Министерство образования и науки РФ

ФГБОУ ВО «Омский государственный технический университет»

Кафедра «ИВТ»

Расчетно-графическая работа

по дисциплине «Проектирование и тестирование программного обеспечения»

на тему: «Разработка Desktop-приложения на языке высокого уровня»

Выполнил:

ст. гр. ПИН-211

Сероухов Е.С.

Принял:

ст. пр.

Карабцов Р.Д.

Омск 2022

СОДЕРЖАНИЕ

Введение

В рамках самостоятельной работы необходимо разработать Desktop-приложение по одной из предложенных спецификаций.

Мною была выбрана тема “Игра Pacman”. Игра была реализована по следующим правилам:

1. Чтобы выиграть, нужно съесть всю еду на уровне.

2. Пакман проигрывает, если его догнал призрак

3. Пакман и призраки не могут ходить сквозь стены и останавливаются при столкновении с ней.

Были реализованы следующие функции:

1. При нажатии на кнопку Новая игра предлагается выбрать уровень сложности. Если не выбрать уровень сложности, то откроется игра без призраков (пасхалка).

2. В игре 4 уровня сложности. На 1- один призрак, на 2 – два, и т.д.

3. Если игрок выигрывает, то сохраняется запись о прохождении уровня сложности в текущей игровой сессии.

4. Пакман анимирован. Он может открывать и закрывать рот, когда движется.

5. Сохранение/загрузку игры с файла (достижения).

6. Раздел “Советы”

7. Несколько видов ботов.

1. Аспекты реализвации.

1.1 Структура проекта.

Для создания игры были созданы следующие классы:

- PacmanMenu – главный класс с меню, наследующий JFrame.
- PacmanTips – класс окна с советами по прохождению, наследующий JFrame.
- Game – класс окна игры, наследующий JFrame.
- Pacman – класс пакмана. В экземпляре этого класса будут хранится координаты и направление движения.
- Ghost - класс призраков. В экземплярах этого класса будут хранится координаты и направление движения.

Далее подробнее расскажу о аспектах реализации в каждом классе.

1.2 PacmanMenu

Кнопки реализованы с помощью элементов JButton. Диалоговые окна с помощью JOptionPane.showOptionDialog и JOptionPane.showInputDialog.

Сохранение достижений игроков осуществляется сохранением соответствующих записей в txt файле. При открытии приложения они считываются из него и выводятся на экран с помощью элемента JLabel.

1.3 PacmanTips

Текст советов выводится на экран с помощью элемента JLabel.

1.4 Game

Карта лабиринта представлена в виде двумерного массива строк. Элементы массива могут принимать значения “ EMPTY”, “WALL” и “FOOD”. Пусто, стена и еда соответственно. Логика движений призраков и правила перемещения пакмана и призраков осуществляются в данном классе.

Кратко расскажу об алгоритме передвижения призраков: при остановке объект призрака оценивает в каком направлении целесообразней двигаться дальше. Рассчитывается разница суммы координат при движении вправо, влево,

вверх и вниз. Если в выбранном направлении нельзя пройти (стена), то выбирается следующие по приоритету.

1.5 Pacman

В этом классе хранятся координаты и направления движения пакмана, так же реализованы геттеры и сеттеры и метод перемещения, изменяющий координаты.

1.6 Ghost

Хранятся координаты и направления движения призрака, так же реализованы геттеры и сеттеры и метод перемещения, изменяющий координаты.

2. Тестирование приложения

При запуске приложения открывается окно с меню и диалоговое окно, предлагающее ввести никнейм игрока (рисунок 1).

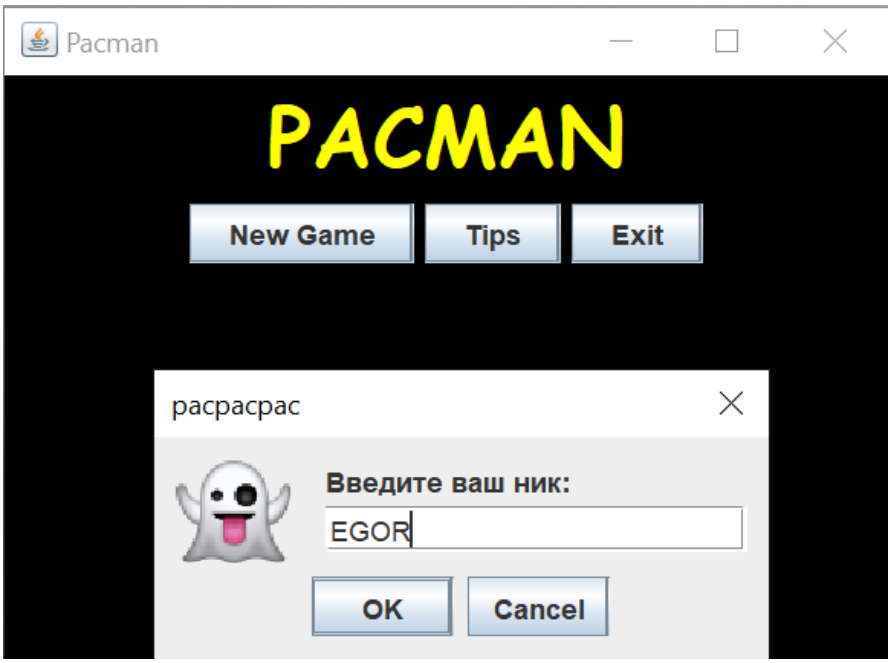

Рисунок 1. – запуск программы и ввод никнейма

Далее, если нажать на кнопку “Tips” откроется окно с советами по прохождению уровней (рисунок 2).

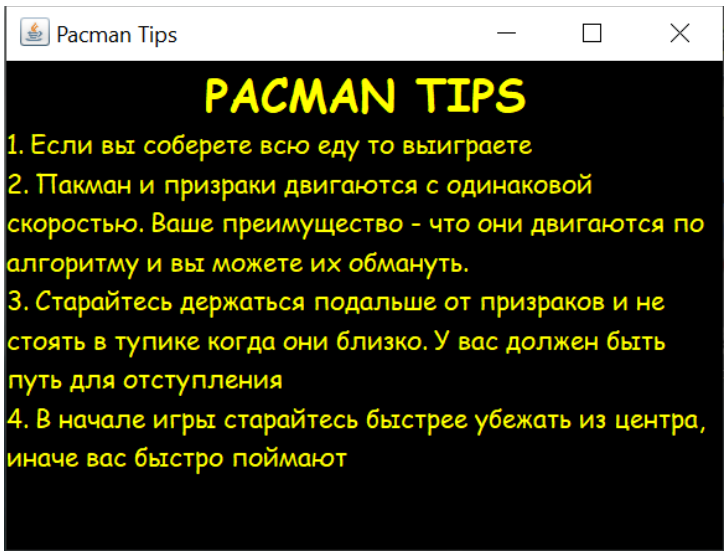

Рисунок 2. – окно советов.

Если нажать на кнопку “Exit”, то приложение завершит работу.

При нажатии на кнопку “New Game” откроется диалоговое окно для выбора уровня сложности. Пасхалка: если не выбрать уровень сложности, то откроется секретный уровень без призраков.

