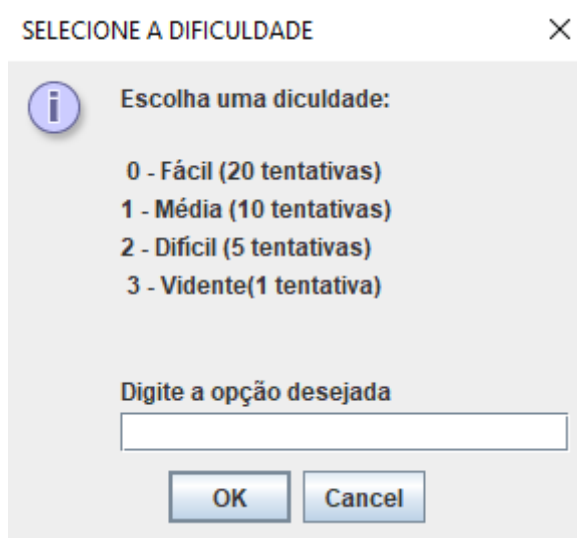
```

```
    TextArea textArea = new TextArea();
    textArea.setBounds(0, 0, 600, 300);
    textArea.setEditable(false);
    Font font = new Font("Segoe Script", Font.BOLD, 14);
    textArea.setFont(font);
    textArea.setText(conteudo);
    container.add(textArea);
}
}
```

5 APRESENTAÇÃO DO PROGRAMA EM FUNCIONAMENTO EM UM COMPUTADOR

Ao iniciar o jogo o jogador será questionado sobre o nível de dificuldade que deseja usar. Cada nível possui um número de tentativas pré-estabelecidas, caso o jogador insira um número de dificuldade que não se encontre no menu, o programa automaticamente escolhera a dificuldade fácil. Vale lembrar que caso o conteúdo do input seja uma string o programa exibira uma mensagem de erro e encerrara.

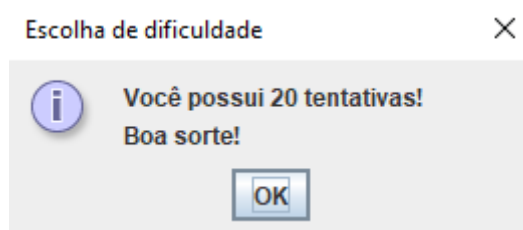
Figura 20 – Caixa de diálogo seleção de dificuldade



Fonte: Autoria Própria

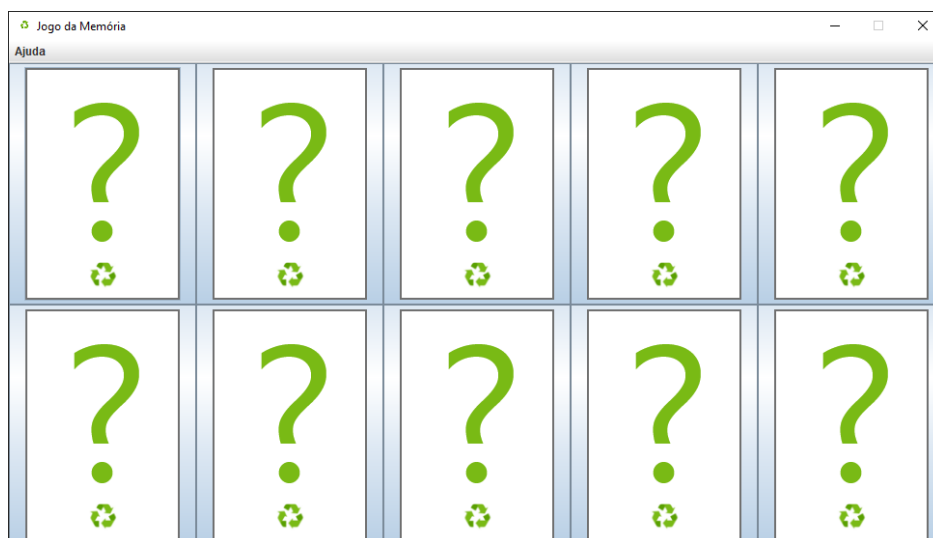
Caso o jogador insira uma opção válida o programa irá confirmar a escolha e exibir a quantidade de tentativas que o usuário terá.

