2LM Game Studio - Projetos de Desenvolvimento de Jogos

RELATÓRIO TÉCNICO

1. Jogo da Forca
- Um clássico jogo onde o jogador deve adivinhar uma palavra antes de completar o desenho da forca.
- Implementado com mecânicas de input do usuário e verificação de letras.

PRINTS:

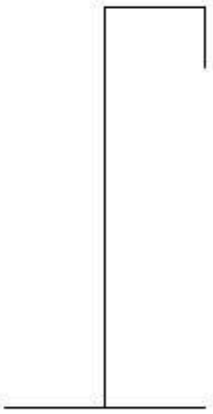
Jogo_da_forca

Categoria: Cidades

Palavra: _____

Tentativas restantes: 5

Letras erradas: []



Jogo_da_forca

Categoria: Filmes

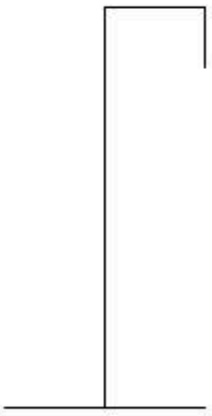
Palavra: titanic

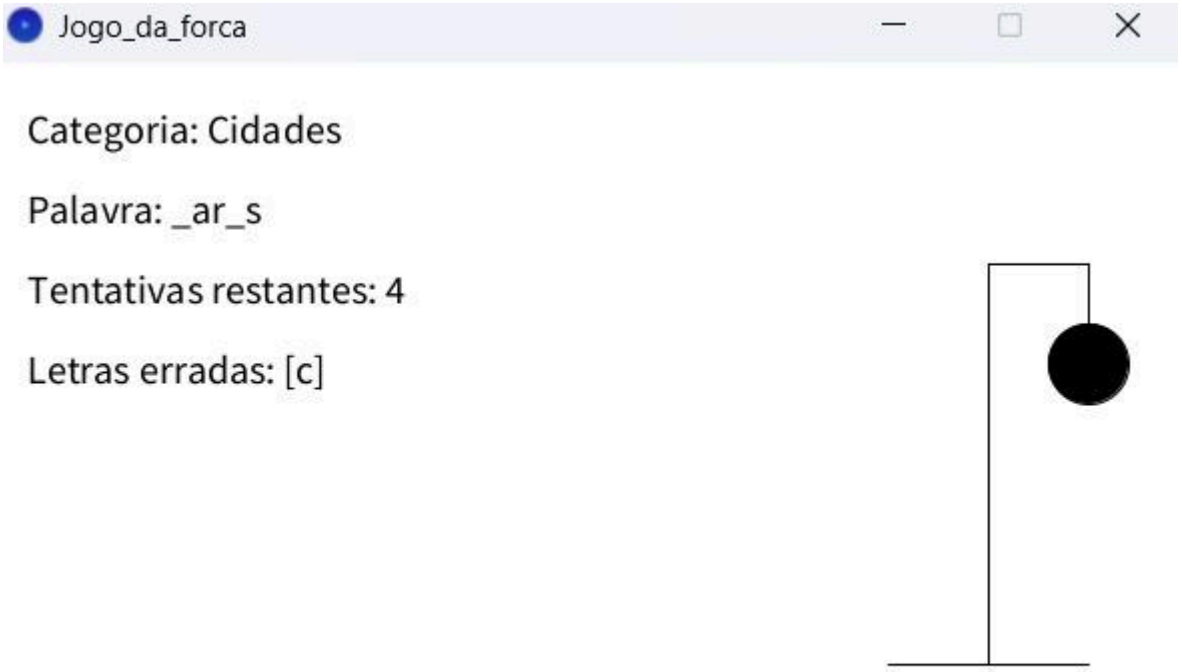
Tentativas restantes: 6

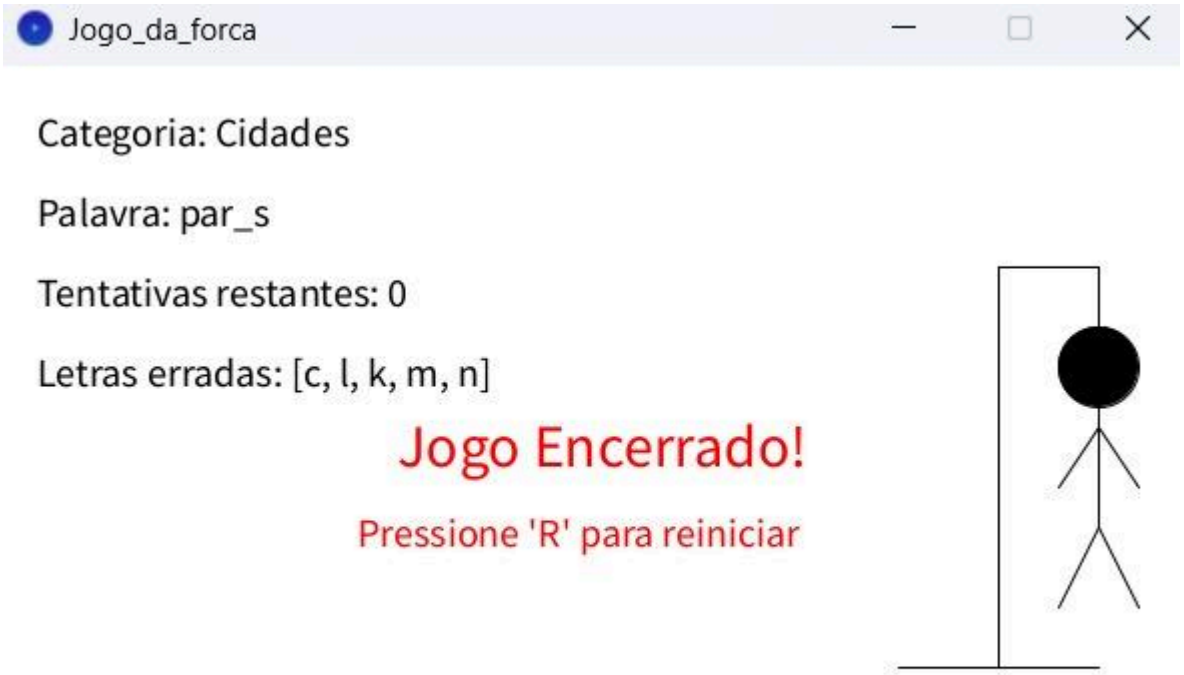
Letras erradas: []

Você venceu!