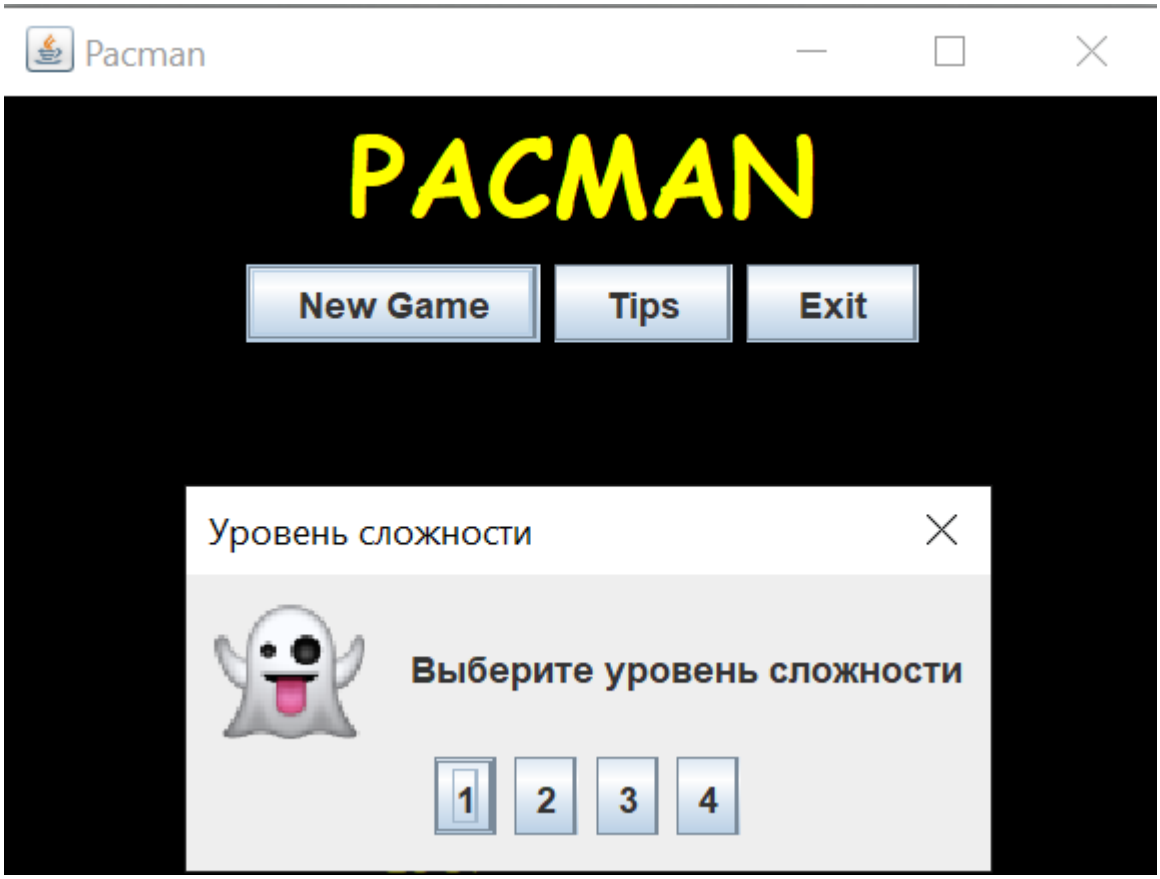

Рисунок 3. – выбор сложности.

После выбора уровня сложности откроется окно с новой игрой. На игровом окне появится соответствующие уровню сложности количество призраков (1, 2, 3 или 4 призрака) (рисунок 4).

Рисунок 4. – окно с игрой.

Игра может закончится двумя вариантами:

А) призраки поймают пакмана раньше, чем тот съест всю еду (процент съеденной еды показывается в нижней части окна на рисунке 4) (рисунок 5);

Б) пакман съест всю еду раньше, чем призраки его поймают (рисунок 6).

Рисунок 5. – пакман проиграл, вариант А.

Рисунок 6. – пакман выиграл, вариант Б.

При победе пакмана запись о прохождении уровня сложности записывается к никнейму, который игрок указал при открытии приложения (рисунок 7).

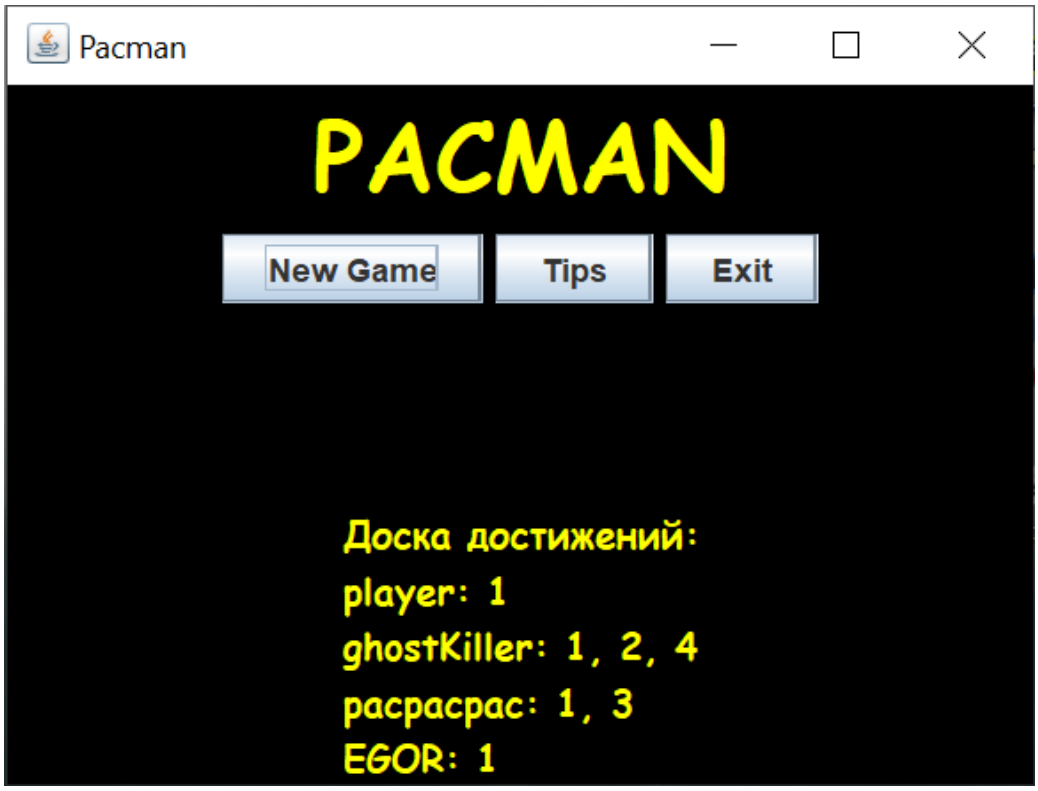

Рисунок 7. – доска достижений.

