Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**Дисциплина: Программирование для мобильных платформ**

Работу выполнила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. Н. Баева

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т. А. Приходько

**Тема работы:** Многопоточность.

**Ход работы:**

В лабораторной работе необходимо было разработать многопоточную игру “Крестики-Нолики” с графическим интерфейсом.

В программе создается окно с игровым полем размером 3x3, где каждая клетка является кнопкой (для этого используется менеджер размещения GridLayout). Игроки по очереди нажимают на кнопки, чтобы поставить крестик или нолик. Если один из игроков выигрывает, то появляется сообщение с результатом игры. Если ничья, то появляется соответствующие сообщение о ничьей.

Код начинается с объявления класса TicTacToe, который наследуется от класса JFrame. В конструкторе класса создается окно размером 500x500 пикселей с заголовоком "Tic Tac Toe". Затем создается игровое поле размером 3x3, которое представляет собой массив кнопок типа JButton. Каждая кнопка имеет пустой текст и обработчик событий ActionListener. Фрагмент кода, выполняющий этот, представлен на рисунке 1.



Рисунок 1 – Инициализация окна и игрового поля.

Обработчик событий ActionListener вызывается при нажатии на кнопку. Если текст кнопки пустой, то в зависимости от текущего игрока на кнопку ставится крестик или нолик. Затем проверяется, выиграл ли один из игроков или наступила ничья. Случай выигрыша игрока и ничьи реализованы в отдельных методах checkWin() и checkDraw() (рисунок 2).

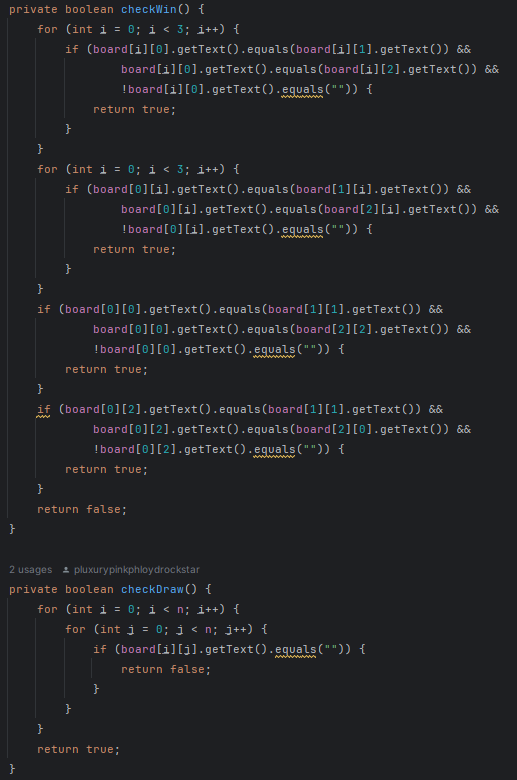


Рисунок 2 – Проверка выигрыша игрока и ничьи.

Если один из игроков выиграл, то появляется сообщение с результатом игры и игра заканчивается. Если наступила ничья, то появляется сообщение о ничьей и игра заканчивается. Затем текущий игрок меняется на другого игрока. В этом фрагменте кода также используется блокировка lock, которая позволяет игрокам играть в игру параллельно, не блокируя друг друга. Когда один игрок сделает ход, он уведомляет другого игрока, что тот может сделать свой ход, вызывая метод notifyAll() на объекте блокировки lock.

В конце обработчика событий создаются два игрока player1 и player2, которые запускаются в отдельных потоках. Для этого определён отдельный класс Player (рисунок 3).



Рисунок 3 – Класс Player.

В этом классе определен метод run(), который выполняется в отдельном потоке. В нем работает бесконечный цикл, который ожидает уведомления от другого потока. Когда другой поток уведомляет этот поток, он проверяет, выиграл ли игрок, и если да, выводит сообщение об этом. Если ничья, то выводится сообщение о ничьей. Для того, чтобы ожидать уведомления, метод run() использует блокировку lock и вызывает метод wait(). Это заставляет поток ожидать, пока другой поток не вызовет метод notify() или notifyAll() на том же объекте блокировки. Если поток получает уведомление, он продолжает выполнение и проверяет, выиграл ли игрок или произошла ничья. Таким образом, данный метод run() позволяет игрокам играть в игру параллельно, не блокируя друг друга.

Диаграмма классов получившейся игры представлена на рисунке 4.

