

DG - RELATÓRIO DE ENTREGA - Trabalhos 1, 2 e 3 (Unidade 1)

- O representante do grupo deve enviar via "Chat" do TEAMS num link de acesso a um drive virtual com o material dos trabalhos (SEM COMPACTAR):

ARQUIVO 1: Relatório Técnico (Documento texto Word ou PDF) com:

- Nome dos integrantes do grupo em ordem alfabética;
- Colocar nesse relatório: "Prints" das telas dos Jogos (colocar quantas telas forem necessárias para registrar cada jogo), agrupados por jogo;
- Colocar no relatório, a cópia do código fonte de cada jogo.

Colocar um trabalho em cada pasta com todos os arquivos necessários para executar o jogo.

Nome do Game Studio: JET BEAVERS STUDIO

Alessandro Mathews Cardoso Dornelas Dos Santos - 01614625

Eduardo Rafael Silva Santos - 01602387

Gabryella Tainá Melo Silva - 01612684

Luis Gabriel da Silva Araújo - 01614692

Victhor Emanuel Soares Brito - 01615125

Wagner Vinícius Cassimiro da Silva – 01615748

Primeiro jogo: Jogo do Marciano, em Java.

```
=== O RESGATE DA PRINCESA CÔSMICA ===  
Há muito tempo, no sistema estelar de Auroria, a paz reinava sob o comando da Princesa Zayra...  
Até que um dia, o temível Lorde Xarkon atacou! Ele sequestrou a princesa e a escondeu em uma de suas 100 naves.  
Seu objetivo: descobrir em qual nave a princesa está antes que seja tarde demais!  
  
A missão começou! Você tem 10 tentativas para encontrar a nave certa.  
Digite o número da nave (1 a 100):
```

```
A missão começou! Você tem 10 tentativas para encontrar a nave certa.  
Digite o número da nave (1 a 100): 100  
A nave correta está em um número menor!  
Digite o número da nave (1 a 100): 50  
A nave correta está em um número menor!  
Digite o número da nave (1 a 100): 30  
A nave correta está em um número maior!  
Digite o número da nave (1 a 100): 43  
A nave correta está em um número menor!  
Digite o número da nave (1 a 100): 37  
A nave correta está em um número menor!  
Digite o número da nave (1 a 100): 32  
Parabéns, Capitão! Você encontrou a Princesa Zayra em 6 tentativas!  
  
* NOVO RECORDE! Você fez o resgate mais rápido até agora! *  
  
Deseja tentar novamente? (s/n):
```

Segundo jogo: Jogo Da Velha

Jogo da Velha

Jogar contra o computador

Jogar contra outro jogador

Você escolheu jogar contra outro jogador. X começa!

X		
	O	
	X	

Terceiro Jogo: Jogo Da Forca:

Categoria: Cidades

Tentativas restantes: 6

Letras erradas: []



Categoria: Países

Tentativas restantes: 6

Letras erradas: []




A u s t r a l i a

Você venceu!

Categoria: Animais

Tentativas restantes: 0

Letras erradas: [D, F, S, W, P, M]



_ a _ _ o _ _ o

Fim de Jogo! A palavra era: Cachorro

Reiniciar Jogo

Cópia dos códigos fontes:

Código fonte do Jogo Do Marciano:

```
package Jogo_marciano;
```

```
import java.util.Random;
```

```
import java.util.Scanner;
```

```
public class Game {
```

