

# Оглавление

Оглавление...............................................................................................................4

Введение...................................................................................................................5

1. Постановка задачи...............................................................................................6

2. Выбор решения....................................................................................................7

3. Алгоритм работы программы.............................................................................8

4. Описание способа организации пользовательского интерфейса..................10

5. Отладка и тестирование.................................................................................13

Заключение.............................................................................................................14

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.................................................15

ПРИЛОЖЕНИЕ А. UML-диаграммы..................................................................16

ПРИЛОЖЕНИЕ Б. Листинг.......................................................................20

# Введение

В настоящей пояснительной записке приведено описание алгоритма программы, реализующей игру «Крестики-нолики». Программа разработана и отлажена на языке программирования Java, среда разработки – IntelliJ IDEA

# Постановка задачи

Требуется разработать программу для клиент-серверной работы, реализующей игру «Крестики-нолики».

Операционная система: Windows;  
Среда разработки: IntelliJ IDEA 2022;  
Язык программирования Java.

1. Java Collections Framework

2) JDBC

3) Механизм обработки исключительных ситуаций

4) JavaStreamAPI

5) Java Multithreading

6) Сетевое взаимодействие

# 2. Выбор решения

**2.1 Сервер**

Сервер для программы написан на языке Java

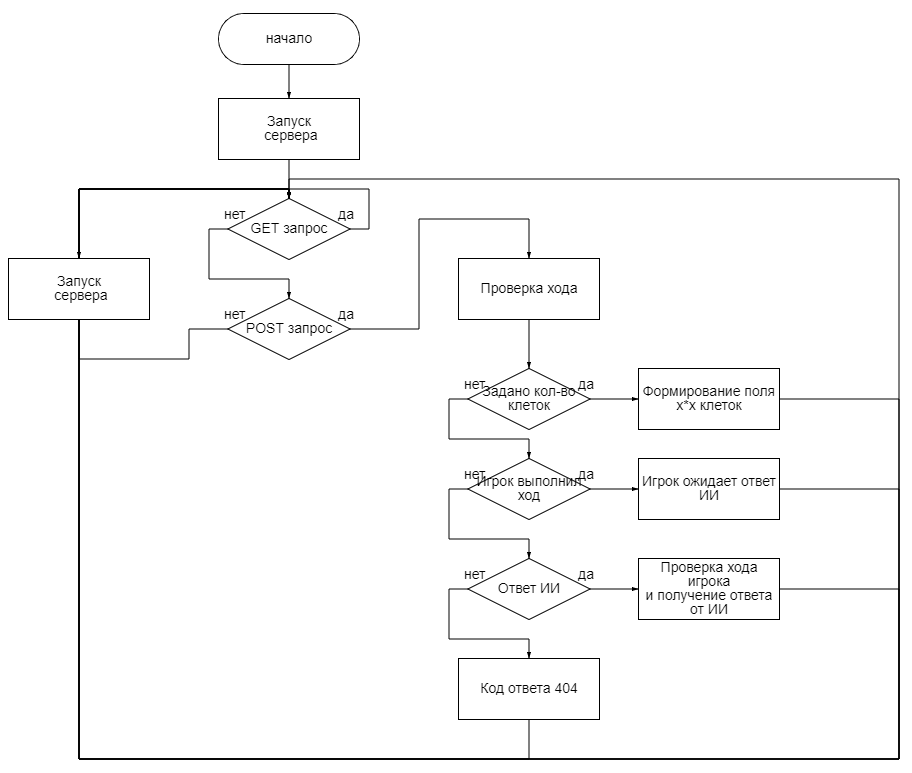


Рисунок 1 - Алгоритм работы сервера

**2.2 Клиент**

Клиент представлен в виде кода на языке Java.

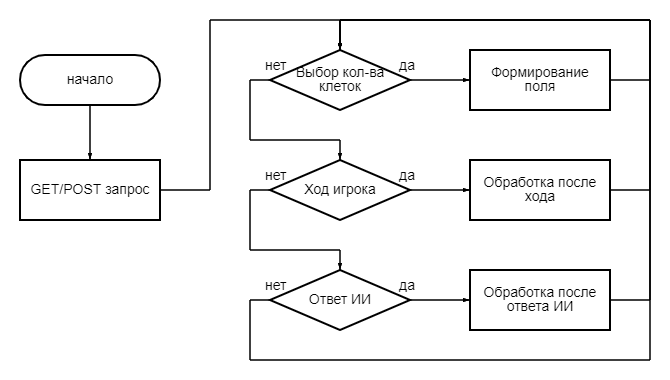


Рисунок 2 - Алгоритм работы клиента

# 3.Описание программы

**3.2 Клиент**

Класс **Game:**

Функция actionPerformed () проверяет выполнено ли действие

Функция drawWin() прописывает условия ничья/победа.

