МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

КУРСОВАЯ РАБОТА

ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

доцент, канд. техн. наук		Л.Н. Бариков
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

ПРОГРАММА
Игра «Змейка» на Java
ОП 44.4041.21 ПЗ

по дисциплине: Основы программирования

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4041		А.В. Комолкин
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ..…… 3

1 Постановка задачи…….6

2 Спецификация……6

3 Текст программы…….10

3.1 Текст программы…….10

3.2 Результат тестирования……..16

4 Описание программы……19

4.1 Общие сведения……..21

4.2 Функциональное назначение…….21

4.3 Описание логической структуры……..21

4.4 Спецификация функции…….21

4.5 Используемые технические средства……21

4.6 Вызов и загрузка…….21

5 Описание применения……...23

5.1 Назначение программы……..25

5.2 Условие применения……..25

5.3 Описание задачи…….25

5.4 Характеристики занимаемой памяти…….25

ЗАКЛЮЧЕНИЕ……..26

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……...27

ВВЕДЕНИЕ

Данная работа предназначена для закрепления учебного материала, изученного по курсу «Основы программирования».

Целью курсовой работы является закрепление и углубление знаний, полученных при изучении дисциплины в ходе лекционных и практических занятий, получение практических навыков создания программ с использованием объектно-ориентированной технологии в современных средах проектирования. Выполнение курсовой работы предусматривает: создание классов, полей и методов класса. А также получение практических навыков программирования алгоритмов обработки данных.

Кроме этого, курсовая работа предназначена для приобретения навыков оформления сопроводительной документации к разработанному программному средству.

1 Постановка задачи

В данном проекте для игры «Змейка» должны быть приняты следующие правила:

- змейка (упорядоченный набор связанных звеньев с явно выделенными концами – головой и хвостом) передвигается по полю 20 х 20; ·
- в начале игры змейка состоит из одного звена;
- перемещение змейки состоит в добавлении одного звена к ее голове в требуемом направлении (в направлении ее движения) и удалении одного звена хвоста;
- если при перемещении змейки ее голова натыкается на препятствие (змейка натыкается на себя или на границу поля), то игра проиграна;
- в каждый момент времени на игровом поле находится кролик, занимающий одну клетку поля;
- если при перемещении голова змейки натыкается на кролика, то змейка его «съедает» и вырастает на одно звено, а для выполнения предыдущего правила на поле в произвольном свободном месте автоматически появляется новый кролик;
- выигрыш состоит в достижении змейкой длины в 40 звеньев.

Программа должна иметь удобный интерфейс, быстро работать.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

ПРОГРАММА

Игра «Змейка» на Java

Спецификация

44.4041.21-01

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4041 _____ А. В. Комолкин
подпись, дата инициалы, фамилия

Санкт-Петербург 2021

2.1 Спецификация

Обозначение	Наименование	Примечание
44.4041.21-12	Документация Программа Игра «Змейка» на Java Текст программы	
44.4041.21-13	Документация Программа Игра «Змейка» на Java Описание программы	
44.4041.21-31	Документация Программа Игра «Змейка» на Java Описание применения	

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

ПРОГРАММА

Игра «Змейка» на Java

Текст программы

44.4041.21-12

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4041 _____ А. В. Комолкин

подпись, дата — инициалы, фамилия

Санкт-Петербург 2021

Аннотация

В документе приводится текст программы «Игра «Змейка»», написанной на языке *Java*, а также результат отладки и тестирования.

