**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Московский технический университет связи и информатики**

Кафедра «Математическая кибернетика и информационные технологии»

**Курсовая работа**

по дисциплине «Java-программирование»

на тему «Разработка приложения-игры Крестики-Нолики»

Выполнил: студенты группы БСУ1901  
Власов А.М.

Козлов А.Э

Проверил: старший преподаватель кафедры МКиИТ

Мосева М.С.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2021

Оглавление

Введение3

Техническое задание на разработку4

Описание программной реализации5-9

Описание процесса тестирования10-12

Руководство пользователя13

Заключение13

Список использованных источников14

Введение

Актуальность моей работы определяется тем, что стремительное развитие компьютерной техники в последние годы, появление мощнейших графических ускорителей и центральных процессоров способствовало не менее бурному развитию индустрии компьютерных игр. Выдающиеся разработки этой отрасли – это сложнейшие программы, как правило, с очень высокими требованиями к аппаратной части компьютера. Однако для возможности отдохнуть в перерыве от выполнения какой-либо работы оператору компьютера не всегда требуется новейшая компьютерная игра, а зачастую использовать её не позволяет маломощное оборудование офисного компьютера. Именно этой цели – отдыху от монотонной работы служит разработанная в рамках данного курсового проекта программа.

**Целью данной курсовой работы** является "создание игрового приложения "крестики-нолики"".

**Техническое задание на разработку**

**Техническое задание**: разработать приложение, анимацию, игру или другое приложение для демонстрации своих знаний об основах программирования на Java.

Требования к разрабатываемому приложению:

* включать хотя бы 2 класса, не считая тестовый класс;
* инкапсулировать эти и дополнительные классы;
* предоставлять хотя бы 2 поля и 2 метода в каждом классе, не включая методы получения или установки;
* включать массив или список ArrayList;
* быть способной принять ввод пользователя хотя бы один раз; всегда принимать ввод пользователя без сбоя.

**Описание программной реализации**

1. Класс GameView, в котором задается шрифт кнопок и полей, иконка приложения, инициализация игрового окна (задание размера окна, запрет на изменяемость размера окна, видимость окна, заголовок окна, логотип, задание кнопок)

package com.week4;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class GameView extends JFrame {  
 private GameController gameController;  
 private JButton[][] fieldButton;  
 private JButton playButton;  
 private JLabel messageLabel;  
 private JLabel infoLabel;  
 Font fieldFont = new Font("TimesRoman", Font.*BOLD*, 25);  
 Font buttonFont = new Font("TimesRoman", Font.*BOLD*, 40);  
 ImageIcon img = new ImageIcon("icon.png");  
  
 public GameView(GameController gameController) {  
 this.gameController = gameController;  
 }  
  
 public void init() {  
 setSize(600, 350);  
 setIconImage(img.getImage());  
 setTitle("Крестики-Нолики");  
 setVisible(true);  
 setLayout(null);  
 setResizable(false);  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
  
 fieldButton = new JButton[3][3];  
 playButton = new JButton("Играть");  
 infoLabel = new JLabel("Крестики-Нолики");  
 messageLabel = new JLabel("");  
  
 for (int i = 0; i < 3; i++)  
 for (int j = 0; j < 3; j++) {  
 fieldButton[i][j] = new JButton("-");  
 fieldButton[i][j].setFont(buttonFont);  
 fieldButton[i][j].setBounds(i \* 100,j \* 100,100,100);  
 fieldButton[i][j].setEnabled(false);  
 add(fieldButton[i][j]);  
 }  
  
 playButton.setBounds(420, 280, 80, 20);  
 add(playButton);  
  
 infoLabel.setBounds(350, 0, 220, 30);  
 infoLabel.setFont(fieldFont);  
 add(infoLabel);  
  
 messageLabel.setBounds(350, 30, 200, 30);  
 add(messageLabel);  
  
 playButton.addActionListener(e ->{  
 gameController = new GameController();  
 clearBtn();  
 enableBtn(true);  
 infoLabel.setText("Ход игрока " + gameController.playerTurn());  
 });  
  