Функция checkWin() проверяет выполнено ли условие победы.

Функция draw()проверяет выполнено ли условие ничьей.

Функции getColls(), getDiagSideTop(), getDiagMainBottom(), getDiagMainTop(), getDiagSideBottom(), getDiagMain(), getDiagSide() нужны для получения стороны диаграммы.

Функция computeMove() для вычисления ряда крестиков и ноликов.

Класс **Cell:**

Класс **InfoDialog:**

Класс **State:**

**2.1 Сервер**

Класс **Cell:**

Метод Cell() задает размер поля

Метод getX() принимает ход X

Метод getY() принимает ход Y

Метод getState() принимает состояние игры

Класс **PlayingField:**

Метод PlayingField() позволяет менять размер поля в меню

getRows() получает строку

getSize() получает размер

isFull() выполняет действие, если поле заполнено

# 4.Описание способа организации пользовательского интерфейса

Для написания пользовательского интерфейса был использован язык Java, среда разработки: IntelliJ IDEA 2022.

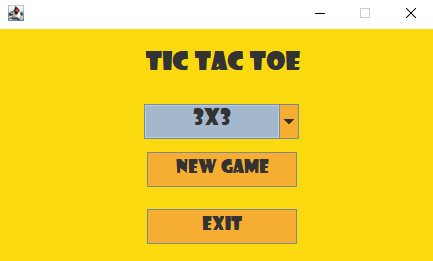


Рисунок 3 - Вид меню игры

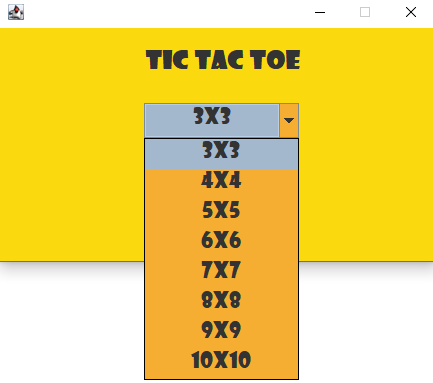


Рисунок 4 – Вид меню при выборе размера поля

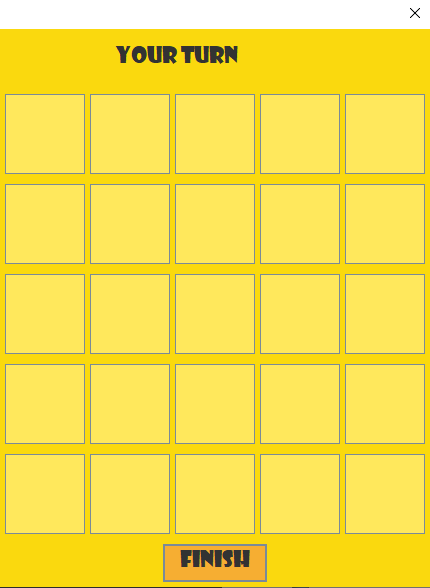


Рисунок 5 – Вид игрового поля после выбора его размера

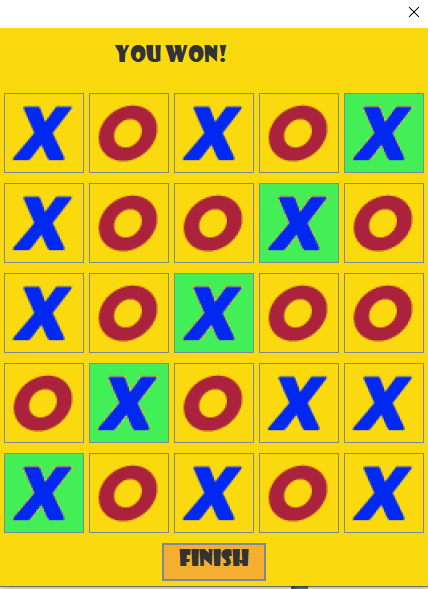


Рисунок 6 - Сообщение о победе

# 5. Отладка и тестирование

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | Ожидаемый результат | Полученный результат | Вывод |
| Попытка выполнить не свой ход | Ничего не происходит | Невозможно походить за компьютер | Верно |
| Попытка выполнить ход на заполненную ячейку | Ничего не происходит | Невозможно пойти на заполненную клетку | Верно |
| После закрытия игры выйти в меню | Выход в меню игры после нажатия на кнопку «finish» | Выход в меню игры | Верно |
| Попытка запустить игру без предварительного запуска сервера | Ничего не происходит | Игра не запускается без сервера | Верно |

По результатам тестирования можно сделать вывод, что программа отрабатывает верно.

