Оглавление

[**Введение** 2](#_Toc106811892)

[**1 Разработка компьютерной игры** 3](#_Toc106811893)

[**1.1 Разработка технического задания** 3](#_Toc106811894)

[*1.1.1 Полное наименование системы и её условное обозначение* 3](#_Toc106811895)

[*1.1.2 Основание для разработки* 3](#_Toc106811896)

[*1.1.3 Назначение и цели создания системы* 3](#_Toc106811897)

[**1.2 Анализ технического задания** 3](#_Toc106811898)

[*1.2.1 Общие положения* 3](#_Toc106811899)

[*1.2.2 Требования к системе* 4](#_Toc106811900)

[**1.3 Стандарты** 4](#_Toc106811901)

[**1.4 Требования к техническим средствам** 5](#_Toc106811902)

[**2 Разработка модели предметной области** 6](#_Toc106811903)

[**2.1 Анализ предметной области** 6](#_Toc106811904)

[**2.2 Разработка структуры классов** 6](#_Toc106811905)

[**3 Разработка структуры приложения** 7](#_Toc106811906)

[**3.1 Разработка архитектуры** 7](#_Toc106811907)

[**3.2 Проектирование пользовательского интерфейса и взаимодействие с ним** 7](#_Toc106811908)

[**Заключение** 9](#_Toc106811909)

[**Список литературы** 12](#_Toc106811910)

[**Приложения** 1](#_Toc106811911)0

[**Приложение 1: Код класса SnakeGame.** 1](#_Toc106811912)0

[**Приложение 2: Код класса MainWindow.**](#_Toc106811913) 11

[**Приложение 3: Код класса GameField.** 15](#_Toc106811914)

**Введение**

В данной пояснительной записке рассматривается описание приложения «Snake Game» на основе объектно-ориентированного подхода.

При объектно-ориентированном подходе программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (или классов), отношений между ними, способов их взаимодействия и операций над объектами (или методов).

Несомненным преимуществом данного подхода является концептуальная близость к предметной области произвольной структуры и назначения. Механизм наследования атрибутов и методов позволяет строить производные понятия на основе базовых и таким образом создавать модель сколь угодно сложной предметной области с заданными свойствами.

Еще одним теоретически интересным и практически важным свойством объектно-ориентированного подхода является поддержка механизма обработки событий, которые изменяют атрибуты объектов и моделируют их взаимодействие в предметной области.

В отличие от других подходов к программированию, объектно-ориентированный подход требует глубокого понимания основных принципов, или концепций, на которых он базируется.

К числу основополагающих понятий ООП обычно относят абстракцию данных, наследование, инкапсуляцию и полиморфизм.

Объектно-ориентированное программирование в настоящее время является абсолютным лидером в области прикладного программирования.

В качестве основного инструмента разработки применяется NetBeans IDE 18. Язык программирования Java.

**1 Разработка компьютерной игры**

**1.1 Разработка технического задания**

*1.1.1 Полное наименование системы и её условное обозначение*

Полное наименование: Разработка приложения «Snake Game» для записи дел, требующих выполнения и учета процесса выполнения этих дел.

Условное обозначение системы: «Snake Game».

*1.1.2 Основание для разработки*

Основанием для разработки данной информационной системы является приказ по НГТУ на курсовое проектирование по дисциплине «Специалист по информационным технологиям».

*1.1.3 Назначение и цели создания системы*

1.1.3.1 Назначение системы

Программный продукт предназначен для игры.

1.1.3.2 Цели создания системы

Программный продукт разрабатывается с целью:

Отработать навыки приобретенные во время прохождения курса;

С помощью приложения пользователь сможет достигнуть вышеперечисленные цели.

1.1.3.3 Основные задачи разработки

1) Обеспечить генерацию «еды» на рандомном месте;

2) Обеспечить окончание игрового процесса при выходе за границы игрового поля или столкновении с туловищем;

3) Обеспечить возможность начать новую игру и отслеживать конечную длину «Змейки».