 fieldButton[0][0].addActionListener(e -> {  
 if (gameController.turn(0,0, fieldButton[0][0], infoLabel, messageLabel))  
 enableBtn(false);  
 });  
 fieldButton[0][1].addActionListener(e -> {  
 if (gameController.turn(0,1, fieldButton[0][1], infoLabel, messageLabel))  
 enableBtn(false);  
 });  
 fieldButton[0][2].addActionListener(e -> {  
 if (gameController.turn(0,2, fieldButton[0][2], infoLabel, messageLabel))  
 enableBtn(false);  
 });  
 fieldButton[1][0].addActionListener(e -> {  
 if (gameController.turn(1,0, fieldButton[1][0], infoLabel, messageLabel))  
 enableBtn(false);  
 });  
 fieldButton[1][1].addActionListener(e -> {  
 if (gameController.turn(1,1, fieldButton[1][1], infoLabel, messageLabel))  
 enableBtn(false);  
 });  
 fieldButton[1][2].addActionListener(e -> {  
 if (gameController.turn(1,2, fieldButton[1][2], infoLabel, messageLabel))  
 enableBtn(false);  
 });  
 fieldButton[2][0].addActionListener(e -> {  
 if (gameController.turn(2,0, fieldButton[2][0], infoLabel, messageLabel))  
 enableBtn(false);  
 });  
 fieldButton[2][1].addActionListener(e -> {  
 if (gameController.turn(2,1, fieldButton[2][1], infoLabel, messageLabel))  
 enableBtn(false);  
 });  
 fieldButton[2][2].addActionListener(e -> {  
 if (gameController.turn(2,2, fieldButton[2][2], infoLabel, messageLabel))  
 enableBtn(false);  
 });  
 }  
  
 public void enableBtn(boolean enableBtn) {  
 for (int i = 0; i < 3; i++)  
 for (int j = 0; j < 3; j++) {  
 fieldButton[i][j].setEnabled(enableBtn);  
 }  
 }  
  
 public void clearBtn() {  
 for (int i = 0; i < 3; i++)  
 for (int j = 0; j < 3; j++) {  
 fieldButton[i][j].setText("-");  
 }  
 }  
}

2. Класс GameModel, который содержит метод создания нового игрового поля через двумерный массив, метод проверки игрока на выигрыш.

package com.week4;  
public class GameModel {  
 private final char[][] field = new char[3][3];  
 private int turnCount;  
 private char player;  
 public int getTurnCount() {  
 return turnCount;  
 }  
 public char getPlayer() {  
 return player;  
 }  
 public void doTurn(){  
 turnCount ++;  
 }  
  
 public GameModel(){  
 for (int i = 0; i < 3; i++){  
 for (int j = 0; j < 3; j++){  
 field[i][j] = '-';  
 }  
 }  
 player = 'X';  
 turnCount = 0;  
 }  
  
 public boolean add(int i, int j){  
 if (i < 3 && j < 3 && field[i][j] == '-'){  
 field[i][j] = player;  
 return true;  
 }  
 else return false;  
 }  
  
 public char win(){  
 for (int i = 0; i < 3; i++){  
 if (field[i][0] + field[i][1] + field[i][2] == 264 || field[0][i] + field[1][i] + field[2][i] == 264)  
 return 'X';  
 else if (field[i][0] + field[i][1] + field[i][2] == 144 || field[0][i] + field[1][i] + field[2][i] == 144)  
 return '0';  
 }  
 if (field[0][0] + field[1][1] + field[2][2] == 264 || field[0][2] + field[1][1] + field[2][0] == 264)  
 return 'X';  
 else if (field[0][0] + field[1][1] + field[2][2] == 144 || field[0][2] + field[1][1] + field[2][0] == 144)  
 return '0';  
 return '-';  
 }  
  
 public void switchPlayer(){  
 if (player == 'X')  
 player = '0';  
 else  
 player = 'X';  
 }  
}