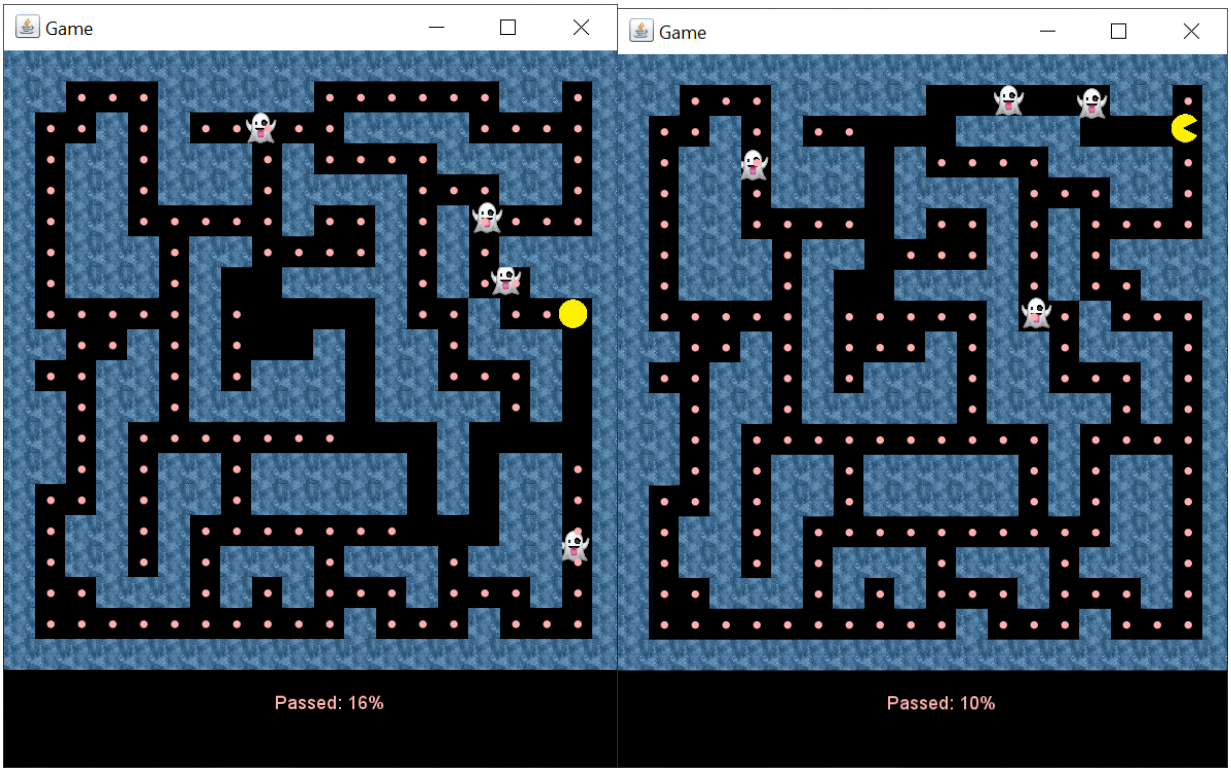

Рисунок 8. – опасные зоны.

Выводы

В ходе проделанной работы были получен опыт и отточены навыки разработки Desktop-приложения на языке программирования Java. В ходе тестирования были найдены особенности получившегося приложения и сделаны выводы.

Карта спроектирована так, что у призраков есть “слепые” места, в которые они, в силу не идеальности алгоритма передвижения, редко заходят. Зная это, игроку легче проходить игру, но в тоже время, если он не будет перемещаться и “отсижеваться” в этих слепых зонах, то никогда не пройдет игру. На карте также есть опасные зоны, в которых призраки могут легко окружить и догнать пакмана (рисунок 8). Это центр, в котором пакман появляется, и правый верхний угол карты.

Алгоритм призраков не идеален, т.к. на игровом поле присутствуют стены и путь, по которому пойдет призрак может оказаться тупиковым, но в рамках расчетно-графической работы я посчитал целесообразным не использовать графовые алгоритмы нахождения кратчайшего пути. Даже без алгоритма кратчайшего пути, поведение призраков достаточно “умное”, чтобы игроку было относительно сложно проходить игру на третьем и четвертом уровне, когда его догоняют три или четыре призрака.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Справка Java. URL: https://www.java.com/ru/download/help/index.html
2. JWindow, JFrame, JDialog. URL: http://java-online.ru/swing-windows.xhtml
3. Обзор Gson . URL: http://www.javenue.info/post/gson-json-api
4. BufferedReader и BufferedWriter Класс. URL: https://javarush.ru/groups/posts/593-bufferedreader-i-bufferedwritter
5. События и слушатели. URL: http://java-online.ru/java-listener.xhtml
6. Таймер – классы Timer и TimerTask. URL: http://developer.alexanderklimov.ru/android/java/timer.php
7. Принципы ООП. URL: https://javarush.ru/groups/posts/principy-oop
8. Диалоговые окна JOptionPane. URL: http://java-online.ru/swing-joptionpane.xhtml
9. Использование JPanel контейнера панели. URL: https://javaswing.wordpress.com/2009/07/19/using-jpanel/
10. Делаем главное меню с помощью JMenuBar. Основа форм. URL: https://javaswing.wordpress.com/2010/02/20/jmenubar/

ПРИЛОЖЕНИЕ А
(обязательное)

Код основных классов программы:

PacmanMenu.java :

```
import javax.swing.*;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

public class PacmanMenu extends JFrame implements ActionListener {

    private JButton newGameButton;
    private JButton tipsButton;
    private JButton exitButton;
    private JLabel usersLabel;

    private ImageIcon icon_ghost;
    public PacmanMenu() throws IOException {
        loadImages();
        setTitle("Pacman");
        setSize(400, 300);
        setLocationRelativeTo(null); // центрируем окно на экране
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel menuPanel = new JPanel();
        menuPanel.setBackground(Color.BLACK); // задаем цвет фона
        menuPanel.setLayout(new BorderLayout());

        JLabel title = new JLabel("PACMAN");
        title.setFont(new Font("Comic Sans MS", Font.BOLD, 36)); // задаем стиль
шрифта
        title.setForeground(Color.YELLOW); // задаем цвет текста
        title.setHorizontalAlignment(JLabel.CENTER); // выравнивание по центру
        menuPanel.add(title, BorderLayout.NORTH);

        JPanel buttonsPanel = new JPanel();
        buttonsPanel.setBackground(Color.BLACK); // задаем цвет фона
        newGameButton = new JButton("New Game");
        newGameButton.addActionListener(this);
        tipsButton = new JButton("Tips");
        tipsButton.addActionListener(this);
        exitButton = new JButton("Exit");
        exitButton.addActionListener(this);
        buttonsPanel.add(newGameButton);
        buttonsPanel.add(tipsButton);
        buttonsPanel.add(exitButton);
        menuPanel.add(buttonsPanel, BorderLayout.CENTER);

        usersLabel = new JLabel("");
        usersLabel.setFont(new Font("Comic Sans MS", Font.BOLD, 14)); // задаем
стиль шрифта
        usersLabel.setForeground(Color.YELLOW); // задаем цвет текста
        usersLabel.setHorizontalAlignment(JLabel.CENTER); // выравнивание по
```

```
центру

        update_usersLabel();
        menuPanel.add(usersLabel, BorderLayout.SOUTH);


        add(menuPanel);
        setVisible(true); // отображаем окно на экране
        String cur_user = getUserName(icon_ghost);
        cur_user+=":";
        update_usersLabel();
        try (PrintWriter out = new PrintWriter(new BufferedWriter(new
FileWriter("players.txt", true)))) {
            out.print(cur_user);
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
    public void update_usersLabel() throws IOException {
        List<String> usersList = readUserNamesToList();
        StringBuilder sb = new StringBuilder();
        sb.append("<html>Доска достижений:<br>");
        for (String element : usersList) {
            sb.append(element).append("<br>");
        }
        sb.append("</html>");
        String html = sb.toString();

        usersLabel.setText(html);
    }
    public static List<String> readUserNamesToList() throws IOException {
        List<String> lines = new ArrayList<>();

        try (
                BufferedReader reader = new BufferedReader(new
InputStreamReader(Objects.requireNonNull(PacmanMenu.class.getResourceAsStream("/pl
ayers.txt"))))) {
            String line;
            while ((line = reader.readLine()) != null) {
                lines.add(line);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return lines;
    }


    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == newGameButton) {
            int level = chooseLevel(icon_ghost);
            new Game(level);        // открыть окно новой игры
        } else
        if (e.getSource() == tipsButton) {
            new PacmanTips(); // создание объекта и отображение окна с советами
        } else
        if (e.getSource() == exitButton) {
            spaceWriter();
            System.exit(0); // передайте значение 0 в качестве параметра для
успешного завершения программы
```

```
        }
    }
    public void spaceWriter(){
        deleteLastCharFromFile();
        try (PrintWriter out = new PrintWriter(new BufferedWriter(new
FileWriter((PacmanMenu.class.getResource("/players.txt")).getPath(), true)))) {
            out.print("");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public static void deleteLastCharFromFile() {
        File file = new
File((Objects.requireNonNull(PacmanMenu.class.getResource("/players.txt"))).getPat
h());

        try {
            BufferedReader reader = new BufferedReader(new FileReader(file));
            StringBuilder sb = new StringBuilder();
            String line = reader.readLine();

            while (line != null) {
                sb.append(line);
                sb.append(System.lineSeparator());
                line = reader.readLine();
            }

            reader.close();
            sb.deleteCharAt(sb.length() - 1);

            FileWriter writer = new FileWriter(file);
            writer.write(sb.toString());
            writer.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }


    public static int chooseLevel(ImageIcon icon_ghost) {
        String[] options = {"1", "2", "3", "4"};
        //ImageIcon icon = new ImageIcon("icon_ghost.png");
        int ch = JOptionPane.showOptionDialog(
                null,
                "Выберите уровень сложности",
                "Уровень сложности",
                JOptionPane.DEFAULT_OPTION,
                JOptionPane.QUESTION_MESSAGE,
                icon_ghost,
                options,
                options[0]
        );
        // Возвращаем выбранный вариант плюс 1, чтобы получить число от 1 до 4
        return ch + 1;
    }
    public static String getUserName(ImageIcon icon_ghost) {
        return JOptionPane.showInputDialog(
                null,
                "Введите ваш ник:",
                "расрасрас",
```

```
                JOptionPane.QUESTION_MESSAGE,
                icon_ghost,
                null,
                "pacpacpac"
        ).toString();
    }
    public void loadImages(){
        icon_ghost = new ImageIcon(getClass().getResource("/icon_ghost.png"));
    }

    public static void main(String[] args) throws IOException {
        new PacmanMenu();
    }
}
```

PacmanTips.java :

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Font;

public class PacmanTips extends JFrame {

    public PacmanTips() {
        setTitle("Pacman Tips");
        setSize(400, 300);
        setLocationRelativeTo(null); // центрируем окно на экране
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // закрыть только это
окно, не всю программу

        JPanel tipsPanel = new JPanel();
        tipsPanel.setBackground(Color.BLACK); // задаем цвет фона
        tipsPanel.setLayout(new BorderLayout());

        JLabel title = new JLabel("PACMAN TIPS");
        title.setFont(new Font("Comic Sans MS", Font.BOLD, 24)); // задаем стиль
шрифта
        title.setForeground(Color.YELLOW); // задаем цвет текста
        title.setHorizontalAlignment(JLabel.CENTER); // выравнивание по центру
        tipsPanel.add(title, BorderLayout.NORTH);

        JLabel tips = new JLabel("<html><body>"
                + "1. Если вы соберете всю еду то выиграете<br>"
                + "2. Пакман и призраки двигаются с одинаковой скоростью. Ваше
преимущество - что они двигаются по алгоритму и вы можете их обмануть.<br>"
                + "3. Старайтесь держаться подальше от призраков и не стоять в
тупике когда они близко. У вас должен быть путь для отступления<br>"
                + "4. В начале игры старайтесь быстрее убежать из центра, иначе
вас быстро поймают"
                + "</body></html>");
        //tips.setFont(new Font("Arial", Font.PLAIN, 14)); // задаем стиль шрифта
        tips.setFont(new Font("Comic Sans MS", Font.PLAIN, 14)); // задаем стиль
шрифта
        tips.setForeground(Color.YELLOW); // задаем цвет текста

        tips.setHorizontalAlignment(JLabel.LEFT); // выравнивание по левому краю
```

```
        tips.setVerticalAlignment(JLabel.TOP); // выравнивание по верхнему краю
        tipsPanel.add(tips, BorderLayout.CENTER);

        add(tipsPanel);
        setVisible(true); // отображаем окно на экране
    }

}
```

Game.java :

```
import javax.swing.*;
import javax.swing.Timer;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.AffineTransform;
import java.awt.image.BufferedImage;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Objects;

public class Game extends JFrame implements KeyListener,ActionListener {
    private JPanel panel;
    private Timer timer;
    private Pacman pacman;
    private int GHOST_COUNT;
    private Ghost[] ghost;
    private static final int BOARD_WIDTH = 20;
    private static final int BOARD_HEIGHT = 20;
    private static final int CELL_SIZE = 20; // размер ячейки в пикселях
    private static String[][] cells = new String[BOARD_WIDTH][BOARD_HEIGHT];
    private int SCORE;
    boolean GAME_OVER = false;
    boolean GAME_WIN = false;
    private JButton exitButton;

    private Image
            wall_img, ghost_img,
            pacman_img, pacman_eat,
            pacman_img_R,pacman_img_L,
            pacman_img_U,pacman_img_D;
    private boolean eat = true;
    private int eat_count = 0;
    public Game(int GHOST_COUNT) {

        super("Game"); // название окна

        exitButton = new JButton("Exit");
        exitButton.addActionListener(this);
        exitButton.setEnabled(false); // кнопка неактивна
        exitButton.setVisible(false); // кнопка невидима

        this.GHOST_COUNT = GHOST_COUNT;
        ghost = new Ghost[GHOST_COUNT];
        loadImage();
        SCORE = 0;
        setSize(BOARD_WIDTH*CELL_SIZE+10, BOARD_HEIGHT*CELL_SIZE+100);
```

```
        setLocationRelativeTo(null); // центрируем окно на экране
        setVisible(true);
        //setExtendedState(MAXIMIZED_BOTH);

        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                // освобождаем ресурсы, связанные с окном Game
                dispose();
            }
        });
        setVisible(true);
        SCORE = 0;
        pacman = new Pacman(150, 150, "LEFT", false); // начальная позиция Пакмана
в пикселях