СОДЕРЖАНИЕ

3.1 Текст программы

3.2 Результат тестирования

3.1 Текст программы

Основной файл Game.java

```
package com.kom.alex.curse.Game;


import java.awt.event.KeyEvent;
import java.util.ArrayList;

/**
* Основной класс программы
*/
public class Game {
   public static Game game;
   private int width;
   private int height;
   private Snake snake;
   private Rabbit rabbit;
   private int initialDelay = 600;
   private int delayStep = 20;

   public Game(int width, int height, Snake snake) {
       this.width = width;
       this.height = height;
       this.snake = snake;
       game = this;
   }

   public static void main(String[] args) throws InterruptedException {
       game = new Game(20, 20, new Snake(10, 10)); // Игровое поле и
изначальные координаты змеи на нём
       game.snake.setDirection(SnakeDirection.DOWN); // Изначально змея
будет двигаться вниз
       game.createRabbit(); // Создаем кролика
       game.run(); // Запуск игры
   }

   public Snake getSnake() {
       return snake;
   }

   public void setSnake(Snake snake) {
       this.snake = snake;
   }

   public Rabbit getRabbit() {
       return rabbit;
   }

   public void setRabbit(Rabbit rabbit) {
       this.rabbit = rabbit;
   }

   public int getWidth() {
       return width;
```

```
    }

    public void setWidth(int width) {
        this.width = width;
    }

    public int getHeight() {
        return height;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    /**
     * Основной цикл программы
     * Тут происходят все важные действия
     */
    public void run() throws InterruptedException {
        //Создаем объект, который будет "наблюдать за клавиатурой" и
стартуем его.
        KeyboardObserver keyboardObserver = new KeyboardObserver();
        keyboardObserver.start();

        //Пока змея жива
        while (snake.isAlive()) {
            //"наблюдатель" содержит события о нажатии клавиш
            if (keyboardObserver.hasKeyEvents()) {
                KeyEvent event = keyboardObserver.getEventFromTop();
                //Если игрок нажимает кнопку 'q' - выйти из игры.
                if (event.getKeyChar() == 'q') return;

                //Если "стрелка влево" - сдвинуть фигурку влево
                if (event.getKeyCode() == KeyEvent.VK_LEFT)
                    snake.setDirection(SnakeDirection.LEFT);
                    //Если "стрелка вправо" - сдвинуть фигурку вправо
                else if (event.getKeyCode() == KeyEvent.VK_RIGHT)
                    snake.setDirection(SnakeDirection.RIGHT);
                    //Если "стрелка вверх" - сдвинуть фигурку вверх
                else if (event.getKeyCode() == KeyEvent.VK_UP)
                    snake.setDirection(SnakeDirection.UP);
                    //Если "стрелка вниз" - сдвинуть фигурку вниз
                else if (event.getKeyCode() == KeyEvent.VK_DOWN)
                    snake.setDirection(SnakeDirection.DOWN);
            }

            snake.move();   //Движение змеи
            print();        //Отображение текущего состояния игры
            sleep();        //Пауза между перерисовкой экрана
        }

        //Выводим сообщение "Game Over" в случае проигрыша
        System.out.println("Game Over!");
    }

    /**
     * Выводим на экран состояние игры на данный момент
```

```
     */
    public void print() {
        //Создаем двумерный целочисленный массив, соответствующий размерам
игрового поля.
        //По умолчанию каждый элемент массива инициализируется нулём.
        int[][] matrix = new int[height][width];

        //Заполняем элементы массива, в которых есть секция змеи единицами.
        ArrayList<SnakeSection> sections = new
ArrayList<SnakeSection>(snake.getSections());
        for (SnakeSection snakeSection : sections) {
            matrix[snakeSection.getY()][snakeSection.getX()] = 1;
        }

        //В элемент массива с координатами головы змеи записываем 2 для
живой змеи, 4 для мёртвой.
        matrix[snake.getY()][snake.getX()] = snake.isAlive() ? 2 : 4;

        //В элемент массива с координатами кролика записываем 3.
        matrix[rabbit.getY()][rabbit.getX()] = 3;

        //Выводим все символы на экран. Нулевые элементы массива рисуются
точками.
        //Элементы массива с 1 рисуется "o". Голова "O", либо "RIP". Кролик
"0_0"
        String[] symbols = {" . ", " o ", " O ", "0_0", "RIP"};
        for (int y = 0; y < height; y++) {
            for (int x = 0; x < width; x++) {
                System.out.print(symbols[matrix[y][x]]);
            }
            System.out.println();
        }
        System.out.println();
        System.out.println();
        System.out.println();
    }

    /**
     * Метод, вызываемый только тогда, когда кролика съели
     */
    public void eatRabbit() {
        createRabbit();
    }

    /**
     * Создание нового кролика
     */
    public void createRabbit() {
        int x = (int) (Math.random() * width);
        int y = (int) (Math.random() * height);

        rabbit = new Rabbit(x, y);
    }

    /**
     * Делаем паузу, длина которой зависит от величины змеи
     */
```

```
   public void sleep() throws InterruptedException {
       try { //Блок {try} позволяет обработать исключения, выброшенные из
методов внутри фигурных скобок.
           int level = snake.getSections().size();
           int delay = level < 15 ? (initialDelay - delayStep * level) :
200;
           //Метод sleep Java позовляет остановить выполнение потока
           //Бросает исключения InterruptedException
           Thread.sleep(delay);
       } catch (InterruptedException e) {  //Блок {catch} ловит
исключения, перечисленные в круглых скобочках
                                           //если они случились внутри
блока {try}
           throw e;                        //Оператор throw позволяет
бросить исключения выше по стеку
       }
   }
}
```