3. Класс GameController, который содержит метод переключения между игроками после хода игрока, метод завершения игры с выводом результата игры (победы одного из игроков или ничьи)

package com.week4;  
  
import javax.swing.\*;  
  
public class GameController {  
 GameModel gameModel;  
  
 public GameController() {  
 gameModel = new GameModel();  
 }  
  
 public char playerTurn() {  
 return gameModel.getPlayer();  
 }  
  
 public boolean turn(int i, int j, JButton button, JLabel info, JLabel message) {  
 if (gameModel.add(i,j)) {  
 button.setText(String.*valueOf*(gameModel.getPlayer()));  
 gameModel.switchPlayer();  
 gameModel.doTurn();  
 info.setText("Ход игрока " + gameModel.getPlayer());  
 message.setText("");  
 return endGame(info);  
 }  
 else {  
 message.setText("Ячейка занята!");  
 return false;  
 }  
 }  
  
 public boolean endGame(JLabel info) {  
 if (gameModel.getTurnCount() == 9){  
 info.setText("Ничья!");  
 return true;  
 }  
 else if (gameModel.win() != '-'){  
 gameModel.switchPlayer();  
 info.setText("Победил игрок " + gameModel.getPlayer() + "!");  
 return true;  
 }  
 return false;  
 }  
}

4. Класс Main, который запускает нашу игру:

package com.week4;  
  
import javax.swing.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 GameController gameController = new GameController();  
 GameView gameView = new GameView(gameController);  
 SwingUtilities.*invokeLater*(gameView::init);  
 }  
}