        ArrayList<Integer> GH_W = new ArrayList<Integer>();
        Collections.addAll(GH_W, 18, 2, 18, 1);

        ArrayList<Integer> GH_H = new ArrayList<Integer>();
        Collections.addAll(GH_H, 1, 1, 17, 15);

        for (int i = 0; i < GHOST_COUNT; i++) {
            ghost[i] = new Ghost((BOARD_WIDTH - GH_W.get(i)) * CELL_SIZE -
CELL_SIZE / 2, (BOARD_HEIGHT - GH_H.get(i)) * CELL_SIZE - CELL_SIZE / 2, "STOP",
pacman, cells, true);
        }

        for (int i = 0; i < GHOST_COUNT; i++) ghost[i].setDirection("STOP");



        cells = new String[][]
                {
                        {"WALL", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL",
"WALL", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL",
"WALL", "WALL", "WALL"},
                        {"WALL", "WALL", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD",
"FOOD", "FOOD", "WALL", "FOOD", "WALL", "WALL", "WALL", "FOOD", "FOOD", "FOOD",
"FOOD", "FOOD", "WALL"},
                        {"WALL", "FOOD", "FOOD", "WALL", "WALL", "WALL", "WALL",
"WALL", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "WALL",
"FOOD", "FOOD", "WALL"},
                        {"WALL", "FOOD", "WALL", "WALL", "WALL", "WALL", "WALL",
"WALL", "FOOD", "FOOD", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL",
"WALL", "FOOD", "WALL"},
                        {"WALL", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "WALL",
"WALL", "FOOD", "WALL", "WALL", "WALL", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD",
"WALL", "FOOD", "WALL"},
                        {"WALL", "WALL", "WALL", "WALL", "WALL", "FOOD", "FOOD",
"FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "WALL", "WALL", "WALL",
"WALL", "WALL", "FOOD", "WALL"},
                        {"WALL", "WALL", "FOOD", "WALL", "WALL", "FOOD", "WALL",
"WALL", "WALL", "WALL", "WALL", "FOOD", "WALL", "WALL", "FOOD", "FOOD", "FOOD",
"FOOD", "FOOD", "WALL"},
                        {"WALL", "WALL", "FOOD", "WALL", "WALL", "FOOD", "WALL",
"FOOD", "FOOD", "FOOD", "FOOD", "WALL", "FOOD", "FOOD", "FOOD", "FOOD", "WALL",
"WALL", "FOOD", "WALL"},
                        {"WALL", "WALL", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD",
"FOOD", "FOOD", "FOOD", "WALL", "WALL", "FOOD", "WALL", "WALL", "FOOD", "WALL",
```

```
"FOOD", "FOOD", "WALL"},
                        {"WALL", "WALL", "FOOD", "WALL", "WALL", "WALL", "FOOD",
"WALL", "FOOD", "FOOD", "WALL", "WALL", "FOOD", "WALL", "WALL", "FOOD", "WALL",
"WALL", "FOOD", "WALL"},
                        {"WALL", "FOOD", "FOOD", "FOOD", "WALL", "FOOD", "FOOD",
"WALL", "FOOD", "WALL", "WALL", "WALL", "FOOD", "WALL", "WALL", "FOOD", "FOOD",
"FOOD", "FOOD", "WALL"},
                        {"WALL", "FOOD", "WALL", "FOOD", "WALL", "FOOD", "FOOD",
"WALL", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "WALL", "WALL", "FOOD", "WALL",
"FOOD", "WALL", "WALL"},
                        {"WALL", "FOOD", "WALL", "FOOD", "WALL", "WALL", "WALL",
"WALL", "WALL", "WALL", "WALL", "WALL", "FOOD", "WALL", "WALL", "FOOD", "WALL",
"FOOD", "FOOD", "WALL"},
                        {"WALL", "FOOD", "WALL", "FOOD", "FOOD", "FOOD", "FOOD",
"FOOD", "FOOD", "WALL", "WALL", "WALL", "FOOD", "FOOD", "FOOD", "FOOD", "WALL",
"WALL", "FOOD", "WALL"},
                        {"WALL", "FOOD", "WALL", "WALL", "FOOD", "WALL", "WALL",
"WALL", "FOOD", "FOOD", "FOOD", "WALL", "WALL", "WALL", "WALL", "FOOD", "FOOD",
"FOOD", "FOOD", "WALL"},
                        {"WALL", "FOOD", "FOOD", "WALL", "FOOD", "FOOD", "FOOD",
"FOOD", "WALL", "WALL", "FOOD", "WALL", "FOOD", "FOOD", "FOOD", "FOOD", "WALL",
"FOOD", "WALL", "WALL"},
                        {"WALL", "WALL", "FOOD", "WALL", "WALL", "FOOD", "WALL",
"FOOD", "FOOD", "WALL", "FOOD", "FOOD", "FOOD", "WALL", "WALL", "WALL", "WALL",
"FOOD", "FOOD", "WALL"},
                        {"WALL", "WALL", "FOOD", "WALL", "WALL", "FOOD", "WALL",
"WALL", "FOOD", "WALL", "WALL", "WALL", "FOOD", "WALL", "WALL", "WALL", "WALL",
"WALL", "FOOD", "WALL"},
                        {"WALL", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "WALL",
"WALL", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD", "FOOD",
"FOOD", "FOOD", "WALL"},
                        {"WALL", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL",
"WALL", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL", "WALL",
"WALL", "WALL", "WALL"}
                    };

/*
        // fill cells with empty cells
        for (int x = 0; x < BOARD_WIDTH; x++)
            for (int y = 0; y < BOARD_HEIGHT; y++)
                cells[x][y] = "EMPTY";


 */
        /* СЛУЧАЙНАЯ КАРТА
        // add walls
        for (int x = 0; x < BOARD_WIDTH; x++) {
            cells[x][0] = "WALL";
            cells[x][BOARD_HEIGHT - 1] = "WALL";
        }
        for (int y = 0; y < BOARD_HEIGHT; y++) {
            cells[0][y] = "WALL";
            cells[BOARD_WIDTH - 1][y] = "WALL";
        }

        // add OBSTACLE and FOOD
        for (int x = 1; x < BOARD_WIDTH - 1; x++)
            for (int y = 1; y < BOARD_HEIGHT - 1; y++) {
                int rand = (int) (Math.random() * 100);
                if ((0 <= rand) && (rand <= 14)) {
                    if (!((x > BOARD_WIDTH - 8) && (y > BOARD_HEIGHT - 8)))
                        cells[x][y] = "OBSTACLE";
```

```
                } else if ((19 <= rand) && (rand <= 23))
                    cells[x][y] = "FOOD";
            }

         */


        panel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                Graphics2D g2d = (Graphics2D) g;

                // включаем сглаживание для более качественной отрисовки
                g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

                // рисуем фон
                g2d.setColor(Color.BLACK);
                g2d.fillRect(0, 0, getWidth(), getHeight());

                if (GAME_OVER) {
                    Font currentFont = g.getFont();