Figura 21 - Confirmação de dificuldade



Fonte: Autoria Própria

Após isso é exibida a tela inicial do jogo onde o jogador fara sua primeira jogada e tentar encontrar o par correto.

Figura 22 - Tabuleiro inicial

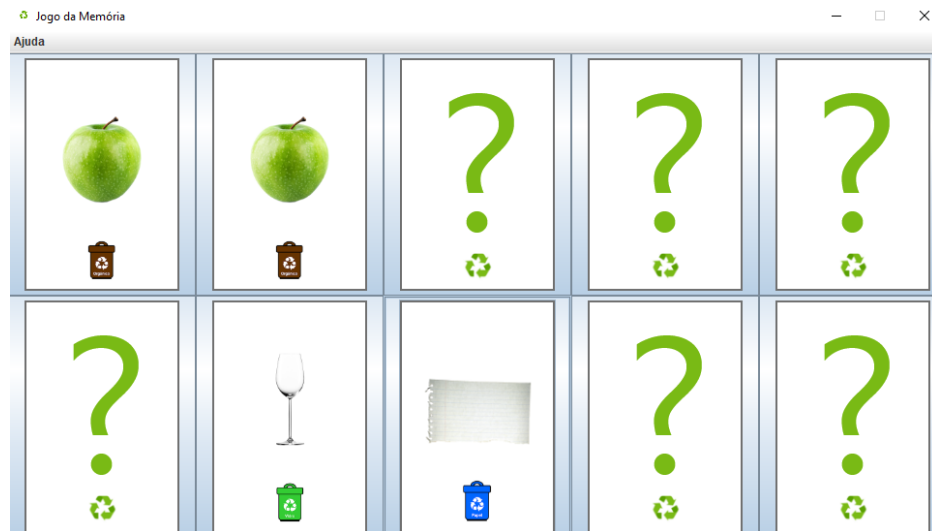
Fonte: Autoria Própria

Caso acerte a combinação de cartas, elas permaneceram viradas para cima demonstrando que já foram encontradas.

Figura 23 - Par combinatório

Fonte: Autoria Própria

Caso o jogador erre a combinação de cartas, elas serão exibidas durante cerca de um segundo e logo depois será mostrada a carta de interrogação demonstrando que o jogador não acertou a combinação de carta.

Figura 24 - Par incorreto

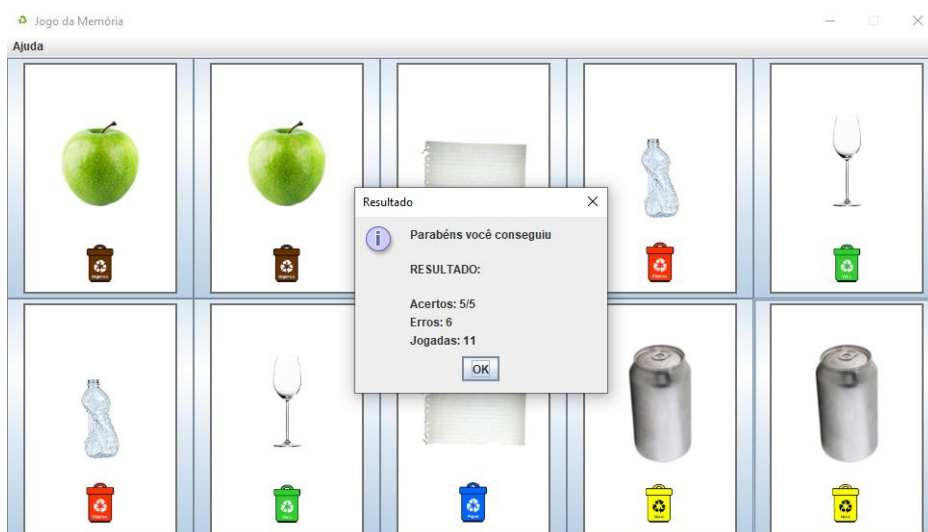
Fonte: Autoria Própria

Figura 25 - Cartas voltam ao estado inicial

Fonte: Autoria Própria

Caso o jogador consiga encontrar todos os pares de cartas, é exibido uma mensagem o parabenizando, bem como seus resultados de jogo.

