Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №4\_2**

**Дисциплина: Платформо-независимое программирование**

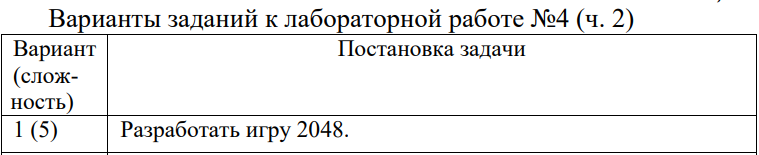
Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Агаджанян

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. И. Шиян

**Тема**: СОБЫТИЯ И ИХ ОБРАБОТКА

**Задание**:



**Листинг:**

import java.awt.\*;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.util.Random;

import javax.swing.\*;

import java.util.ArrayList;

import java.util.List;

public class Game2048 extends JPanel {

    private static final Color BG\_COLOR = new Color(0xbbada0);

    private static final String FONT\_NAME = "Arial";

    private static final int TILE\_SIZE = 64;

    private static final int TILE\_MARGIN = 16;

    private int[][] board;

    private int size = 4;

    private boolean gameOver = false;

    private boolean gameWon = false;

    private int score = 0;

    private Random random = new Random();

    private List<FireworkParticle> fireworks = new ArrayList<>();

    public Game2048() {

        setPreferredSize(new Dimension(size \* (TILE\_SIZE + TILE\_MARGIN) + TILE\_MARGIN,

                size \* (TILE\_SIZE + TILE\_MARGIN) + TILE\_MARGIN));

        setFocusable(true);

        addKeyListener(new KeyAdapter() {

            @Override

            public void keyPressed(KeyEvent e) {

                if (gameOver) {

                    if (e.getKeyCode() == KeyEvent.VK\_SPACE) {

                        resetGame();

                    }

                    return;

                }

                switch (e.getKeyCode()) {

                    case KeyEvent.VK\_UP:

                        move(Direction.UP);

                        break;

                    case KeyEvent.VK\_DOWN:

                        move(Direction.DOWN);

                        break;

                    case KeyEvent.VK\_LEFT:

                        move(Direction.LEFT);

                        break;

                    case KeyEvent.VK\_RIGHT:

                        move(Direction.RIGHT);

                        break;

                }

                repaint();

            }

        });

        resetGame();

    }

    private void resetGame() {

        board = new int[size][size];

        gameOver = false;

        gameWon = false;

        score = 0;

        fireworks.clear();

        addRandomTile();

        addRandomTile();

        repaint();

    }

    private void addRandomTile() {

        int row, col;

        do {

            row = random.nextInt(size);

            col = random.nextInt(size);

        } while (board[row][col] != 0);

        board[row][col] = (random.nextDouble() < 0.9 ? 2 : 4);

    }

    private void move(Direction direction) {

        boolean moved = false;

        switch (direction) {

            case UP:

                moved = moveUp();

                break;

            case DOWN:

                moved = moveDown();

                break;

            case LEFT:

                moved = moveLeft();

                break;

            case RIGHT:

                moved = moveRight();

                break;

        }

        if (moved) {

            addRandomTile();

            checkGameOver();

            checkForWin();

        }

    }

    private boolean moveUp() {

        boolean moved = false;

        for (int j = 0; j < size; j++) {

            for (int i = 1; i < size; i++) {

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

                    int row = i;

                    while (row > 0 && board[row - 1][j] == 0) {

                        board[row - 1][j] = board[row][j];

                        board[row][j] = 0;

                        row--;

                        moved = true;

                    }

                    if (row > 0 && board[row - 1][j] == board[row][j]) {

                        board[row - 1][j] \*= 2;

                        score += board[row - 1][j];

                        board[row][j] = 0;

                        moved = true;

                    }

                }

            }

        }

        return moved;

    }

    private boolean moveDown() {

        boolean moved = false;

        for (int j = 0; j < size; j++) {

            for (int i = size - 2; i >= 0; i--) {

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

                    int row = i;

                    while (row < size - 1 && board[row + 1][j] == 0) {

                        board[row + 1][j] = board[row][j];

                        board[row][j] = 0;

                        row++;

                        moved = true;

                    }

                    if (row < size - 1 && board[row + 1][j] == board[row][j]) {

                        board[row + 1][j] \*= 2;

                        score += board[row + 1][j];

                        board[row][j] = 0;

                        moved = true;

                    }

                }

            }

        }

        return moved;