Pressione 'R' para reiniciar







CÓDIGO-FONTE:

```
String[][] categorias = {
  {"maça", "banana", "uva", "laranja", "manga"},
```

```
  {"paris", "londres", "berlim", "roma", "lisboa"},
  {"titanic", "avatar", "matrix", "interestelar", "inception"},
  {"vermelho", "azul", "verde", "amarelo", "roxo"}
};


String[] nomesCategorias = {"Frutas", "Cidades", "Filmes", "Cores"};
int categoriaSelecionada;
String palavra;
char[] palavraOculta;
ArrayList<Character> letrasErradas = new ArrayList<Character>();
int tentativasRestantes = 5;
boolean jogoEncerrado = false;
boolean venceu = false;

void setup() {
  size(600, 400);
  escolherPalavra();
}

void escolherPalavra() {
  categoriaSelecionada = int(random(categorias.length));
  palavra =
categorias[categoriaSelecionada][int(random(categorias[categoriaSelecionada].length))];
  palavraOculta = new char[palavra.length()];
  for (int i = 0; i < palavraOculta.length; i++) {
    palavraOculta[i] = '_';
  }
}

void draw() {
  background(255);
  fill(0);
  textSize(20);
  text("Categoria: " + nomesCategorias[categoriaSelecionada], 20, 40);
  text("Palavra: " + new String(palavraOculta), 20, 80);
  text("Tentativas restantes: " + tentativasRestantes, 20, 120);
  text("Letras erradas: " + letrasErradas, 20, 160);
  desenharForca();

  if (jogoEncerrado) {
    textSize(30);
    if (venceu) {
      fill(0, 255, 0);
      text("Você venceu!", 220, 200);
    } else {
      fill(255, 0, 0);
      text("Jogo Encerrado!", 200, 200);
```

```
    }
    textSize(20);
    text("Pressione 'R' para reiniciar", 180, 240);
  }
}

void desenharForca() {
  stroke(0);
  line(450, 300, 550, 300);
  line(500, 300, 500, 100);
  line(500, 100, 550, 100);
  line(550, 100, 550, 130);

  if (tentativasRestantes <= 4) {
    ellipse(550, 150, 40, 40);
  }
  if (tentativasRestantes <= 3) {
    line(550, 170, 550, 230);
  }
  if (tentativasRestantes <= 2) {
    line(550, 180, 530, 210);
    line(550, 180, 570, 210);
  }
  if (tentativasRestantes <= 1) {
    line(550, 230, 530, 270);
    line(550, 230, 570, 270);
  }
}

void keyPressed() {
  if (jogoEncerrado && key == 'r') {
    reiniciarJogo();
    return;
  }

  if (tentativasRestantes > 0 && !jogoEncerrado) {
    char letra = Character.toLowerCase(key);
    boolean acertou = false;

    for (int i = 0; i < palavra.length(); i++) {
      if (palavra.charAt(i) == letra) {
        palavraOculta[i] = letra;
        acertou = true;
      }
    }

    if (!acertou && !letrasErradas.contains(letra)) {
      letrasErradas.add(letra);
```

```
      tentativasRestantes--;
    }

    if (new String(palavraOculta).equals(palavra)) {
      jogoEncerrado = true;