4) Создать удобный и понятный графический интерфейс;

**1.2 Анализ технического задания**

*1.2.1 Общие положения*

Согласно техническому заданию необходимо разработать приложение для компьютера, которое предназначено для игры.

*1.2.2 Требования к системе*

1.2.2.1 Требования к структуре и функционированию системы

Программный продукт, разрабатываемый в рамках курсового проекта, должен удовлетворять следующему перечню функциональных требований:

1. Удобный и понятный графический интерфейс
2. Вывод конечной длины

1.2.2.2 Входные данные

Входными данными при работе с программным продуктом должны быть нажатия на кнопку мыши (левая) и нажатия на клавиатуру.

1.2.2.3 Выходные данные

Выходными данными при работе программы являются данные, выводимые на экран пользователя.

1.2.2.4 Требования к надёжности

Система должна сохранять работоспособность и обеспечивать восстановление своих функций при возникновении внештатных ситуаций.

1.2.2.5 Требования к эргономике и технической эстетике

Подсистемы ввода данных, а также формирования и визуализации отчетности должны обеспечивать удобный для конечного пользователя интерфейс.

Главное окно программного продукта должно позволять пользователю увидеть текущие положение Змейки на экране.

**1.3 Стандарты**

Программный продукт разрабатывается на основании следующих государственных стандартов:

1. 2.103-68 ЕСКД. Стадии разработок
2. 2.104-68 ЕСКД. Основные надписи
3. 2.105-95 ЕСКД. Общие требования к текстовым документам
   1. -96 ЕСКД. Текстовые документы
4. 2.111- 68 ЕСКД. Общие требования к текстовым документам.
5. 2.118-73 ЕСКД. Техническое предложение
6. 2.120-73 ЕСКД. Технический проект
7. 2.316-68 ЕСКД. Правила нанесения на чертежах надписей, технических требований и таблиц
8. 7.1-2003 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.

**1.4 Требования к техническим средствам**

Для удобства работы система должна обеспечивать отображение *GUI* с расширенным набором пользовательских элементов, что соответствует платформе *Java*, *NetBeans* не ниже версии 8.0.2.

Минимальные технические характеристики компьютера, на котором гарантируется стабильная работа программы:

1. компьютер/процессор: компьютер-терминал с процессором класса *Intel* *Core i3* с тактовой частотой 2,3 ГГц и новее;
2. память: не менее 512 МБ ОЗУ;
3. монитор: монитор с разрешением 800x600 точек или более высоким;
4. наличие свободного дискового пространства на жёстком диске не менее 2 Гб.

**2 Разработка модели предметной области**

**2.1 Анализ предметной области**

Программируемое приложение должно позволять пользователю управлять Змейкой на экране, начинать новую игру.

Нефункциональные требования к программному средству:

1. эффективность: программа должна иметь минимальные требования к аппаратному обеспечению. Не должно требоваться дополнительных периферийных средств (сканер, дополнительные дисководы и т. д.);
2. понятность: программа должна быть интуитивно понятна, иметь удобный пользовательский интерфейс.

**2.2 Разработка структуры классов**

В соответствие с анализом предметной области можно выделить основные классы информационной системы и их атрибуты:

1. Главный класс *SnakeGame* содержит:

* Методы видимости рабочей формы;
* Методы видимости рабочей панели;

1. Класс *MainWindow* – содержит основное рабочее окно, создание объектов на форме, а также ссылку на класс *GameField*;
2. Класс *GameField* – содержит панель, на которой происходят все игровые действия;

**3 Разработка структуры приложения**

**3.1 Разработка архитектуры**

Так как «Snake Game», программа несложная, нам понадобятся несколько классов с описаниями форм и панель, которая описывает алгоритмы при совершении действия пользователем.

Перед пользователем сразу открывается игровое поле, то есть движущаяся «Змейка» (начальной длины: три) и клетка с едой. Управление движениями «Змейки» происходит с помощью стрелок на клавиатуре. Если «Змейка» достигает клетки с едой, она ее ест, и длина «Змейки» увеличивается. В то же время в рандомном месте игрового поля генерируется новая клетка с едой. Если голова Змейки сталкивается с туловищем или выходит за пределы игрового поля, то игра заканчивается. Пользователь видит надпись «Game over», итоговую длину «Змейки» и предложение начать новую игру путем нажатия клавиши «Enter».