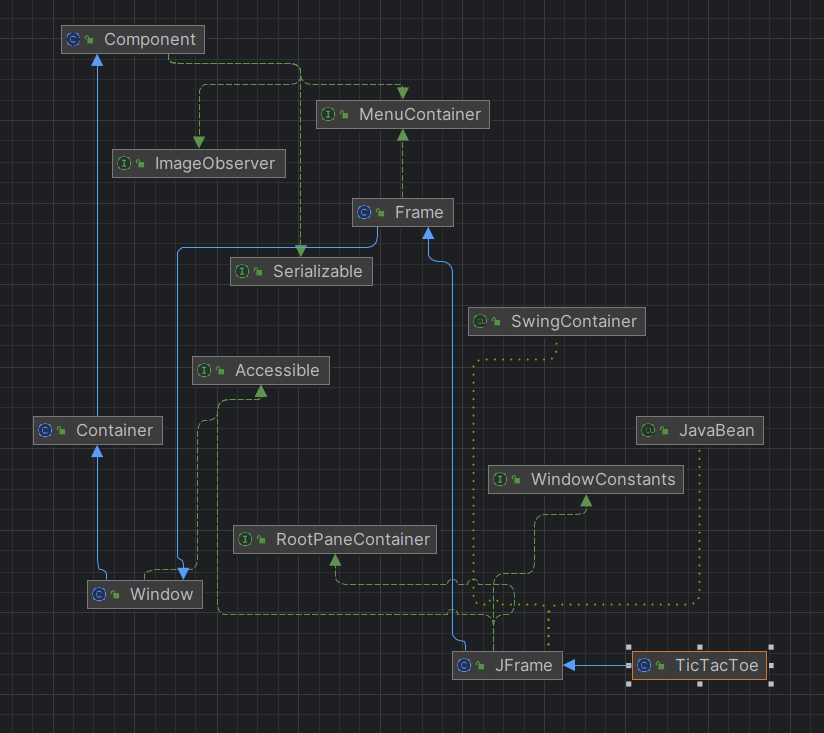


Рисунок 4 – Диаграмма классов.

**Листинг программ**

Файл TicTacToe.java

package Game\_Tic\_Tac\_Toe;  
  
import java.awt.\*;  
import java.awt.event.\*;  
import javax.swing.\*;  
  
public class TicTacToe extends JFrame {  
 private JButton[][] board;  
 private int n = 3;  
 private int i, j;  
 private int currentPlayer = 1;  
 private Thread player1, player2;  
 private Object lock = new Object();  
  
 public TicTacToe() {  
 super("Tic Tac Toe");  
 setSize(500, 500);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setLayout(new GridLayout(n, n));  
 board = new JButton[n][n];  
 for (i = 0; i < n; i++) {  
 for (j = 0; j < n; j++) {  
 board[i][j] = new JButton("");  
 board[i][j].addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 JButton buttonClicked = (JButton) e.getSource();  
 if (buttonClicked.getText().equals("")) {  
 if (currentPlayer == 1) {  
 buttonClicked.setText("X");  
 } else {  
 buttonClicked.setText("O");  
 }  
 if (checkWin()) {  
 JOptionPane.*showMessageDialog*(null, (currentPlayer == 1 ? "X" : "O") + " выйграл!");  
 System.*exit*(0);  
 }  
 if (checkDraw()) {  
 JOptionPane.*showMessageDialog*(null, "Ничья!");  
 System.*exit*(0);  
 }  
 if (currentPlayer == 1) {  
 currentPlayer = 2;  
 } else {  
 currentPlayer = 1;  
 }  
 // когда один игрок сделал ход, он уведомляет другого игрока, что тот может сделать свой ход,  
 // вызов метод notifyAll() на объекте блокировки lock  
 synchronized (lock) {  
 lock.notifyAll();  
 }  
 }  
 }  
 });  
 add(board[i][j]);  
 }  
 }  
 setVisible(true);  
 // создание потоков для каждого игрока  
 player1 = new Thread(new Player(1, lock));  
 player2 = new Thread(new Player(2, lock));  
 // запуск созданных потоков  
 player1.start();  
 player2.start();  
 }  
  
 private boolean checkWin() {  
 for (int i = 0; i < 3; i++) {  
 if (board[i][0].getText().equals(board[i][1].getText()) &&  
 board[i][0].getText().equals(board[i][2].getText()) &&  
 !board[i][0].getText().equals("")) {  
 return true;  
 }  
 }  
 for (int i = 0; i < 3; i++) {  
 if (board[0][i].getText().equals(board[1][i].getText()) &&  
 board[0][i].getText().equals(board[2][i].getText()) &&  
 !board[0][i].getText().equals("")) {  
 return true;  
 }  
 }  
 if (board[0][0].getText().equals(board[1][1].getText()) &&  
 board[0][0].getText().equals(board[2][2].getText()) &&  
 !board[0][0].getText().equals("")) {  
 return true;  
 }  
 if (board[0][2].getText().equals(board[1][1].getText()) &&  
 board[0][2].getText().equals(board[2][0].getText()) &&  
 !board[0][2].getText().equals("")) {  
 return true;  
 }  
 return false;  
 }  
  
 private boolean checkDraw() {  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 if (board[i][j].getText().equals("")) {  
 return false;  
 }  
 }  
 }  
 return true;  
 }  
  
 private class Player implements Runnable {  
 private int player;  
 private Object lock;  
  
 public Player(int player, Object lock) {  
 this.player = player;  
 this.lock = lock;  
 }  
  
 @Override  
 // данный метод run() позволяет игрокам играть в игру параллельно, не блокируя друг друга  
 public void run() {  
 while (true) {  
 synchronized (lock) {  
 try {  
 lock.wait(); // поток ожидает  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 if (checkWin()) {  
 JOptionPane.*showMessageDialog*(null, (currentPlayer == 1 ? "X" : "O") + " выйграл!");  
 System.*exit*(0);  
 }  
 if (checkDraw()) {  
 JOptionPane.*showMessageDialog*(null, "Ничья!");  
 System.*exit*(0);  
 }  
 }  
 }  
 }  
  
 }  
  
 public static void main(String[] args) {  
 new TicTacToe();  
 }  
}