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

```
        Scanner scanner = new Scanner(System.in);
```

```
        Random random = new Random();
```

```
        int melhorTentativa = Integer.MAX_VALUE;
```

```
        boolean jogarNovamente;
```

```
        // Introdução com a história
```

```
        System.out.println("=== O RESGATE DA PRINCESA CÓSMICA ===");
```

```
        System.out.println("Há muito tempo, no sistema estelar de Auroria, a paz reinava  
sob o comando da Princesa Zayra...");
```

```
        System.out.println("Até que um dia, o temível Lorde Xarkon atacou! Ele  
sequestrou a princesa e a escondeu em uma de suas 100 naves.");
```

```
        System.out.println("Seu objetivo: descobrir em qual nave a princesa está antes que  
seja tarde demais!");
```

```
        do {
```

```
            int marcianoPosicao = random.nextInt(100) + 1;
```

```
            int tentativa;
```

```
            int tentativasMaximas = 10;
```

```
int tentativas = 0;

boolean acertou = false;

System.out.println("\nA missão começou! Você tem " + tentativasMaximas + "
tentativas para encontrar a nave certa.");

while (!acertou && tentativas < tentativasMaximas) {

    System.out.print("Digite o número da nave (1 a 100): ");

    while (!scanner.hasNextInt()) {

        System.out.print("Sério? Isso nem parece um número! Tenta de novo, mas
agora entre 1 e 100: ");

        scanner.next();

    }

    tentativa = scanner.nextInt();

    if (tentativa < 1 || tentativa > 100) {

        System.out.println("Número fora do intervalo! Digite um valor entre 1 e
100.");

        continue;

    }

    tentativas++;

    if (tentativa == marcianoPosicao) {

        System.out.println("Parabéns, Capitão! Você encontrou a Princesa Zayra
em " + tentativas + " tentativas!");

        acertou = true;
```

```

        if (tentativas < melhorTentativa) {

            melhorTentativa = tentativas;

            System.out.println("\n\u2B50 NOVO RECORDE! Você fez o resgate
            mais rápido até agora! \u2B50");

        }

        } else if (tentativa < marcianoPosicao) {

            System.out.println("A nave correta está em um número maior!");

        } else {

            System.out.println("A nave correta está em um número menor!");

        }

    }

    if (tentativas == tentativasMaximas && !acertou) {

        System.out.println("\nMissão falhou! Lorde Xarkon escapou e a Princesa
        Zayra estava na nave número " + marcianoPosicao + ".");

    }

}

System.out.print("\nDeseja tentar novamente? (s/n): ");

jogarNovamente = scanner.next().equalsIgnoreCase("s");

} while (jogarNovamente);

System.out.println("\nObrigado por jogar! Seu melhor recorde foi " +
(melhorTentativa == Integer.MAX_VALUE ? "-" : melhorTentativa) + " tentativas.");

scanner.close();

}

}