KeyboardObserver.java

```
package com.kom.alex.curse.Game;

import javax.swing.*;
import java.awt.*;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.Queue;
import java.util.concurrent.ArrayBlockingQueue;

public class KeyboardObserver extends Thread {
                private     Queue<KeyEvent>     keyEvents     =     new
ArrayBlockingQueue<KeyEvent>(100);

   private JFrame frame;

   @Override
    public void run() { // Некоторые нижние элементы были найдены мной в
процессе разработки игры,
                        // я разобрал каждую, изучил детали их работы и
вставил в свою программу
       frame = new JFrame("Структура для нажатия клавиш клавиатуры");
       frame.setTitle("Демонстрация невидимого для пользавателя JFrame");
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

       frame.setUndecorated(true);
       frame.setSize(400, 400);
       frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
       frame.setLayout(new GridBagLayout());

       frame.setOpacity(0.0f);
       frame.setVisible(true);

       frame.addFocusListener(new FocusListener() {
```

```
            @Override
            public void focusGained(FocusEvent e) {
            }

            @Override
            public void focusLost(FocusEvent e) {
                System.exit(0);
            }
        });


        frame.addKeyListener(new KeyListener() {

            public void keyTyped(KeyEvent e) {
            }

            public void keyReleased(KeyEvent e) {
            }

            public void keyPressed(KeyEvent e) {
                keyEvents.add(e);
            }
        });
    }


    public boolean hasKeyEvents() {
        return !keyEvents.isEmpty();
    }

    public KeyEvent getEventFromTop() {
        return keyEvents.poll();
    }
}
```

Snake.java

```
package com.kom.alex.curse.Game;

import java.util.ArrayList;

/**
* Класс змеи
*/
public class Snake {
    //Направление движения змеи
    private SnakeDirection direction;
    //Состояние змеи(жива или нет)
    private boolean Alive;
    //Список кусков змеи
    private ArrayList<SnakeSection> sections;

    public Snake(int x, int y) {
        sections = new ArrayList<SnakeSection>();
        sections.add(new SnakeSection(x, y));
        Alive = true;
    }
```

```
    public boolean isAlive() {
        return Alive;
    }

    public int getX() {
        return sections.get(0).getX();
    }

    public int getY() {
        return sections.get(0).getY();
    }

    public SnakeDirection getDirection() {
        return direction;
    }

    public void setDirection(SnakeDirection direction) {
        this.direction = direction;
    }

    public ArrayList<SnakeSection> getSections() {
        return sections;
    }

    /**
     * Этот метод перемещает змею на один ход
     * Направление перемещения задано переменной direction
     */
    public void move() {

        if (!Alive) return; // Если змея жива

        if (direction == SnakeDirection.UP)
            move(0, -1);
        else if (direction == SnakeDirection.RIGHT)
            move(1, 0);
        else if (direction == SnakeDirection.DOWN)
            move(0, 1);
        else if (direction == SnakeDirection.LEFT)
            move(-1, 0);
    }


    /**
     * С помощью этого метода перемещаем змею в соседнюю клетку
       * Координаты клетки заданы относительно текущей головы с помощью
переменных (dx, dy)
     */
    private void move(int dx, int dy) {
        //Здесь создаем новый "кусочек змеи"
        SnakeSection head = sections.get(0);
        head = new SnakeSection(head.getX() + dx, head.getY() + dy);

        //Проверяем, не вылезла ли голова за границу комнаты
```

```
        checkBorders(head);
            if (!Alive) return;

        //Проверяем, не пересекает ли змея саму себя
        checkSnakeBody(head);
            if (!Alive) return;

        //Проверяем, не съела ли змея кролика
        Rabbit rabbit = Game.game.getRabbit();
                    if (head.getX() == rabbit.getX() && head.getY() ==
rabbit.getY())
                 //Если координаты головы змеи и координаты кролика совпали,
значит кролика съели
        {
            sections.add(0, head);                    //Добавляем новый кусок
змеи
               Game.game.eatRabbit();                                  //Хвост не
удаляем, но создаем нового кролика
        }
             else                                                  //В случае если
кролик не съеден, змея просто движется дальше
        {
              sections.add(0, head);                          //Добавляем новую
голову
                     sections.remove(sections.size() - 1);      //Удаляем
последнийэлемент с хвоста
        }
    }

    /**
     * Проверяем, находится ли новая голова в пределах комнаты
     */
    private void checkBorders(SnakeSection head) {
        if ((head.getX() < 0 || head.getX() >= Game.game.getWidth()) ||
                head.getY() < 0 || head.getY() >= Game.game.getHeight()) {
            Alive = false;
        }
    }

    /**
     * Проверяем, не пересекает ли змея свой же участок тела
     */
    private void checkSnakeBody(SnakeSection head) {
        if (sections.contains(head)) {
            Alive = false;
        }
    }
}
```