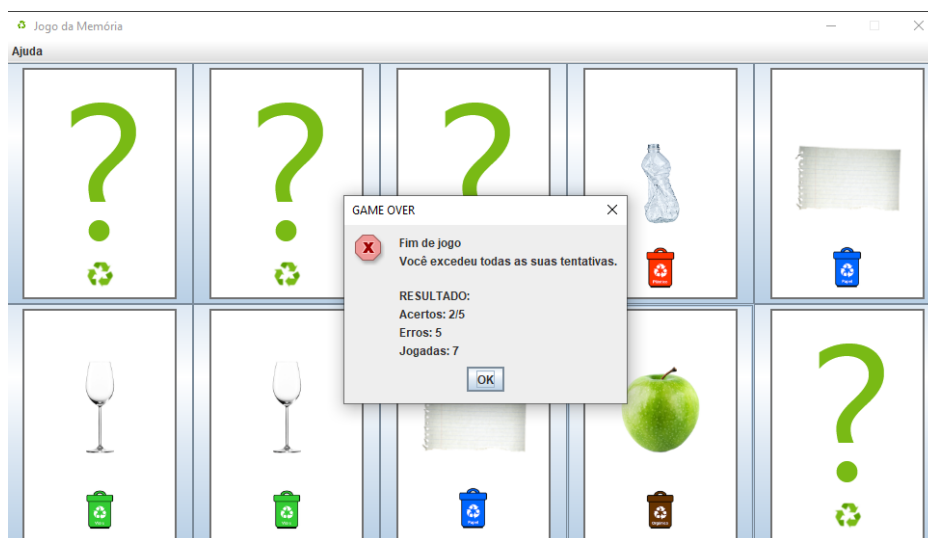
Figura 26 - Todos os pares encontrados



Fonte: Autoria Própria

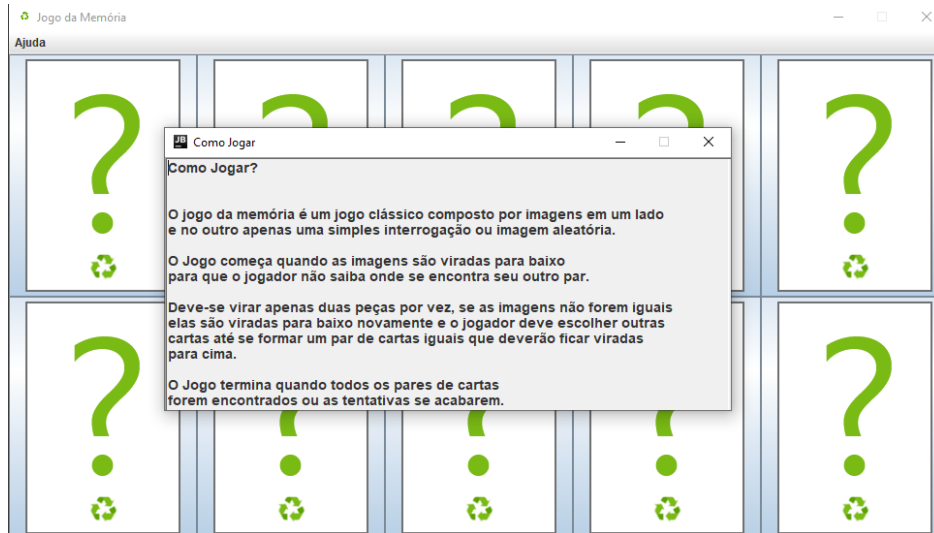
Se não conseguir acertar todas as combinações, o jogo termina e é exibido uma mensagem de “GAME OVER” seguido dos resultados do jogo.