                    Font newFont = currentFont.deriveFont(currentFont.getSize() *
3F);
                    g2d.setFont(newFont);
                    g2d.setColor(Color.BLUE);
                    g2d.drawString("GAME OVER", 95, 350);
                }
                else if (GAME_WIN){
                    Font currentFont = g.getFont();
                    Font newFont = currentFont.deriveFont(currentFont.getSize() *
3F);
                    g2d.setFont(newFont);
                    g2d.setColor(Color.PINK);
                    g2d.drawString("!!! YOU WIN !!!", 95, 350);
                }
                else {

                    //g2d.setColor(Color.YELLOW);
                    //g2d.fillOval(pacman.getX() - 10, pacman.getY() - 10,
CELL_SIZE, CELL_SIZE);
/*
                    ImageIcon ii = new ImageIcon("pacman.png");
                    Image pacUP = ii.getImage();

                    if(pacman.getDirection().equals("UP")) pac =
rotateImage(pacman_img,-90);

 */
                    // рисуем Пакмана
                    //if(pacman.getDirection().equals(""))

                    if(eat)
                        pacman_img = pacman_eat;
                    else
                        switch (pacman.getDirection()) {
                            case "RIGHT" ->
                                    pacman_img = pacman_img_R;
                            case "UP" ->
                                    pacman_img = pacman_img_U;
```

```
                            case "LEFT" ->
                                    pacman_img = pacman_img_L;
                            case "DOWN" ->
                                    pacman_img = pacman_img_D;
                        }
                    g2d.drawImage(pacman_img, pacman.getX() - 10, pacman.getY() -
10, CELL_SIZE, CELL_SIZE, null);



                    // рисуем Призраков
                    g2d.setColor(Color.RED);

                    for (int i = 0; i < GHOST_COUNT; i++)
                        g2d.drawImage(ghost_img, ghost[i].getX() - 10,
ghost[i].getY() - 10, CELL_SIZE, CELL_SIZE, null);

                    //g2d.fillOval(ghost[i].getX() - 10, ghost[i].getY() - 10,
CELL_SIZE, CELL_SIZE);

                    //рисуем карту
                    drawMap(g2d);
                }
            }
        };
        panel.setPreferredSize(new Dimension(200, 100));
        add(panel, BorderLayout.CENTER);
        //add(panel); // добавляем панель на фрейм
        addKeyListener(this); // добавляем слушатель клавиатуры
        setFocusable(true);
        setFocusTraversalKeysEnabled(false);

        timer = new Timer(10, this); // создаем таймер для обновления положения
квадрата
        timer.start(); // запускаем таймер
    }

    @Override
    public void actionPerformed(ActionEvent e){

        // анимация, закрываем рот пакмана
        eat_count +=1;
        if (eat_count==15){
            eat = !eat;
        }
        if (eat_count==30){
            eat = !eat;
            eat_count=0;
        }

        int pacmanX = pacman.getX();
        int pacmanY = pacman.getY();

        int radius = (int) (Math.sqrt(Math.pow(10, 2) / 2));
        int aura = 1;

        // габариты пакмана (нужны так, как мы не можем принять пакмана за
материальную точку)
        int[][] pacmanDimensions = new int[2][3];
        switch (pacman.getDirection()) { // куда повернут
            case "LEFT" -> {
                pacmanDimensions[0][0] = pacmanX - radius;
```

```
                pacmanDimensions[0][1] = pacmanX - CELL_SIZE / 2 - aura - aura;
                pacmanDimensions[0][2] = pacmanX - radius;

                pacmanDimensions[1][0] = pacmanY + radius;
                pacmanDimensions[1][1] = pacmanY;
                pacmanDimensions[1][2] = pacmanY - radius;
            }
            case "RIGHT" -> {
                pacmanDimensions[0][0] = pacmanX + radius;
                pacmanDimensions[0][1] = pacmanX + CELL_SIZE / 2 + aura + aura;
                pacmanDimensions[0][2] = pacmanX + radius;

                pacmanDimensions[1][0] = pacmanY + radius;
                pacmanDimensions[1][1] = pacmanY;
                pacmanDimensions[1][2] = pacmanY - radius;
            }
            case "UP" -> {
                pacmanDimensions[0][0] = pacmanX - radius;
                pacmanDimensions[0][1] = pacmanX;
                pacmanDimensions[0][2] = pacmanX + radius;

                pacmanDimensions[1][0] = pacmanY - radius;
                pacmanDimensions[1][1] = pacmanY - CELL_SIZE / 2 - aura - aura;
                pacmanDimensions[1][2] = pacmanY - radius;
            }
            case "DOWN" -> {
                pacmanDimensions[0][0] = pacmanX - radius;
                pacmanDimensions[0][1] = pacmanX;
                pacmanDimensions[0][2] = pacmanX + radius;

                pacmanDimensions[1][0] = pacmanY + radius;
                pacmanDimensions[1][1] = pacmanY + CELL_SIZE / 2 + aura + aura;
                pacmanDimensions[1][2] = pacmanY + radius;
            }
            case "STOP" -> {
            }
        }
        //  ЛВ В ПВ  0 1 2
        //  Л   _  П  3 4 5
        //  ЛН Н ПН  6 7 8

        //  UP    :  0,1,2
        //  LEFT  :  0,3,6
        //  RIGHT :  2,5,8
        //  DOWN  :  6,7,8
        ///////////////////////
        int[][] pacmanDimensions_inCells = new int[2][3];
        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 3; j++)
                pacmanDimensions_inCells[i][j] = (int) pacmanDimensions[i][j] /
CELL_SIZE;

        //взаимодействие пакамана со стенами и едой
        for (int i = 0; i < 3; i++) {
            if
(((cells[pacmanDimensions_inCells[0][i]][pacmanDimensions_inCells[1][i]]).equals("
OBSTACLE"))
                    ||

((cells[pacmanDimensions_inCells[0][i]][pacmanDimensions_inCells[1][i]]).equals("W
ALL")))
                pacman.setDirection("STOP");
```

```
            else if
((cells[pacmanDimensions_inCells[0][i]][pacmanDimensions_inCells[1][i]]).equals("F
OOD")) {
                SCORE++;

cells[pacmanDimensions_inCells[0][i]][pacmanDimensions_inCells[1][i]] = "EMPTY";
            }
        }

        pacman.move(); // перемещаем Пакмана

        ///////////search path

        for (int i = 0; i < GHOST_COUNT; i++) {
            if ((ghost[i].getDirection().equals("STOP")) && (!GAME_OVER) &&
(!GAME_WIN)) ghost[i].setDirection("UP");

            switch (ghost[i].getDirection()) {
                case "LEFT" -> {
                    if (!isFreePath(ghost[i].getDirection(), ghost[i].getX(),
ghost[i].getY())) {
                        boolean G1 = isFreePath("DOWN", ghost[i].getX(),
ghost[i].getY());
                        boolean G2 = isFreePath("UP", ghost[i].getX(),
ghost[i].getY());

                        if (G1 && G2) {
                            //выбираем куда идти
                            int dx1 = Math.abs(ghost[i].getX() + 10 -
pacman.getX());
                            int dx2 = Math.abs(ghost[i].getX() - 10 -
pacman.getX());
                            if (dx1 < dx2)
                                ghost[i].setDirection("DOWN");
                            else
                                ghost[i].setDirection("UP");
                        } else if (G1)
                            ghost[i].setDirection("DOWN");//