Rabbit.java

```
package com.kom.alex.curse.Game;

public class Rabbit { // Создание класса Rabbit, элемент, который должна
съесть змея
    private int x;
```

```
    private int y;

    public Rabbit(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }
}
```

SnakeDirection.java

```
package com.kom.alex.curse.Game;


public enum SnakeDirection { //Здесь задаются все направления,
                             // которые пользователь может использовать для
перемещения змеи
    UP,
    RIGHT,
    DOWN,
    LEFT
}
```

SnakeSection.java

```
package com.kom.alex.curse.Game;

public class SnakeSection extends Object { // Класс SnakePosition отвечает
за координаты,
                              // в которых будет появляться змея в начале
игры и координаты,
                             // по которым она будет перемещаться во время
игры
    private int x;
    private int y;



    public SnakeSection(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }
```

```
    public int getY() {
        return y;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        SnakeSection that = (SnakeSection) o;

        if (x != that.x) return false;
        if (y != that.y) return false;

        return true;
    }

    @Override
    public int hashCode() {
        return 50 * x + y;
    }
}
```

3.2 Результаты тестирования

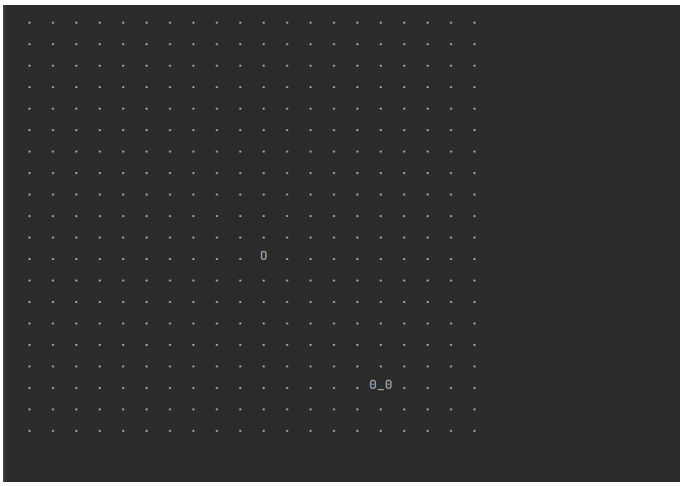

Рисунок 1 – Запуск игры

Рисунок 2 – Змейка съела двух кроликов

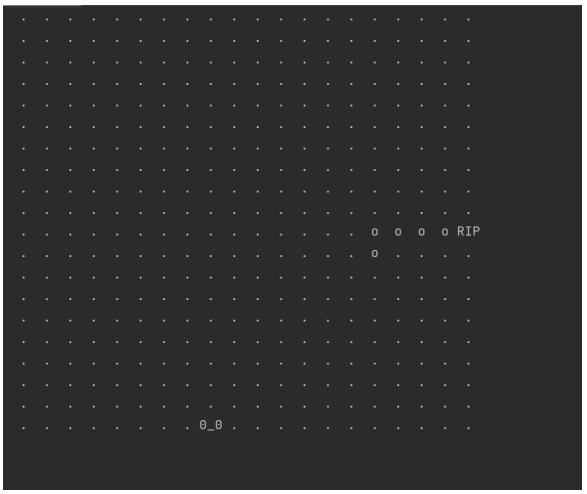

Рисунок 3 – Змейка врезается в бортик

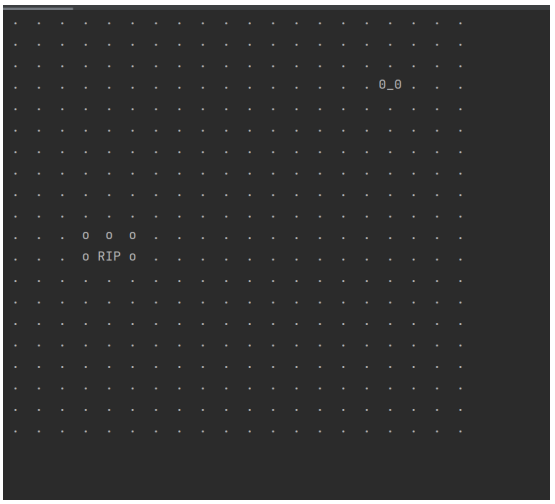

Рисунок 4 – Змейка врезается в саму себя

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

ПРОГРАММА

Игра «Змейка» на Java

Описание программы

44.4041.21-13

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4041 _____ А. В. Комолкин
подпись, дата инициалы, фамилия

Санкт-Петербург 2021

Аннотация

В документе указаны общие сведения о программе «Игра «Змейка»». Приведено общее описание функционирования программы. Приведены общие сведения о языке программирования. Описаны исходные и выходные параметры и логика работы программы, даны сведения об используемых технических средствах и запуске программы.