```

|Código fonte do Jogo Da Velha:

```
int[][] board = new int[3][3];

boolean gameStarted = false;

boolean playingWithComputer = false;

boolean playerX = true;

boolean gameOver = false;

int winner = 0;

String gameMessage = "Escolha uma opção";

boolean canGoBackToMenu = false;
```

```
int boardSize = 400;

int cellSize = boardSize / 3;
```

```
void setup() {

    size(600, 600);

    textAlign(CENTER, CENTER);

    textSize(32);

    drawMenu();

}
```

```
void draw() {

    background(235);

    if (!gameStarted) {

        drawMenu();

    } else {

        drawBoard();

    }

}
```

```
    displayMessage();

    if (gameOver) {

        drawBackButton();

    }

}

}
```

```
void drawMenu() {

    background(235);

    fill(0);

    textSize(50);

    text("Jogo da Velha", width / 2, height / 4 - 30);

    fill(70, 130, 180);

    noStroke();

    rect(width / 4, height / 2 - 40, width / 2, 50, 20);

    rect(width / 4, height / 2 + 50, width / 2, 50, 20);

    fill(255);

    textSize(20);

    text("Jogar contra o computador", width / 2, height / 2 - 15);

    text("Jogar contra outro jogador", width / 2, height / 2 + 73);

}
```

```
void drawBoard() {

    int startX = (width - boardSize) / 2;

    int startY = (height - boardSize) / 2;
```



```
for (int i = 0; i < 3; i++) {  
    for (int j = 0; j < 3; j++) {  
        float x = startX + i * cellSize;  
        float y = startY + j * cellSize;  
        if (board[i][j] == 1) {  
            fill(255, 0, 0);  
            textSize(64);  
            text("X", x + cellSize / 2, y + cellSize / 2);  
        } else if (board[i][j] == -1) {  
            fill(0, 0, 255);  
            textSize(64);  
            text("O", x + cellSize / 2, y + cellSize / 2);  
        }  
        stroke(0);  
        noFill();  
        rect(x, y, cellSize, cellSize);  
    }  
}  
}
```

```
void displayMessage() {  
    fill(0);  
    textSize(24);  
    text(gameMessage, width / 2, height - 550);  
}
```

```
}
```

```
void drawBackButton() {  
  
    fill(34, 139, 34);  
  
    noStroke();  
  
    rect(width / 4, height - 90, width / 2, 40, 20);  
  
    fill(255);  
  
    textAlign(CENTER, CENTER);  
  
    textSize(20);  
  
    text("Voltar ao Menu", width / 2, height - 70);  
  
}
```

```
void mousePressed() {  
  
    if (!gameStarted) {  
  
        if (mouseX > width / 4 && mouseX < width / 4 + width / 2 &&  
            mouseY > height / 2 - 30 && mouseY < height / 2 + 10) {  
  
            playingWithComputer = true;  
  
            gameStarted = true;  
  
            gameMessage = "Você jogará contra o computador. X começa!";  
  
            initializeBoard();  
  
        }  
  
        if (mouseX > width / 4 && mouseX < width / 4 + width / 2 &&  
            mouseY > height / 2 + 40 && mouseY < height / 2 + 80) {  
  
            playingWithComputer = false;  
  
            gameStarted = true;  

```

```

    gameMessage = "Você escolheu jogar contra outro jogador. X começa!";

    initializeBoard();

}

} else if (gameOver && canGoBackToMenu) {

    if (mouseX > width / 4 && mouseX < width / 4 + width / 2 &&
        mouseY > height - 90 && mouseY < height - 50) {

        gameStarted = false;

        gameOver = false;

        canGoBackToMenu = false;

        gameMessage = "Escolha uma opção";

        initializeBoard();

        redraw();

    }

} else {

    int i = floor((mouseX - (width - boardSize) / 2) / cellSize);

    int j = floor((mouseY - (height - boardSize) / 2) / cellSize);

    if (board[i][j] == 0) {

        board[i][j] = playerX ? 1 : -1;

        playerX = !playerX;

        checkWinner();

        if (!gameOver && playingWithComputer && !playerX) {

            computerMove();

            checkWinner();

            playerX = !playerX;

```

```
    }  
    redraw();  
}  
}  
}
```

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

```
    if (abs(board[0][0] + board[1][1] + board[2][2]) == 3) {  
        gameOver = true;  
        winner = board[0][0];  
    }  
    if (abs(board[0][2] + board[1][1] + board[2][0]) == 3) {  
        gameOver = true;  
        winner = board[0][2];  
    }
```

```
}
```

```
boolean full = true;
```

```
for (int i = 0; i < 3; i++) {
```

```
    for (int j = 0; j < 3; j++) {
```

```
        if (board[i][j] == 0) {
```

```
            full = false;
```

```
        }
```

```
    }
```

```
}
```

```
if (full && !gameOver) {
```

```
    gameOver = true;
```

```
    winner = 2;
```

```
}
```

```
if (gameOver) {
```

```
    gameMessage = winner == 1 ? "Jogador X venceu!" : (winner == -1 ? "Jogador O  
venceu!" : "Empate!");
```

```
    canGoBackToMenu = true;
```

```
    redraw();
```

```
}
```

```
}
```

```
void initializeBoard() {
```

```
    for (int i = 0; i < 3; i++) {
```

```
        for (int j = 0; j < 3; j++) {
```

```
        board[i][j] = 0;

    }

}

playerX = true;

gameOver = false;

winner = 0;

canGoBackToMenu = false;

redraw();

}
```

```
void computerMove() {

    int i, j;

    do {

        i = int(random(3));

        j = int(random(3));

    } while (board[i][j] != 0);

    board[i][j] = -1;

}
```