**3.2 Проектирование пользовательского интерфейса и взаимодействие с ним**

В данном проекте для продуктивной работы реализован удобный пользовательский интерфейс (рис. 1, 2).

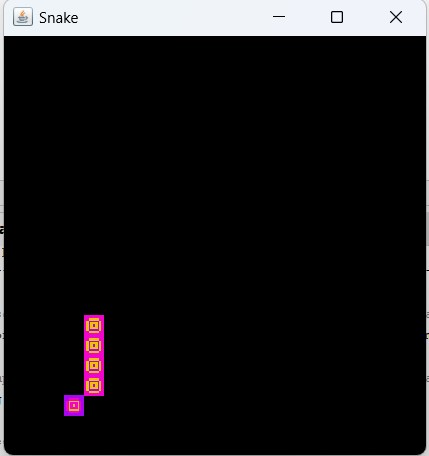


Рис. 1 – Интерфейс игрового поля.

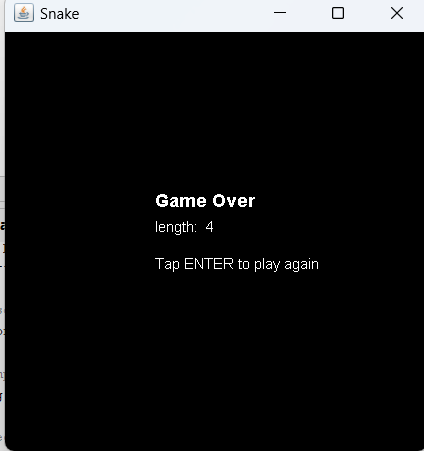


Рис. 2 – Интерфейс приложения по завершении игры.

Основной сценарий работы пользователя с приложением:

1. Пользователь начинает игру и управляет «Змейкой» с помощью стрелок на клавиатуре.
2. После завершения игры пользователь видит конечную длину и при необходимости может начать новую игру.

**Заключение**

В данной курсовой работе было разработано приложение «Snake Game» на основе объектно-ориентированного подхода. Этот метод позволяет обеспечить разделение данных, управление доступом к ним, а также расширяемость и модифицируемость.

В проекте удалось реализовать все поставленные задачи:

1. Создать удобный и понятный графический интерфейс;
2. Обеспечить генерацию «еды» на рандомном месте;
3. Обеспечить окончание игрового процесса при выходе за границы игрового поля или столкновении с туловищем;
4. Обеспечить возможность начать новую игру и отслеживать конечную длину «Змейки».

Таким образов, было реализовано приложение «Snake Game», которое позволяет пользователю весело провести время, отдохнуть.

**Приложения**

**Приложение 1: Код класса SnakeGame.**

package com.mycompany.snakegame;

/\*\*

\*

\* @author Egor

\*/

public class SnakeGame {

public static void main(String[] args) {

MainWindow NJF = new MainWindow();

NJF.setVisible(true);

NJF.setTitle("Snake");

GameField gf = new GameField();

gf.setVisible(true);

}

}

**Приложение 2: Код класса MainWindow.**

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template

\*/

package com.mycompany.snakegame;

import javax.swing.WindowConstants;

/\*\*

\*

\* @author Egor

\*/

public class MainWindow extends javax.swing.JFrame {

/\*\*

\* Creates new form MainWindow

\*/

public MainWindow() {

//initComponents();

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

setSize(352,373);

setLocation(400,400);

add(new GameField());

setVisible(true);

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 100, Short.MAX\_VALUE)

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 100, Short.MAX\_VALUE)

);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 400, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 300, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MainWindow.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MainWindow.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MainWindow.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MainWindow.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new MainWindow().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JPanel jPanel1;