Figura 27 - Tentativas insuficientes



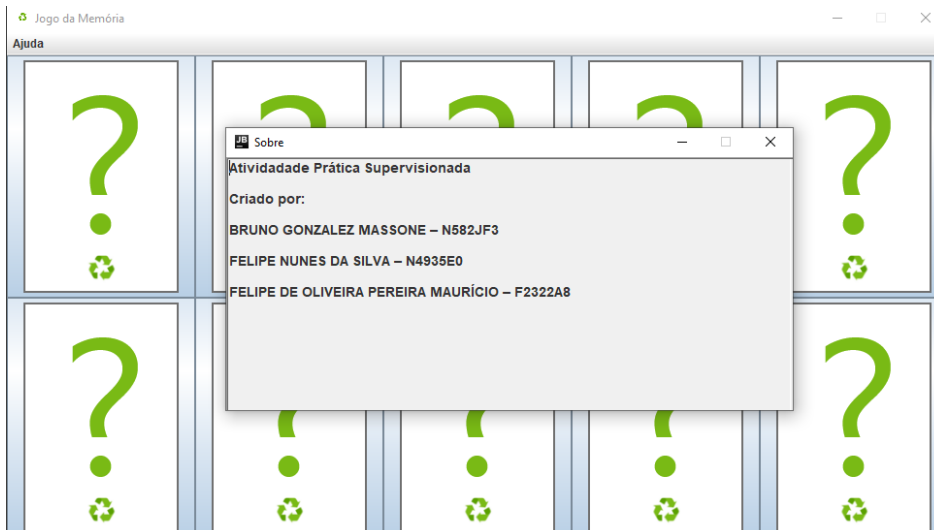
Fonte: Autoria Própria

Caso o jogador não saiba como jogar, no menu ajuda se encontra a tela “Como Jogar”.

Figura 28 – Caixa de diálogo como jogar

Fonte: Autoria Própria

Ainda no menu ajuda existe a opção “Sobre”, nela se encontra o nome dos integrantes que participaram do processo de desenvolvimento do jogo.

Figura 29 – Caixa de diálogo sobre

Fonte: Autoria Própria

BIBLIOGRAFIA

China e Estados Unidos lideram lista de países que mais geram lixo eletrônico. **ONU News**. Disponível em: < <https://news.un.org/pt/story/2020/07/1719142>.> . Acesso em 15 de abril de 2021.

Jogo da Memória. **Museu Histórico Prudente de Moraes**. Disponível em: <http://museuprudentedemoraes.piracicaba.sp.gov.br/pt_BR/quebra-cabecas>. Acesso em 15 de abril de 2021.

Um jogo para todas as disciplinas. **Appai**. Disponível em: <<https://www.appai.org.br/appai-educacao-revista-appai-educar-edicao-122-um-jogo-para-todas-as-disciplinas/>>. Acesso em 21 de abril de 2021.

TEIXEIRA, Ricardo Roberto Plaza; APRESENTAÇÃO, Katia Regina dos Santos da. Jogos em sala de aula e seus benefícios para a aprendizagem da matemática. Revista Linhas, Florianópolis, v. 15, n. 28, p. 302-323, jan./jun. 2014.

Em discussão - Rumo aos 4 bilhões de toneladas por ano. **Senado Federal**. Disponível em: <<http://www.senado.gov.br/noticias/jornal/emdiscussao/residuos-solidos/materia.html?materia=rumo-a-4-bilhoes-de-toneladas-por-ano.html>>. Acesso em 22 de abril de 2021.

LAYARGUES, Philippe. O cinismo da reciclagem: o significado ideológico da reciclagem da lata de alumínio e suas implicações para a educação ambiental.