**Описание процесса тестирования**

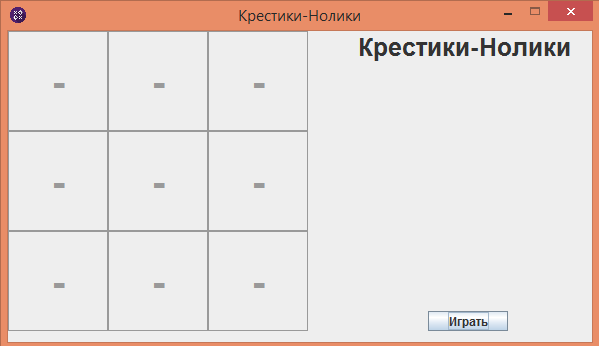
****

Рисунок 1 – Окно при запуске программы

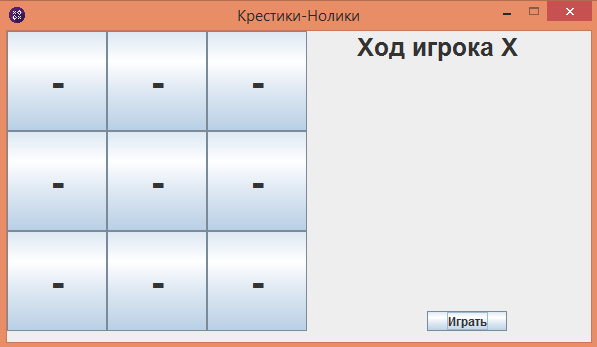


Рисунок 2 – Нажатие кнопки Играть

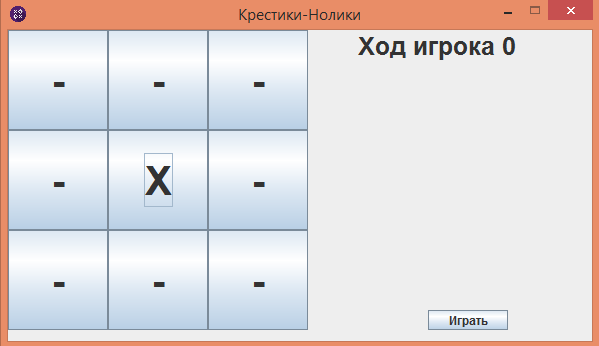


Рисунок 3 – Первый ход

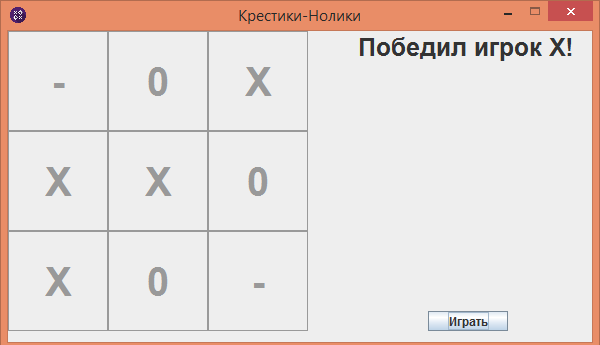


Рисунок 4 – Победа игрока X

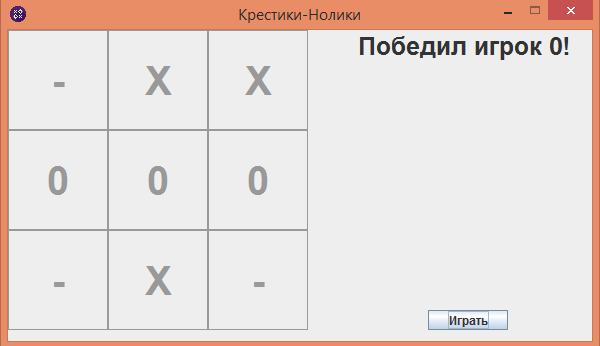
****

Рисунок 5 – Победа игрока O

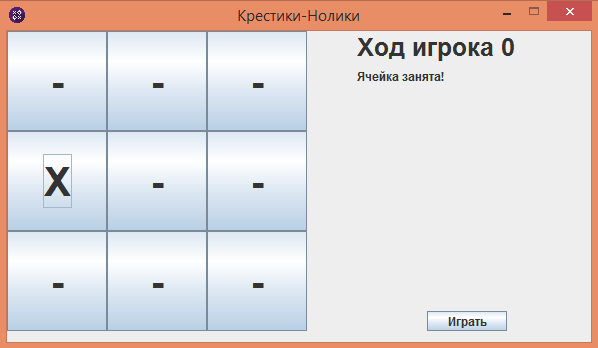
****

Рисунок 6 – При нажатии на занятую ячейку программа выдает сообщение “Ячейка занята”

**Руководство пользователя**

1. Для запуска игры необходимо предварительно скачать файлы игры и установить Java и среду разработки на свой ПК.

2. Запускаем приложение “IntelliJ IDEA”, затем открываем наш проект с помощью кнопок “File”и “Open”, далее вводим путь к папке с нашей игрой.

3. Нажимаем кнопку “Run” и наш проект запущен

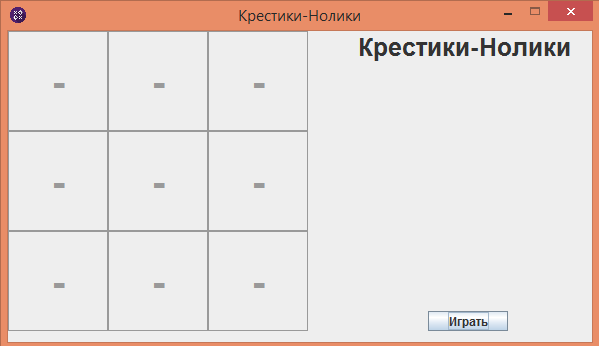


Рисунок 7 – Приложение запущено

**Заключение**

Мы проанализировали актуальность данной работы. Все поставленные задачи были выполнены, и разработка популярной игры «Крестики-нолики» прошла успешно.

**Список использованных источников**

1. Герберт, Шилдт Java. Руководство для начинающих / Шилдт Герберт. - М.: Диалектика / Вильямс, 2014.

2. Давыдов, Станислав IntelliJ IDEA. Профессиональное программирование на Java / Станислав Давыдов , Алексей Ефимов. - М.: БХВ-Петербург, 2015.