// End of variables declaration

}

**Приложение 3: Код класса GameField.**

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JPanel.java to edit this template

\*/

package com.mycompany.snakegame;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Image;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.util.Random;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.Timer;

/\*\*

\*

\* @author Egor

\*/

public class GameField extends javax.swing.JPanel implements ActionListener {

private final int SIZE = 320;

private final int DOT\_SIZE = 16;

private final int ALL\_DOTS = 400;

private Image dot;

private Image apple;

private int appleX;

private int appleY;

private int[] x = new int[ALL\_DOTS];

private int[] y = new int[ALL\_DOTS];

private int dots;

private Timer timer;

private boolean left = false;

private boolean right = true;

private boolean up = false;

private boolean down = false;

private boolean inGame = true;

/\*\*

\* Creates new form GameField

\*/

public GameField() {

setBackground(Color.black);

loadImages();

initGame();

addKeyListener(new FieldKeyListener());

setFocusable(true);

}

public void initGame(){

dots = 3;

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

x[i] = 48 - i\*DOT\_SIZE;

y[i] = 48;

}

timer = new Timer(150,this);

timer.start();

createApple();

}

public void createApple(){

appleX = new Random().nextInt(20)\*DOT\_SIZE;

appleY = new Random().nextInt(20)\*DOT\_SIZE;

}

public void loadImages(){

ImageIcon iia = new ImageIcon("apple.png");

apple = iia.getImage();

ImageIcon iid = new ImageIcon("dot.png");

dot = iid.getImage();

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

if(inGame){

g.drawImage(apple,appleX,appleY,this);

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

g.drawImage(dot,x[i],y[i],this);

}

} else{

timer.stop();

g.setColor(Color.white);

String str = "Game Over";

String str1 = "length: " + dots;

String str2 = "Tap ENTER to play again";

g.drawString(str1,120,SIZE/2);

g.drawString(str2,120,SIZE/2+30);

Font f = new Font("Arial",Font.BOLD,14);

g.setFont(f);

g.drawString(str,120,SIZE/2-20);

}

}

public void move(){

for (int i = dots; i > 0; i--) {

x[i] = x[i-1];

y[i] = y[i-1];

}

if(left){

x[0] -= DOT\_SIZE;

}

if(right){

x[0] += DOT\_SIZE;

} if(up){

y[0] -= DOT\_SIZE;

} if(down){

y[0] += DOT\_SIZE;

}

}

public void checkApple(){

if(x[0] == appleX && y[0] == appleY){

dots++;

createApple();

}

}

public void checkCollisions(){

for (int i = dots; i >0 ; i--) {

if(dots>4 && x[0] == x[i] && y[0] == y[i]){

inGame = false;

}

}

if(x[0]>SIZE){

inGame = false;

}

if(x[0]<0){

inGame = false;

}

if(y[0]>SIZE){

inGame = false;

}

if(y[0]<0){

inGame = false;

}

}

@Override

public void actionPerformed(ActionEvent e) {

if(inGame){

checkApple();

checkCollisions();

move();

}

repaint();

}

class FieldKeyListener extends KeyAdapter{

@Override

public void keyPressed(KeyEvent e) {

super.keyPressed(e);

int key = e.getKeyCode();

if(key == KeyEvent.VK\_LEFT && !right){

left = true;

up = false;

down = false;

}

if(key == KeyEvent.VK\_RIGHT && !left){

right = true;

up = false;

down = false;

}

if(key == KeyEvent.VK\_UP && !down){

right = false;

up = true;

left = false;

}

if(key == KeyEvent.VK\_DOWN && !up){

right = false;

down = true;

left = false;

}

if(key == KeyEvent.VK\_ENTER && !inGame ){

inGame = true;

left = false;

right = true;

up = false;

down = false;

initGame();

}

}

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);

this.setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 400, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 300, Short.MAX\_VALUE)

);

}// </editor-fold>

/\* @Override

public void actionPerformed(ActionEvent e) {

throw new UnsupportedOperationException("Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody

}

\*/

// Variables declaration - do not modify

// End of variables declaration

}