# Заключение

Таким образом, в данной работе была спроектирована и реализована программа сервер и клиент для игры «Крестики-нолики».

При выполнении данной курсовой работы были получены навыки разработки программ в Java. Приобретены навыки по созданию клиент-серверных приложений на языке Java.

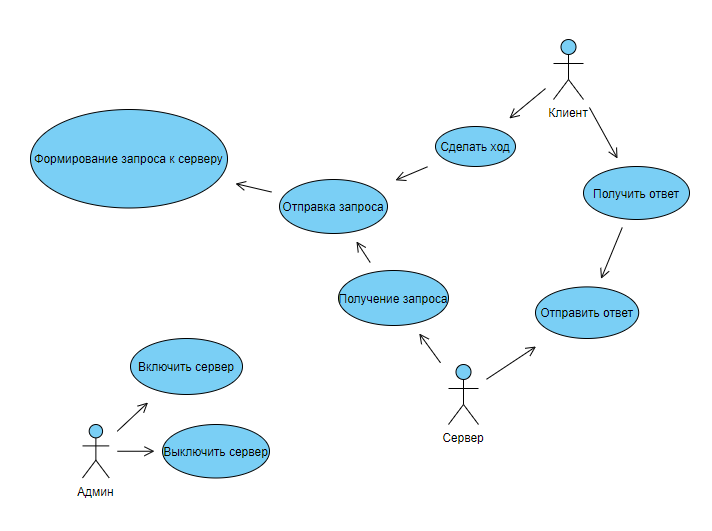
# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Сайт со статьями [Электронный ресурс]: <https://habr.com>. - Режим доступа: <https://habr.com/ru/post/441150/> - Загл. с экрана.

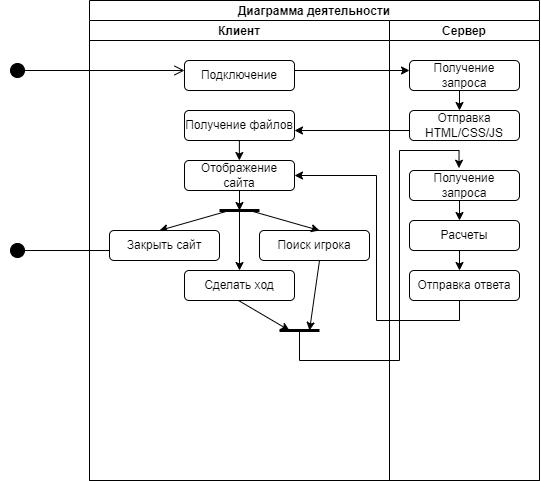
2 Форум программистов и сисадминов[Электронный ресурс]: <https://stackoverflow.com>. - Режим доступа: <https://stackoverflow.com/questions/3732109/simple-http-server-in-java-using-only-java-se-api> - Загл. с экрана.

# ПРИЛОЖЕНИЕ А. UML-диаграммы.

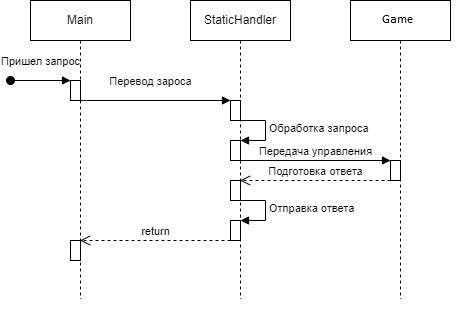
# Диаграмма вариантов использования



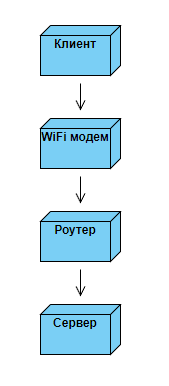
# Диаграмма деятельности



# Диаграмма последовательностей



# Диаграмма развёртывания



# ПРИЛОЖЕНИЕ Б. Текст программы.

# Клиент

# Листинг Game.java

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.game;  
  