LOUREIRO, F.; LAYARGUES, P.; CASTRO, R. (Orgs.) Educação ambiental: repensando o espaço da cidadania. São Paulo: Cortez, 2002, 179-22

Reciclagem além das latas de alumínio. **Recicla Sampa**. Disponível em: <<https://www.reciclasampa.com.br/artigo/entenda-por-que-a-reciclagem-de-metais-deve-ir-alem-das-latas-de-aluminio>>. Acesso em 25 de abril de 2021.

Reciclagem de Papel. **Portal resíduos sólidos**. Disponível em: <<https://portalresiduossolidos.com/reciclagem-de-papel-2/>>. Acesso em 2 de maio de 2021

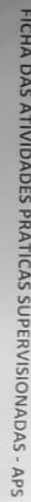
Reciclagem de Plástico. **Recicla Sampa**. Disponível em: <<https://www.reciclasampa.com.br/artigo/tudo-o-que-voce-precisa-saber-sobre-reciclagem-de-plastico>>. Acesso em 13 de maio de 2021.

Reciclagem de Vidro: tudo o que você precisa saber. **PS do Vidro**. Disponível em: <<https://www.psdovidro.com.br/descubra-tudo-sobre-a-reciclagem-de-vidro/>>. Acesso em 13 de maio de 2021.

DEITEL, H.; DEITEL, P. Java – Como Programar. 8ª. ed. São Paulo: Prentice-Hall, 2011.

FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS

[illegible]



RA: N4935E0

TURNO: NoiteANO GRADE: 2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
11/4/2021	Organização do grupo	6h	<i>Paula M.</i>		
12/4/2021	Discussão sobre o tema do trabalho	6h	<i>Paula M.</i>		
18/4/2021	Especificação de requisitos funcionais e não funcionais	5h	<i>Paula M.</i>		
20/4/2021	Início do desenvolvimento do jogo	6h	<i>Paula M.</i>		
21/4/2021	Pesquisas relacionadas ao tema	7h	<i>Paula M.</i>		
25/4/2021	Criação da parte visual do jogo	7h	<i>Paula M.</i>		
28/4/2021	Pesquisas e código	6h	<i>Paula M.</i>		
5/5/2021	Pesquisas e código	6h	<i>Paula M.</i>		
15/5/2021	Pesquisas e código	6h	<i>Paula M.</i>		
18/5/2021	Pesquisas e código	6h	<i>Paula M.</i>		
20/5/2021	Pesquisas e código	6h	<i>Paula M.</i>		
24/5/2021	Finalização e correção de erros	8h	<i>Paula M.</i>		

(1) Horas atribuídas de acordo com o regulamento das Atividades Supervisionadas do curso.

TOTAL DE HORAS ATRIBUIDAS: _____

AValiação:

NOTA: _____

DATA: _____/

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Adriano de Oliveira Pereira MouraCURSO: Conciliação da Competência com a CondiçãoCÓDIGO DA ATIVIDADE: 76B9SEMESTRE: 2021/1ANO GRÁFICO: 2021

CC3P1P.º

F2322A-8

SEMESTRE: 3ºANO GRÁFICO: 2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO TUTOR
17/4	Discursão sobre o tema da tarefa	6h			
18/4	Exatificação de pesquisa sobre	10h			
20/4	Análise da diversidade de dados e A	10h			
21/4	Aplicação de dados de dados e A	10h			
25/4	Exatificação de dados de dados e A	10h			
28/4	Exatificação de dados de dados e A	10h			
05/5	Exatificação de dados de dados e A	10h			
18/5	Exatificação de dados de dados e A	10h			
24/5	Exatificação de dados de dados e A	10h			

(1) Horas atribuídas de acordo com o planejamento da disciplina e/ou da supervisão da APS.

TOTAL DE HORAS ATRIBUÍDAS:

AVALIANDO:

Aprovação ou Rejeição:

NOTA:

DATA:

CARRANDO E ASSINATURA DO COORDENADOR DO CURSO