Código fonte Jogo Da Forca:

```
import java.util.HashSet;
```

```
String[][] categorias = {

    {"Frutas", "Banana", "Maca", "Laranja", "Morango", "Uva", "Abacaxi", "Kiwi",
    "Melancia", "Cabeluda"},

    {"Animais", "Cachorro", "Gato", "Elefante", "Tigre", "Leao", "Girafa", "Zebra",
    "Rato", "Macaco"},
```

```
{ "Cidades", "Sao Paulo", "Rio de Janeiro", "Salvador", "Recife", "Brasilia", "Porto Alegre", "Curitiba", "Fortaleza", "Manaus"},
```

```
{ "Veiculos", "Carro", "Moto", "Bicicleta", "Onibus", "Aviao", "Navio", "Trem", "Helicoptero", "Caminhao"},
```

```
{ "Esportes", "Futebol", "Basquete", "Volei", "Natacao", "Handebol", "Golfe", "Boxe", "Rugby", "Xadrez", "Corrida"},
```

```
{ "Comidas", "Pizza", "Hamburguer", "Macarrao", "Sushi", "Feijoada", "Lasanha", "Salada", "Churrasco", "Arroz", "Bife"},
```

```
{ "Países", "Brasil", "Argentina", "Franca", "Italia", "Espanha", "Alemanha", "Estados Unidos", "Japao", "Canada", "Australia" }
```

```
};
```

```
String categoriaEscolhida;
```

```
String palavraEscolhida;
```

```
String palavraOculta;
```

```
String dica;
```

```
int tentativas = 6;
```

```
ArrayList<Character> letrasErradas;
```

```
HashSet<Character> letrasTentadas;
```

```
boolean jogoAtivo;
```

```
boolean jogadorVenceu;
```

```
int tempoReinicio = 0;
```

```
void setup() {
```

```
    size(800, 600);
```

```
    frameRate(60);
```

```
    iniciarJogo();
```

```
}
```

```
void draw() {  
  
    background(245);  
  
  
    textSize(24);  
  
    fill(50);  
  
    textAlign(LEFT);  
  
    text(dica, 30, 50);  
  
  
    mostrarPalavraOculta();  
  
  
    textSize(20);  
  
    fill(200, 0, 0);  
  
    text("Tentativas restantes: " + tentativas, 30, 120);  
  
  
    fill(0);  
  
    text("Letras erradas: " + letrasErradas.toString().toUpperCase(), 30, 150);  
  
  
    desenharForca();  
  
  
    if (!jogoAtivo) {  
  
        fill(0);  
  
        textSize(28);  
  
        textAlign(CENTER, CENTER);
```



```

if (jogadorVenceu) {

    fill(0, 200, 0);

    text("Você venceu!", width / 2, height / 2 + 200);

    if (millis() - tempoReinicio > 3000) {

        iniciarJogo();

    }

} else {

    fill(200, 0, 0);

    text("Fim de Jogo! A palavra era: " + palavraEscolhida, width / 2, height / 2 + 200);

    desenharBotaoReiniciar();

}

}

}

void keyPressed() {

    if (jogoAtivo && (key >= 'a' && key <= 'z' || key >= 'A' && key <= 'Z')) {

        char letra = Character.toLowerCase(key);

        if (letrasTentadas.contains(letra)) {

            return;

        }

        letrasTentadas.add(letra);

        if (palavraEscolhida.toLowerCase().contains(String.valueOf(letra))) {