import com.game.rmi.GameService;  
import java.awt.Color;  
import java.awt.Component;  
import java.awt.Dimension;  
import java.awt.FlowLayout;  
import java.awt.Font;  
import java.awt.GridLayout;  
import java.awt.Image;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.IOException;  
import java.net.MalformedURLException;  
import java.rmi.Naming;  
import java.rmi.NotBoundException;  
import java.rmi.RemoteException;  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
import javax.imageio.ImageIO;  
import javax.swing.ImageIcon;  
import javax.swing.JButton;  
import javax.swing.JDialog;  
import javax.swing.JLabel;  
import javax.swing.JOptionPane;  
import javax.swing.JPanel;  
import javax.swing.SwingUtilities;  
import javax.swing.border.EmptyBorder;  
  
public class Game extends JDialog implements ActionListener {  
 private final JPanel gridPanel = new JPanel();  
 private final JPanel topPanel = new JPanel();  
 JLabel info = new JLabel();  
 JButton[][] buttons;  
 int size = 0;  
 PlayingField fields;  
 Image cross;  
 Image zerro;  
 Image crossWin;  
 Image zerroWin;  
 Image white;  
 static int sizeCell = 80;  
 boolean gameOver = false;  
 boolean isComputer = false;  
 GameService service;  
  
 public Game(PlayingField fields) {  
 try {  
 this.service = (GameService)Naming.lookup("//localhost/GameService");  
 } catch (RemoteException | NotBoundException | MalformedURLException var7) {  
 JOptionPane.showMessageDialog((Component)null, "Не могу подключиться к серверу");  
 this.dispose();  
 }  
  
 if (fields.getSize() > 6) {  
 sizeCell = 75;  
 }  
  
 try {  
 this.cross = ImageIO.read(this.getClass().getResource("/resources/img/cross.png")).getScaledInstance(sizeCell, sizeCell, 4);  
 this.zerro = ImageIO.read(this.getClass().getResource("/resources/img/zero.png")).getScaledInstance(sizeCell, sizeCell, 4);  
 this.crossWin = ImageIO.read(this.getClass().getResource("/resources/img/crosswin.png")).getScaledInstance(sizeCell, sizeCell, 4);  
 this.zerroWin = ImageIO.read(this.getClass().getResource("/resources/img/zerowin.png")).getScaledInstance(sizeCell, sizeCell, 4);  
 this.white = ImageIO.read(this.getClass().getResource("/resources/img/white.png")).getScaledInstance(sizeCell, sizeCell, 4);  
 } catch (IOException var6) {  
 var6.printStackTrace();  
 }  
  
 this.setModal(true);  
 this.setResizable(false);  
 this.fields = fields;  
 this.size = fields.getSize();  
 this.setBounds(100, 100, 212, 261);  
 this.gridPanel.setLayout(new GridLayout(this.size, 10, this.size, 10));  
 this.gridPanel.setBorder(new EmptyBorder(5, 5, 5, 5));  
 this.gridPanel.setBackground(new Color(250, 217, 14));  
 this.getContentPane().add(this.gridPanel, "Center");  
 this.topPanel.setLayout(new FlowLayout(1, 10, 10));  
 this.info.setFont(new Font("Showcard Gothic", 0, 22));  
 this.info.setPreferredSize(new Dimension(200, 40));  
 this.info.setText("YOUR TURN");  
 this.topPanel.add(this.info);  
 this.topPanel.setBackground(new Color(250, 217, 14));  
 this.getContentPane().add(this.topPanel, "North");  
 Dimension btSize = new Dimension(sizeCell, sizeCell);  
 this.buttons = new JButton[this.size][this.size];  
  
 for(int i = 0; i < this.size; ++i) {  
 for(int j = 0; j < this.size; ++j) {  
 JButton button = new JButton();  
 button.setActionCommand(String.format("%d,%d", i, j));  
 button.setPreferredSize(btSize);  
 button.setBackground(new Color(250, 217, 14));  
 button.addActionListener(this);  
 button.setIcon(new ImageIcon(this.white));  
 this.buttons[i][j] = button;  
 this.gridPanel.add(button);  
 }  
 }  
  