      venceu = true;
    } else if (tentativasRestantes == 0) {
      jogoEncerrado = true;
      venceu = false;
    }
  }
}


void reiniciarJogo() {
  tentativasRestantes = 6;
  letrasErradas.clear();
  jogoEncerrado = false;
  venceu = false;
  escolherPalavra();
}
```

2. Jogo da Velha

- O tradicional Tic-Tac-Toe, onde dois jogadores competem para formar uma linha de três símbolos iguais.
- Inclui verificação de vitória e lógica de jogabilidade.

PRINTS:

CÓDIGO-FONTE:

```
int[][] board = new int[3][3];
boolean playerTurn = true;
boolean gameOver = false;
```

```
int winner = 0;
boolean againstComputer = false;
boolean modeSelected = false;

void setup() {
  size(300, 300);
}

void draw() {
  background(245, 235, 220);
  if (!modeSelected) {
    drawModeSelection();
  } else {
    drawBoard();
    if (gameOver) {
      displayWinner();
    }
  }
}

void mousePressed() {
  if (!modeSelected) {
    if (mouseY > height / 4 && mouseY < height / 4 + 100) {
      if (mouseX < width / 2) {
        againstComputer = false;
      } else {
        againstComputer = true;
      }
      modeSelected = true;
      resetBoard();
    }
    return;
  }

  if (!gameOver) {
    int row = mouseY / 100;
    int col = mouseX / 100;
    if (row >= 0 && row < 3 && board[row][col] == 0) {
      board[row][col] = playerTurn ? 1 : 2;
      playerTurn = !playerTurn;
      checkWinner();
      if (againstComputer && !playerTurn && !gameOver) {
        computerMove();
      }
    }
  } else {
    resetBoard();
  }
```

```
}

void drawBoard() {
  stroke(139, 69, 19);
  strokeWeight(6);
  for (int i = 1; i < 3; i++) {
    line(i * 100, 0, i * 100, 300);
    line(0, i * 100, 300, i * 100);
  }
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      if (board[i][j] == 1) {
        drawX(j, i);
      } else if (board[i][j] == 2) {
        drawO(j, i);
      }
    }
  }
}

void drawX(int x, int y) {
  stroke(139, 69, 19);
  strokeWeight(8);
  line(x * 100 + 20, y * 100 + 20, x * 100 + 80, y * 100 + 80);
  line(x * 100 + 80, y * 100 + 20, x * 100 + 20, y * 100 + 80);
}

void drawO(int x, int y) {
  stroke(139, 69, 19);
  strokeWeight(8);
  noFill();
  ellipse(x * 100 + 50, y * 100 + 50, 60, 60);
}

void checkWinner() {
  for (int i = 0; i < 3; i++) {
    if (board[i][0] != 0 && board[i][0] == board[i][1] && board[i][1] == board[i][2]) {
      winner = board[i][0];
      gameOver = true;
      return;
    }
    if (board[0][i] != 0 && board[0][i] == board[1][i] && board[1][i] == board[2][i]) {
      winner = board[0][i];
      gameOver = true;
      return;
    }
  }
  if (board[0][0] != 0 && board[0][0] == board[1][1] && board[1][1] == board[2][2]) {
```