    }

    private boolean moveLeft() {

        boolean moved = false;

        for (int i = 0; i < size; i++) {

            for (int j = 1; j < size; j++) {

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

                    int col = j;

                    while (col > 0 && board[i][col - 1] == 0) {

                        board[i][col - 1] = board[i][col];

                        board[i][col] = 0;

                        col--;

                        moved = true;

                    }

                    if (col > 0 && board[i][col - 1] == board[i][col]) {

                        board[i][col - 1] \*= 2;

                        score += board[i][col - 1];

                        board[i][col] = 0;

                        moved = true;

                    }

                }

            }

        }

        return moved;

    }

    private boolean moveRight() {

        boolean moved = false;

        for (int i = 0; i < size; i++) {

            for (int j = size - 2; j >= 0; j--) {

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

                    int col = j;

                    while (col < size - 1 && board[i][col + 1] == 0) {

                        board[i][col + 1] = board[i][col];

                        board[i][col] = 0;

                        col++;

                        moved = true;

                    }

                    if (col < size - 1 && board[i][col + 1] == board[i][col]) {

                        board[i][col + 1] \*= 2;

                        score += board[i][col + 1];

                        board[i][col] = 0;

                        moved = true;

                    }

                }

            }

        }

        return moved;

    }

    private void checkGameOver() {

        for (int i = 0; i < size; i++) {

            for (int j = 0; j < size; j++) {

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

                    return;

                }

            }

        }

        for (int i = 0; i < size; i++) {

            for (int j = 0; j < size; j++) {

                if (i > 0 && board[i][j] == board[i - 1][j]) return;

                if (i < size - 1 && board[i][j] == board[i + 1][j]) return;

                if (j > 0 && board[i][j] == board[i][j - 1]) return;

                if (j < size - 1 && board[i][j] == board[i][j + 1]) return;

            }

        }

        gameOver = true;

    }

    private void checkForWin() {

        for (int i = 0; i < size; i++) {

            for (int j = 0; j < size; j++) {

                if (board[i][j] == 8) {

                    gameWon = true;

                    gameOver = true;

                    launchFireworks(50);

                    return;

                }

            }

        }

    }

    private void launchFireworks(int count) {

        for (int i = 0; i < count; i++) {

            int startX = getWidth() / 2 + random.nextInt(100) - 50;

            int startY = getHeight() / 2 + random.nextInt(100) - 50;

            Color color = new Color(random.nextInt(256), random.nextInt(256), random.nextInt(256));

            fireworks.add(new FireworkParticle(startX, startY, color));

        }

        Timer timer = new Timer(25, e -> {

            updateFireworks();

            repaint();

            if (fireworks.isEmpty()) {

                ((Timer) e.getSource()).stop();

            }

        });

        timer.start();

    }

    private void updateFireworks() {

        List<FireworkParticle> toRemove = new ArrayList<>();

        for (FireworkParticle firework : fireworks) {

            firework.update();

            if (firework.isDead()) {

                toRemove.add(firework);

            }

        }

        fireworks.removeAll(toRemove);

    }

    private Color getTileColor(int value) {

        switch (value) {

            case 2:    return new Color(0xeee5da);

            case 4:    return new Color(0xede0c8);

            case 8:    return new Color(0xf2b179);

            case 16:   return new Color(0xf59563);

            case 32:   return new Color(0xf67c5f);

            case 64:   return new Color(0xf65e3b);

            case 128:  return new Color(0xedcf72);

            case 256:  return new Color(0xedcc61);

            case 512:  return new Color(0xedc850);

            case 1024: return new Color(0xedc53f);

            case 2048: return new Color(0xedc22e);

            default:   return new Color(0xcdc1b4);

        }

    }

    private Color getTileFontColor(int value) {

        return value < 16 ? new Color(0x776e65) : new Color(0xf9f6f2);

    }

    @Override

    public void paint(Graphics g) {

        super.paint(g);

        Graphics2D g2d = (Graphics2D) g;

        g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

        g.setColor(BG\_COLOR);

        g.fillRect(0, 0, this.getWidth(), this.getHeight());

        for (int row = 0; row < size; row++) {

            for (int col = 0; col < size; col++) {

                int value = board[row][col];

                int x = col \* (TILE\_SIZE + TILE\_MARGIN) + TILE\_MARGIN;

                int y = row \* (TILE\_SIZE + TILE\_MARGIN) + TILE\_MARGIN;

                g.setColor(getTileColor(value));

                g.fillRoundRect(x, y, TILE\_SIZE, TILE\_SIZE, 14, 14);

                g.setColor(getTileFontColor(value));

                if (value != 0) {

                    String s = String.valueOf(value);