                        else if (G2)
                            ghost[i].setDirection("UP");//

                        else ghost[i].setDirection("RIGHT");
                    }
                }
                case "RIGHT" -> {
                    if (!isFreePath(ghost[i].getDirection(), ghost[i].getX(),
ghost[i].getY())) {
                        boolean G1 = isFreePath("UP", ghost[i].getX(),
ghost[i].getY());
                        boolean G2 = isFreePath("DOWN", ghost[i].getX(),
ghost[i].getY());

                        if (G1 && G2) {
                            //выбираем куда идти
                            int dx1 = Math.abs(ghost[i].getX() + 10 -
pacman.getX());
                            int dx2 = Math.abs(ghost[i].getX() - 10 -
pacman.getX());
                            if (dx1 < dx2)
                                ghost[i].setDirection("UP");
                            else
```

```
                            ghost[i].setDirection("DOWN");
                        } else if (G1)
                            ghost[i].setDirection("UP");//

                        else if (G2)
                            ghost[i].setDirection("DOWN");//

                        else ghost[i].setDirection("LEFT");
                    }
                }
                case "UP" -> {
                    if (!isFreePath(ghost[i].getDirection(), ghost[i].getX(),
ghost[i].getY())) {
                        boolean G1 = isFreePath("LEFT", ghost[i].getX(),
ghost[i].getY());
                        boolean G2 = isFreePath("RIGHT", ghost[i].getX(),
ghost[i].getY());

                        if (G1 && G2) {
                            //выбираем куда идти
                            int dx1 = Math.abs(ghost[i].getX() + 10 -
pacman.getX());
                            int dx2 = Math.abs(ghost[i].getX() - 10 -
pacman.getX());
                            if (dx1 < dx2)
                                ghost[i].setDirection("LEFT");
                            else
                                ghost[i].setDirection("RIGHT");
                        } else if (G1)
                            ghost[i].setDirection("LEFT");//

                        else if (G2)
                            ghost[i].setDirection("RIGHT");//

                        else ghost[i].setDirection("DOWN");
                    }
                }
                case "DOWN" -> {
                    if (!isFreePath(ghost[i].getDirection(), ghost[i].getX(),
ghost[i].getY())) {
                        boolean G1 = isFreePath("RIGHT", ghost[i].getX(),
ghost[i].getY());
                        boolean G2 = isFreePath("LEFT", ghost[i].getX(),
ghost[i].getY());

                        if (G1 && G2) {
                            //выбираем куда идти
                            int dx1 = Math.abs(ghost[i].getX() + 10 -
pacman.getX());
                            int dx2 = Math.abs(ghost[i].getX() - 10 -
pacman.getX());
                            if (dx1 < dx2)
                                ghost[i].setDirection("RIGHT");
                            else
                                ghost[i].setDirection("LEFT");
                        } else if (G1)
                            ghost[i].setDirection("RIGHT");//

                        else if (G2)
                            ghost[i].setDirection("LEFT");//

                        else ghost[i].setDirection("UP");
```

```
                }
            }
        }
    }

    /////////

    for (int i = 0; i < GHOST_COUNT; i++) {
        ghost[i].move();
        if (ghostEatPac(ghost[i], pacman))
            gameOver();
    } // перемещаем приведение и смотрим, поймало ли оно пакмана

    if(SCORE==177){
        gameWin();
        WinWriter();
        SCORE++;
    }

    panel.repaint();          // перерисовываем игровое поле
    System.out.println();
}



public boolean ghostEatPac(Ghost ghost, Pacman pacman) {
    return ((ghost.getX() / CELL_SIZE) == (pacman.getX() / CELL_SIZE)) &&
            ((ghost.getY() / CELL_SIZE) == (pacman.getY() / CELL_SIZE));
}

public void gameWin(){
    GAME_WIN = true;
    for (int i = 0; i < GHOST_COUNT; i++) {
        pacman.setDirection("STOP");
        for (int j = 0; j < GHOST_COUNT; j++)
            ghost[j].setDirection("STOP");

    }


    System.out.println("YOU WIN");
}

public void gameOver() {
    GAME_OVER = true;

    for (int i = 0; i < GHOST_COUNT; i++) {
        pacman.setDirection("STOP");
        for (int j = 0; j < GHOST_COUNT; j++)
            ghost[j].setDirection("STOP");

    }
    System.out.println("GAME OVER");
}

public boolean pacChangDir(String oldDir, String newDir) {
    return !oldDir.equals(newDir);
}

public boolean isFreePath(String direction, int X, int Y) {

    int radius = (int) (Math.sqrt(Math.pow(10, 2) / 2));
```

```
        int aura = 1;

        // габариты объекта (нужны так, как мы не можем принять объект за
материальную точку)
        int[][] ghostDimensions = new int[2][3];

        switch (direction) { // куда повернут
            case "LEFT" -> {
                ghostDimensions[0][0] = X - radius;
                ghostDimensions[0][1] = X - CELL_SIZE / 2 - aura - aura;
                ghostDimensions[0][2] = X - radius;

                ghostDimensions[1][0] = Y + radius;
                ghostDimensions[1][1] = Y;
                ghostDimensions[1][2] = Y - radius;
            }
            case "RIGHT" -> {
                ghostDimensions[0][0] = X + radius;
                ghostDimensions[0][1] = X + CELL_SIZE / 2 + aura + aura;
                ghostDimensions[0][2] = X + radius;

                ghostDimensions[1][0] = Y + radius;
                ghostDimensions[1][1] = Y;
                ghostDimensions[1][2] = Y - radius;
            }
            case "UP" -> {
                ghostDimensions[0][0] = X - radius;
                ghostDimensions[0][1] = X;
                ghostDimensions[0][2] = X + radius;

                ghostDimensions[1][0] = Y - radius;
                ghostDimensions[1][1] = Y - CELL_SIZE / 2 - aura - aura;
                ghostDimensions[1][2] = Y - radius;
            }
            case "DOWN" -> {
                ghostDimensions[0][0] = X - radius;
                ghostDimensions[0][1] = X;
                ghostDimensions[0][2] = X + radius;

                ghostDimensions[1][0] = Y + radius;
                ghostDimensions[1][1] = Y + CELL_SIZE / 2 + aura + aura;
                ghostDimensions[1][2] = Y + radius;
            }
            case "STOP" -> {
            }
        }