СОДЕРЖАНИЕ

4.1 Общие сведения

4.2 Функциональное назначение

4.3 Описание логической структуры

4.4 Спецификация функции

4.5 Используемые технические средства

4.6 Вызов и загрузка

4.1 Общие сведения

Программа написана на языке *Java* и состоит из набора методов. Игра змейка веселая и занимательная, разработка которой помогает отточить навыки программирования. Первый вариант игры “Змейка” появился в 1979 году.

4.2 Функциональное назначение

Компьютерная игра, которая развивает мелкую моторику, с ее помощью можно улучшить свою реакцию. Функциональным назначением программного средства является предоставление пользователю возможности электронного прохождения игры «Змейка»

4.3 Описание логической структуры

Программа написана на языке *Java*. При открытии программы появляется окно с игровым полем, кроликом и змейкой. Для продолжения работы пользователю необходимо выбрать соответствующее направление для змейки, которое будет приводить к тому, что змейка будет съедать кролика.

Варианты направление следующие:

1. Вверх.
2. Вниз.
3. Вправо.
4. Влево.

4.4 Спецификация основных функций

Класс Game - основной класс программы:

Содержит метод “Game”, который принимает параметры змеи, длины и высоты игрового поля. В этом классе также существуют методы: основной

метод “main”, который отвечает за то, чтобы игра запускалась, методы “run”, “print”, “createRabbit”, “eatRabbit”, “sleep”,

Класс Keyboard:

Содержит в себе метод “run”, отвечающий за структуру нажатия клавиш. Он также содержит внутренние методы.

Класс Snake:

Содержит метод “Snake”, принимающий параметры x,y. Он создает список и добавляет в него змею. Здесь главным является метод “move”.

Класс Rabbit:

Элемент, который должна съесть змея.

Класс SnakeDirection:

Здесь задаются все направления, которые пользователь может использовать для перемещения змейки.

Класс SnakeSection:

Отвечает за координаты, в которых будет появляться змея вначале игры, и координаты, по которым она будет перемещаться во время игры.