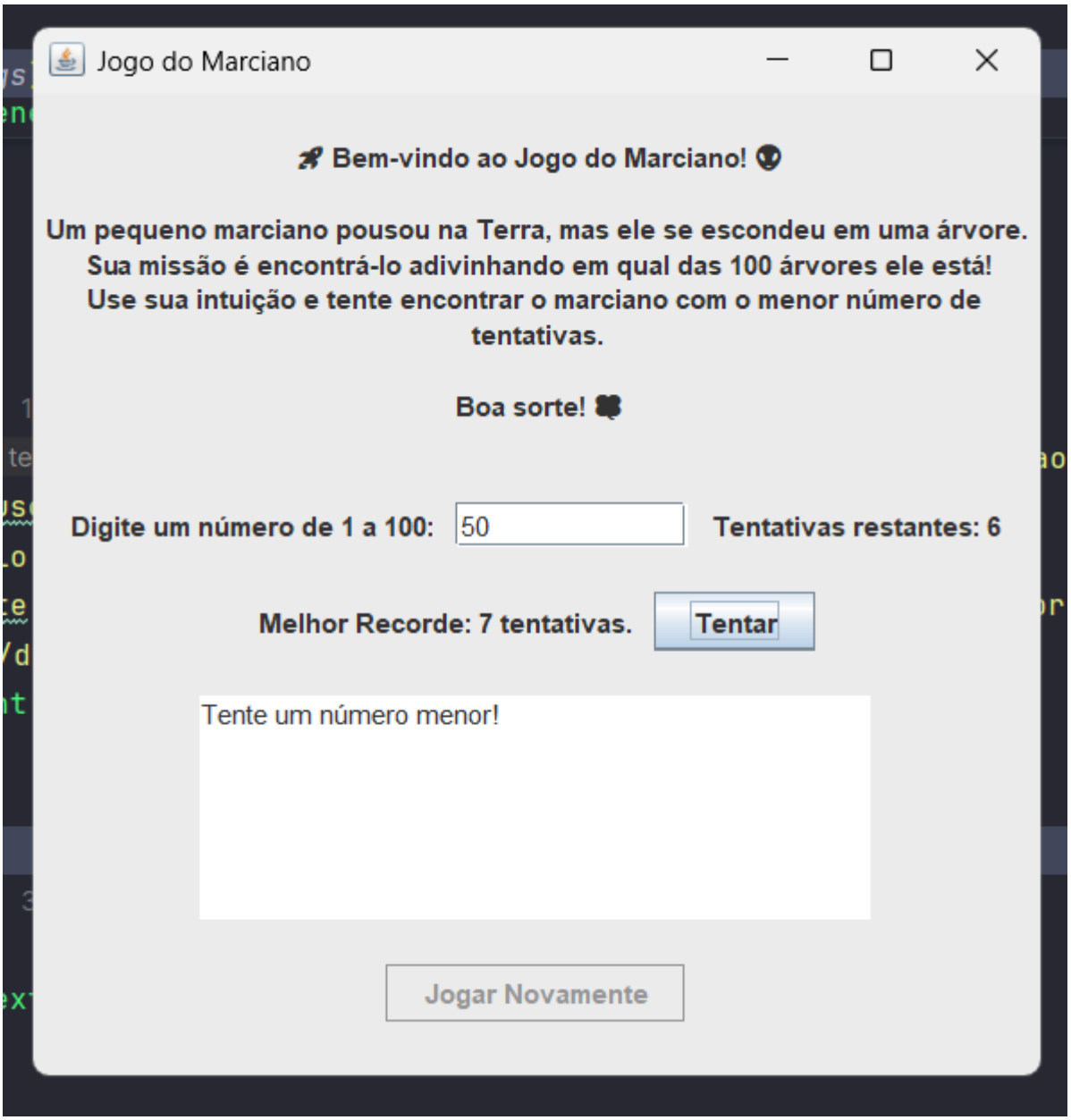
```
    winner = board[0][0];
    gameOver = true;
    return;
  }
  if (board[0][2] != 0 && board[0][2] == board[1][1] && board[1][1] == board[2][0]) {
    winner = board[0][2];
    gameOver = true;
    return;
  }
  boolean draw = true;
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      if (board[i][j] == 0) {
        draw = false;
      }
    }
  }
  if (draw) {
    gameOver = true;
    winner = 0;
  }
}

void displayWinner() {
  fill(139, 69, 19);
  rectMode(CENTER);
  rect(width / 2, height / 2, 220, 50);

  fill(245, 235, 220);
  textSize(24);
  textAlign(CENTER, CENTER);

  if (winner == 0) {
    text("Empate!", width / 2, height / 2);
  } else {
    text("Jogador " + (winner == 1 ? "X" : "O") + " venceu!", width / 2, height / 2);
  }
}


void resetBoard() {
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      board[i][j] = 0;
    }
  }
  gameOver = false;
  winner = 0;
```

```
  playerTurn = true;
}

void computerMove() {
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      if (board[i][j] == 0) {
        board[i][j] = 2;
        checkWinner();
        playerTurn = true;
        return;
      }
    }
  }
}

void drawModeSelection() {
  fill(139, 69, 19);
  rect(0, height / 4, width / 2, 100);
  rect(width / 2, height / 4, width / 2, 100);
  fill(245, 235, 220);
  textSize(22);
  textAlign(CENTER, CENTER);
  text("2 Jogadores", width / 4, height / 4 + 50);
  text("Contra PC", 3 * width / 4, height / 4 + 50);
}
```