        //  ЛВ В ПВ  0 1 2
        //  Л  _  П  3 4 5
        //  ЛН Н ПН  6 7 8

        //  UP    :  0,1,2
        //  LEFT  :  0,3,6
        //  RIGHT :  2,5,8
        //  DOWN  :  6,7,8
        ///////////////////////

        int[][] Dimensions_inCells = new int[2][3];
        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 3; j++)
                Dimensions_inCells[i][j] = (int) ghostDimensions[i][j] /
CELL_SIZE;
```

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

((cells[Dimensions_inCells[0][i]][Dimensions_inCells[1][i]]).equals("OBSTACLE"))
                                ||

((cells[Dimensions_inCells[0][i]][Dimensions_inCells[1][i]]).equals("WALL")))
                return false;
        return true;
    }

    public void pause(boolean ONorOFF) {
        pacman.setGAME_OVER(ONorOFF);
        for (int i = 0; i < GHOST_COUNT; i++)
            ghost[i].setGAME_OVER(ONorOFF);
    }

    @Override
    public void keyPressed(KeyEvent e) {
        if (!GAME_OVER)
            switch (e.getKeyCode()) {
                case KeyEvent.VK_LEFT -> pacman.setDirection("LEFT");
                case KeyEvent.VK_RIGHT -> pacman.setDirection("RIGHT");
                case KeyEvent.VK_UP -> pacman.setDirection("UP");
                case KeyEvent.VK_DOWN -> pacman.setDirection("DOWN");

                case KeyEvent.VK_SPACE -> pause(true);
            }
    }

    @Override
    public void keyTyped(KeyEvent e) {
    }

    @Override
    public void keyReleased(KeyEvent e) {
    }

    private void drawMap(Graphics2D g2d) {
        // текстуры
        for (int x = 0; x < cells.length; x++) {
            for (int y = 0; y < cells[0].length; y++) {
                switch (cells[x][y]) {
                    case "EMPTY":
                        break;
                    case "FOOD":
                        g2d.setColor(Color.PINK);
                        g2d.fillOval(x * CELL_SIZE + CELL_SIZE / 4 + CELL_SIZE /
6, y * CELL_SIZE + CELL_SIZE / 4 + CELL_SIZE / 6, CELL_SIZE / 4, CELL_SIZE / 4);
                        break;
                    case "WALL":
                        //g2d.setColor(Color.BLUE);
                        //g2d.fillRect(x * CELL_SIZE, y * CELL_SIZE, CELL_SIZE,
CELL_SIZE);

                        ////
                        g2d.drawImage(wall_img, x * CELL_SIZE, y * CELL_SIZE,
CELL_SIZE, CELL_SIZE, null);
                        break;
                    case "OBSTACLE":
                        //g2d.setColor(Color.CYAN);
```

```
                        //g2d.fillRect(x * CELL_SIZE, y * CELL_SIZE, CELL_SIZE,
CELL_SIZE);

                        ////
                        g2d.drawImage(wall_img, x * CELL_SIZE, y * CELL_SIZE,
CELL_SIZE, CELL_SIZE, null);
                        break;
                }
            }
        }
        g2d.drawString("Passed: " + SCORE*100/177+"%", CELL_SIZE*BOARD_WIDTH/2-25,
CELL_SIZE*BOARD_HEIGHT+25);
    }
    private void loadImage() {

        // Load images

        ImageIcon ii = new ImageIcon(getClass().getResource("/wall.jpg"));
        wall_img = ii.getImage();

        ii = new ImageIcon(getClass().getResource("/pacman.png"));
        pacman_img_R = ii.getImage();
        pacman_img_D = rotateImage(pacman_img_R,90);
        pacman_img_L = rotateImage(pacman_img_D,90);
        pacman_img_U = rotateImage(pacman_img_L,90);

        pacman_eat = new
ImageIcon(getClass().getResource("/pacman_eat.png")).getImage();

        ghost_img = new
ImageIcon(getClass().getResource("/ghost.png")).getImage();



    }
    public static Image rotateImage(Image image, double angle) {
        // Преобразуем исходное изображение в BufferedImage
        BufferedImage bimg = new BufferedImage(image.getWidth(null),
image.getHeight(null), BufferedImage.TYPE_INT_ARGB);

        Graphics2D g = bimg.createGraphics();
        g.drawImage(image, 0, 0, null);
        g.dispose();

        // Создаем AffineTransform и поворачиваем изображение на заданный угол
        AffineTransform at = new AffineTransform();
        at.rotate(Math.toRadians(angle), bimg.getWidth() / 2, bimg.getHeight() /
2);

        // Создаем новый BufferedImage и возвращаем его в виде Image
        BufferedImage rotatedImg = new BufferedImage(bimg.getWidth(),
bimg.getHeight(), bimg.getType());
        Graphics2D g2 = rotatedImg.createGraphics();
        g2.setTransform(at);
        g2.drawImage(bimg, 0, 0, null);
        g2.dispose();

        return rotatedImg;
    }
    public void WinWriter(){
        try (PrintWriter out = new PrintWriter(new BufferedWriter(new
FileWriter((Objects.requireNonNull(PacmanMenu.class.getResource("/players.txt"))).
```

```
getPath(), true))) {
                out.print(" "+GHOST_COUNT + ",");
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

/*
    public static void main(String[] args) {
        Game game = new Game();
        game.setVisible(true);
    }


 */
}
```

Pacman.java :

```
public class Pacman {
    private boolean GAME_OVER;
    private int x;
    private int y;
    private String direction;
    final static int speed = 2;
    private static final int CELL_SIZE = 20; // размер ячейки в пикселях

    public Pacman(int x, int y, String direction, boolean GAME_OVER) {
        this.x = x;
        this.y = y;
        this.direction = "RIGHT";
        this.GAME_OVER = GAME_OVER;
    }
    public int getX() {
        return x;
    }
    public int getY() {
        return y;
    }
    public String getDirection() {
        return direction;
    }
    public void setDirection(String direction) {
        this.direction = direction;
    }
    public void setGAME_OVER(boolean GAME_OVER) {
        this.GAME_OVER = GAME_OVER;
    }
    public void move() {
        switch (direction) {
            case "LEFT" -> x -= speed;
            case "RIGHT" -> x += speed;
            case "UP" -> y -= speed;
            case "DOWN" -> y += speed;
            case "STOP" -> {}
        }
    }
}
```

Ghost.java :

```
public class Ghost {
    private boolean GAME_OVER;
    private static final int CELL_SIZE = 20; // размер ячейки в пикселях
    private int x;
    private int y;
    private String direction;
    private String[][] cells;
    private Pacman pacman;
    final static int speed = 2;
    public Ghost(int x, int y, String direction, Pacman pacman, String[][] cells,
boolean GAME_OVER) {
        this.x = x;
        this.y = y;
        this.direction = direction;
        this.pacman = pacman;
        this.cells = cells;
        this.GAME_OVER = GAME_OVER;
    }
    public int getX() {
        return x;
    }
    public int getY() {
        return y;
    }
    public String getDirection() {
        return direction;
    }
    public void setDirection(String direction) {
        this.direction = direction;
    }
    public void setGAME_OVER(boolean GAME_OVER) {
        this.GAME_OVER = GAME_OVER;
    }
    public void move() {
        switch (direction) {
            //case "STOP" -> locator();
            case "LEFT" -> x -= speed;
            case "RIGHT" -> x += speed;
            case "UP" -> y -= speed;
            case "DOWN" -> y += speed;
        }
    }
    public void locator(){
        int dx = Math.abs(pacman.getX() / CELL_SIZE - x / CELL_SIZE);
        int dy = Math.abs(pacman.getY() / CELL_SIZE - y / CELL_SIZE);
        if (dx > dy)// лево/право
            if (dx > (x + CELL_SIZE) / CELL_SIZE)
                direction = "RIGHT";
            else
                direction = "LEFT";
        else if (dy > (y + CELL_SIZE) / CELL_SIZE)// верх/низ
            direction = "DOWN";
        else
            direction = "UP";
    }
}
```