Метод “print” отвечает за алгоритм заполнения матрицы цифрами. Он создает двумерный целочисленный массив, соответствующий размерам игрового поля, инициализирует каждый элемент нулём, а элементы массива, в которых есть секция змеи единицами. В элемент массива с координатами головы змеи записывается 2 для живой змеи, 4 для мёртвой. В элемент массива с координатами кролика записывается 3. Все символы выводятся на экран. Нулевые элементы массива рисуются точками. Элементы массива с 1 рисуется "o". Голова "O", либо "RIP". Кролик "0_0"

Метод “run” подписывается на события клавиатуры и в цикле, пока змея живая, двигает её, отображает текущее состояние игры и делает паузу между перерисовкой.

Метод “Game” принимает параметры змеи, длины и высоты игрового поля. В этом классе также существуют методы: основной метод “main”, который отвечает за то, чтобы игра запускалась, методы “run”, “print”, “createRabbit”, “eatRabbit”, “sleep”.

Метод “run” наблюдает за клавиатурой и содержит в себе события о нажатии клавиш. Также он вызывает методы “move”, “print” и “sleep”.

Метод “print” создает двумерный целочисленный массив, который выводит игровое поле с помощью символов.

Метод “sleep” отвечает за скорость игры и позволяет бросить исключения выше по стеку.

“move” отвечает за движение змеи. Методы “checkBorders” и “checkSnakeBody” делают так, чтобы змейка не была бессмертной.

Метод “Snake”, принимающий параметры x,y. Создает список и добавляет в него змею.

Входные данные:

Адрес файла для чтения

Выходные данные:

Игра выводит игровое поле, змею и кролика в окно терминала.

4.5 Используемые технические средства

Для нормального функционирования программы необходимо наличие ПК, на который установлена ОС Windows 10.

4.6 Вызов и загрузка

Запуск игры осуществляется при помощи “run” конфигурации и среды разработки идеи.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

ПРОГРАММА

Игра «Змейка» на Java

Описание применения

44.4041.21-31

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4041 _____ А. В. Комолкин
подпись, дата инициалы, фамилия

Санкт-Петербург 2021

Аннотация

В документе приводится назначение программы, её основные характеристики и область применения. Также приводятся общие характеристики входной и выходной информации, описывается метод решения задачи.

СОДЕРЖАНИЕ

5.1 Назначение программы

5.2 Условие применения

5.3 Описание задачи

5.4 Характеристика занимаемой памяти

5.1 Назначение программы

Змейка, которая может двигаться в ограниченной плоскости.

Маневрами змейки должен управлять играющий. Змейка должна увеличиваться в размерах при столкновении с яблоком и погибать при столкновении со стеной, при пересечении самой себя и при выходе за границы плоскости заданной программистом.

Стена - объект при столкновении с которым змейка погибает.

5.2 Условия применения

Программа написана на языке *Java*. Для нормального функционирования программы необходимо наличие ПК, на котором установлена ОС Windows 10. Технические средства должны обеспечить вывод результатов выполнения на экран. Для запуска программы необходимо запустить исполняемый файл.

5.3 Описание задачи

Разработать программу, реализующую игру «Змейка» в режиме пользователь-компьютер.

5.4 Характеристики занимаемой памяти

Файл .exe – 44 Кб

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы мною были рассмотрены основные аспекты программирования на языке Java и решены следующие задачи:

1. Изучены материалы, касающиеся темы классов на Java.
2. Приобретены практические навыки по определению класса, созданию полей и методов, вызове созданных методов.
3. Осуществлена программная реализация проекта

Данная работа не отличается графической составляющей и удобством, но программа читабельна и легко модифицируема.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Использованная литература:

1. Блинов И.Н., Романчик В.С. Java 2. Практическое руководство. - Мн.: Универсалпресс, 2005.
2. Блинов И.Н., Романчик В.С. Java. Промышленное программирование: практ. пособие. - Мн: УниверсалПресс, 2007.
3. Павловская Т.А., Щупак Ю.А. С++. Объектно-ориентированное программирование: Практикум. - СПб.: Питер, 2004.
4. Вязовик Н.А. Программирование на Java http://www.intiut.ru/department/pl/javapl.

Internet-ресурсы:

1. http://onlinelibrary.wiley.com - научные журналы издательства Wiley&Sons
2. http://www.sciencedirect.com/ - научные журналы издательства Elsevier
3. www.intuit.ru - национальный открытый университет
4. el.asu.tusu.ru - электронные курс по дисциплине Основы технологии Java 2.