                    Font font = new Font(FONT\_NAME, Font.BOLD, 36);

                    g.setFont(font);

                    int stringWidth = g.getFontMetrics().stringWidth(s);

                    int stringHeight = g.getFontMetrics().getAscent();

                    g.drawString(s, x + (TILE\_SIZE - stringWidth) / 2,

                            y + TILE\_SIZE - (TILE\_SIZE - stringHeight) / 2 - 4);

                }

            }

        }

        g.setFont(new Font(FONT\_NAME, Font.BOLD, 24));

        g.setColor(new Color(0x776e65));

        g.drawString("Score: " + score, TILE\_MARGIN, this.getHeight() - 15);

        if (gameWon) {

            g.setColor(new Color(100, 255, 100, 100));

            g.fillRect(0, 0, getWidth(), getHeight());

            g.setColor(new Color(0x776e65));

            g.setFont(new Font(FONT\_NAME, Font.BOLD, 48));

            String s = "You Win!";

            int stringWidth = g.getFontMetrics().stringWidth(s);

            g.drawString(s, getWidth() / 2 - stringWidth / 2, getHeight() / 2 - 48);

            g.setFont(new Font(FONT\_NAME, Font.BOLD, 24));

            s = "Press SPACE to restart";

            stringWidth = g.getFontMetrics().stringWidth(s);

            g.drawString(s, getWidth() / 2 - stringWidth / 2, getHeight() / 2 + 24);

        } else if (gameOver) {

            g.setColor(new Color(255, 100, 100, 100));

            g.fillRect(0, 0, getWidth(), getHeight());

            g.setColor(new Color(0x776e65));

            g.setFont(new Font(FONT\_NAME, Font.BOLD, 48));

            String s = "Game Over!";

            int stringWidth = g.getFontMetrics().stringWidth(s);

            g.drawString(s, getWidth() / 2 - stringWidth / 2, getHeight() / 2 - 48);

            g.setFont(new Font(FONT\_NAME, Font.BOLD, 24));

            s = "Press SPACE to restart";

            stringWidth = g.getFontMetrics().stringWidth(s);

            g.drawString(s, getWidth() / 2 - stringWidth / 2, getHeight() / 2 + 24);

        }

        for (FireworkParticle firework : fireworks) {

            firework.draw(g2d);

        }

    }

    public static void main(String[] args) {

        JFrame game = new JFrame();

        game.setTitle("2048 Game");

        game.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        game.setSize(340, 400);

        game.setResizable(false);

        game.add(new Game2048());

        game.setLocationRelativeTo(null);

        game.setVisible(true);

    }

    private enum Direction {

        UP, DOWN, LEFT, RIGHT

    }

    private class FireworkParticle {

        private double x, y;

        private double vx, vy;

        private Color color;

        private int lifespan = 150;

        public FireworkParticle(int x, int y, Color color) {

            this.x = x;

            this.y = y;

            this.color = color;

            double angle = random.nextDouble() \* 2 \* Math.PI;

            double speed = random.nextDouble() \* 5 + 2;

            this.vx = Math.cos(angle) \* speed;

            this.vy = Math.sin(angle) \* speed -2;

        }

        public void update() {

            x += vx;

            y += vy;

            vy += 0.1;

            lifespan--;

        }

        public boolean isDead() {

            return lifespan <= 0;

        }

        public void draw(Graphics2D g) {

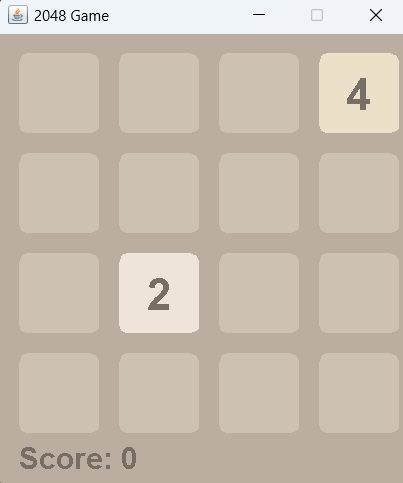
            g.setColor(color);

            g.fillRect((int) x, (int) y, 5, 5);

        }

    }

}

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Обработка события «Нажать кнопку вверх»