```

```

    atualizarPalavraOculta(letra);

    if (palavraOculta.equalsIgnoreCase(palavraEscolhida)) {

        jogoAtivo = false;

        jogadorVenceu = true;

        tempoReinicio = millis();

    }

    } else {

        letrasErradas.add(Character.toUpperCase(letra));

        tentativas--;

        if (tentativas <= 0) {

            jogoAtivo = false;

            jogadorVenceu = false;

        }

    }

}

void mousePressed() {

    if (!jogoAtivo && !jogadorVenceu &&

        mouseX > width - 200 && mouseX < width - 50 &&

        mouseY > height - 60 && mouseY < height - 20) {

        iniciarJogo();

    }

}

```

```
void iniciarJogo() {  
  
    int categoriaIndex = int(random(categorias.length));  
  
    String[] categoria = categorias[categoriaIndex];  
  
    categoriaEscolhida = categoria[0];  
  
    dica = "Categoria: " + categoriaEscolhida;  
  
  
    int palavraIndex = int(random(1, categoria.length));  
  
    palavraEscolhida = categoria[palavraIndex];  
  
    palavraOcultada = "";  
  
  
    for (int i = 0; i < palavraEscolhida.length(); i++) {  
  
        if (palavraEscolhida.charAt(i) == ' ') {  
  
            palavraOcultada += " ";  
  
        } else {  
  
            palavraOcultada += "_";  
  
        }  
  
    }  
  
  
  
    letrasErradas = new ArrayList<Character>();  
  
    letrasTentadas = new HashSet<Character>();  
  
    tentativas = 6;  
  
    jogoAtivo = true;  
  
    jogadorVenceu = false;  
  
}
```

```

void mostrarPalavraOculta() {

    textSize(48);

    fill(0);

    float x = width / 2 - (textWidth(palavraOculta.replace(" ", "_")) / 2);

    for (int i = 0; i < palavraOculta.length(); i++) {

        char c = palavraOculta.charAt(i);

        if (c == ' ') {

            // Não mostra nada para espaços

        } else if (c == '_') {

            text("_", x + i * 30, height / 2 + 30);

        } else {

            text(c, x + i * 30, height / 2 + 30);

        }

    }

}

void atualizarPalavraOculta(char letra) {

    StringBuilder sb = new StringBuilder(palavraOculta);

    for (int i = 0; i < palavraEscolhida.length(); i++) {

        if (Character.toLowerCase(palavraEscolhida.charAt(i)) == letra) {

            sb.setCharAt(i, palavraEscolhida.charAt(i));

        }

    }

    palavraOculta = sb.toString();

```

```
}
```

```
void desenharForca() {
```

```
    float baseX = width / 9;
```

```
    float baseY = height / 1.7;
```

```
    float largura = 8;
```

```
    stroke(0);
```

```
    strokeWeight(largura);
```

```
    line(baseX, baseY + 200, baseX, baseY - 120);
```

```
    line(baseX, baseY - 120, baseX + 80, baseY - 120);
```

```
    line(baseX + 80, baseY - 120, baseX + 80, baseY - 40);
```

```
    if (tentativas <= 5) ellipse(baseX + 80, baseY - 40, 30, 30);
```

```
    if (tentativas <= 4) line(baseX + 80, baseY - 20, baseX + 80, baseY + 40);
```

```
    if (tentativas <= 3) line(baseX + 80, baseY - 15, baseX + 60, baseY + 20);
```

```
    if (tentativas <= 2) line(baseX + 80, baseY - 15, baseX + 100, baseY + 20);
```

```
    if (tentativas <= 1) line(baseX + 80, baseY + 40, baseX + 60, baseY + 80);
```

```
    if (tentativas <= 0) line(baseX + 80, baseY + 40, baseX + 100, baseY + 80);
```

```
}
```

```
void desenharBotaoReiniciar() {
```

```
fill(255, 120, 120);

noStroke();

rect(width - 200, height - 60, 150, 40, 40);


fill(255);

textSize(18);

textAlign(CENTER, CENTER);

text("Reiniciar Jogo", width - 125, height - 40);

}
```