3. Jogo do Marciano

- Um pequeno marciano pousou na Terra, mas ele se escondeu em uma árvore. Sua missão é encontrá-lo adivinhando em qual das 100 árvores ele está! Use sua intuição e tente encontrar o marciano com o menor número de tentativas.
- Inclui um contador de tentativas, mecânica de input do usuário e verificação dos números.

PRINTS:



CÓDIGO-FONTE:

```
import javax.swing.*;
import java.awt.*;
import java.util.Random;

public class JogoDoMarcianoGUI {

    private static final int maxTentativas = 7;
    private static int recorde = 7;
    private static int tentativas;
    private static int numeroMarciano;
```

```
public static void main(String[] args) {
    JFrame frame = new JFrame("Jogo do Marciano");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(500, 500);
    frame.setLayout(new BorderLayout());

    JPanel painel = new JPanel();
    painel.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 20));

    JLabel textoIntro = getJLabel();
    painel.add(textoIntro);

    JTextField entradaNumero = new JTextField();
    entradaNumero.setColumns(10);
    painel.add(new JLabel("Digite um número de 1 a 100:"));
    painel.add(entradaNumero);

    JLabel labelTentativas = new JLabel("Tentativas restantes: " + maxTentativas);
    JLabel labelRecorde = new JLabel("Melhor Recorde: " + recorde + " tentativas.");
    painel.add(labelTentativas);
    painel.add(labelRecorde);

    JButton btnTentar = new JButton("Tentar");
    painel.add(btnTentar);

    JTextArea resultado = new JTextArea(3, 30);
    resultado.setEditable(false);
    resultado.setLineWrap(true);
    resultado.setWrapStyleWord(true);
    resultado.setPreferredSize(new Dimension(300, 100));
    painel.add(resultado);

    JButton btnJogarNovamente = new JButton("Jogar Novamente");
    btnJogarNovamente.setEnabled(false);
    painel.add(btnJogarNovamente);

    frame.add(painel, BorderLayout.CENTER);

    frame.setVisible(true);

    btnTentar.addActionListener(e -> {
        if (tentativas == 0) {
            resetarJogo();
        }

        if (tentativas == 0) {
            numeroMarciano = new Random().nextInt(100) + 1;
        }
```

```
            String textoEntrada = entradaNumero.getText();
            try {
                int tentativa = Integer.parseInt(textoEntrada);
                if (tentativa < 1 || tentativa > 100) {
                    resultado.setText("Por favor, insira um número entre 1 e 100.");
                    return;
                }
                tentativas++;

                if (tentativa < numeroMarciano) {
                    resultado.setText("Tente um número maior!");
                } else if (tentativa > numeroMarciano) {
                    resultado.setText("Tente um número menor!");
                } else {
                    resultado.setText("🎉 Parabéns! Você encontrou o marciano na árvore " +
numeroMarciano + " em " + tentativas + " tentativas!");
                    if (tentativas < recorde) {
                        recorde = tentativas;
                        labelRecorde.setText("Melhor Recorde: " + recorde + " tentativas.");
                        resultado.append("\n🏆 Novo recorde!");
                    }
                    btnTentar.setEnabled(false);
                    btnJogarNovamente.setEnabled(true);
                }

                labelTentativas.setText("Tentativas restantes: " + (maxTentativas - tentativas));

                if (tentativas >= maxTentativas) {
                    resultado.append("\n😢 Você atingiu o número máximo de tentativas!");
                    btnTentar.setEnabled(false);
                    btnJogarNovamente.setEnabled(true);
                }

            } catch (NumberFormatException ex) {
                resultado.setText("Por favor, insira um número válido.");
            }
        });

        btnJogarNovamente.addActionListener(e -> {
            resetarJogo();
            resultado.setText("Boa sorte na próxima tentativa! 🍀");
            labelTentativas.setText("Tentativas restantes: " + maxTentativas);
            btnTentar.setEnabled(true);
            btnJogarNovamente.setEnabled(false);
        });

        resetarJogo();
```

```
    }

    private static JLabel getJLabel() {
        JLabel textoIntro = new JLabel("<html><div style='text-align: center;'>🚀 Bem-vindo ao
Jogo do Marciano! 👽<br><br>" +
                "Um pequeno marciano pousou na Terra, mas ele se escondeu em uma
árvore.<br>" +
                "Sua missão é encontrá-lo adivinhando em qual das 100 árvores ele está!<br>" +
                "Use sua intuição e tente encontrar o marciano com o menor número de
tentativas.<br><br>" +
                "Boa sorte! 🍀<br><br></div></html>");
        textoIntro.setHorizontalAlignment(SwingConstants.CENTER);
        return textoIntro;
    }

    private static void resetarJogo() {
        tentativas = 0;
        numeroMarciano = new Random().nextInt(100) + 1;
    }
}
```

Tecnologias Utilizadas:
- Processing.org (https://processing.org/) - Plataforma para desenvolvimento visual e interativo.
- Linguagem de programação baseada em Java.

Como Executar os Jogos:
1. Baixe e instale o Processing (https://processing.org/download/).
2. Clone este repositório:
   git clone https://github.com/DiegoRodrigues76/desenvolvimento_de_jogos.git
3. Abra o arquivo .pde de cada jogo no Processing.
4. Clique no botão Run para iniciar o jogo.

Integrantes da Equipe "2LM":
- Diego Henrique Rodrigues - 01650828
- Lucas Oliveira Carneiro - 01636600
- Lucas Vinicius França Aires - 01627405
- Monique Rafaela Carvalho Lopes - 01633424