 JPanel buttonPane = new JPanel();  
 buttonPane.setBackground(new Color(250, 217, 14));  
 this.getContentPane().add(buttonPane, "South");  
 buttonPane.setLayout(new FlowLayout(1, 5, 5));  
 JButton btExit = new JButton("FINISH");  
 btExit.setFont(new Font("Showcard Gothic", 0, 22));  
 btExit.setBackground(new Color(246, 174, 50));  
 btExit.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 Game.this.dispose();  
 }  
 });  
 buttonPane.add(btExit);  
 this.getRootPane().setDefaultButton(btExit);  
 this.pack();  
 this.setLocationRelativeTo((Component)null);  
 }  
  
 public void actionPerformed(ActionEvent e) {  
 if (!this.gameOver) {  
 if (!this.isComputer) {  
 JButton button = (JButton)e.getSource();  
 String[] temp = button.getActionCommand().split(",");  
 int i = Integer.parseInt(temp[0]);  
 int j = Integer.parseInt(temp[1]);  
 if (this.fields.isFree(i, j)) {  
 this.fields.setRow(i, j, State.X);  
 this.draw();  
 if (this.fields.isFull()) {  
 this.checkWin();  
 }  
  
 this.isComputer = true;  
 if (this.gameOver) {  
 return;  
 }  
  
 this.computeMove();  
 if (this.fields.isFull()) {  
 this.checkWin();  
 }  
 }  
  
 }  
 }  
 }  
  
 public void drawWin(List<Cell> list, State state) {  
 Image image = this.crossWin;  
 if (state == State.O) {  
 image = this.zerroWin;  
 }  
  
 Iterator var4 = list.iterator();  
  
 while(var4.hasNext()) {  
 Cell cell = (Cell)var4.next();  
 this.buttons[cell.getX()][cell.getY()].setIcon(new ImageIcon(image));  
 }  
  
 }  
  
 public void checkWin() {  
 this.gameOver = true;  
 Cell[][] cells = this.toCellFieild(this.fields);  
 List<List<Cell>> listAll = new ArrayList();  
 List<List<Cell>> tempList = this.getRows(cells);  
 listAll.addAll(tempList);  
 tempList = this.getColls(cells);  
 listAll.addAll(tempList);  
 tempList = this.getDiagMain(cells);  
 listAll.addAll(tempList);  
 tempList = this.getDiagMainTop(cells);  
 listAll.addAll(tempList);  
 tempList = this.getDiagMainBottom(cells);  
 listAll.addAll(tempList);  
 tempList = this.getDiagSide(cells);  
 listAll.addAll(tempList);  
 tempList = this.getDiagSideTop(cells);  
 listAll.addAll(tempList);  
 tempList = this.getDiagSideBottom(cells);  
 listAll.addAll(tempList);  
 List<Cell> listX = new ArrayList();  
 int max = 0;  
 Iterator var6 = listAll.iterator();  
  
 while(var6.hasNext()) {  
 List<Cell> list = (List)var6.next();  
 List<Cell> temp = new ArrayList();  
 Iterator var9 = list.iterator();  
  
 while(var9.hasNext()) {  
 Cell cell = (Cell)var9.next();  
 if (cell.getState() == State.X) {  
 temp.add(cell);  
 if (temp.size() > max) {  
 listX.clear();  
 listX.addAll(temp);  
 max = listX.size();  
 }  
 } else {  
 temp = new ArrayList();  
 }  
 }  
 }  
  
 List<Cell> listO = new ArrayList();  
 max = 0;  
 Iterator var13 = listAll.iterator();  
  
 while(var13.hasNext()) {  
 List<Cell> list = (List)var13.next();  
 List<Cell> temp = new ArrayList();  
 Iterator var18 = list.iterator();  
  
 while(var18.hasNext()) {  
 Cell cell = (Cell)var18.next();  
 if (cell.getState() == State.O) {  
 temp.add(cell);  
 if (temp.size() > max) {  
 listO.clear();  
 listO.addAll(temp);  
 max = listO.size();  
 }  
 } else {  
 temp = new ArrayList();  
 }  
 }  
 }  
  
 int x = listX.size();  
 int o = listO.size();  
 if (x > o) {  
 this.drawWin(listX, State.X);  
 this.info.setText("YOU WON!");  
 } else if (o > x) {  
 this.drawWin(listO, State.O);  
 this.info.setText("YOU LOSE!");  
 } else {  
 this.info.setText("DRAW!");  
 }  
  
 }  
  
 private Cell[][] toCellFieild(PlayingField fields) {  
 Cell[][] cells = new Cell[fields.getSize()][fields.getSize()];  
  
 for(int i = 0; i < fields.getSize(); ++i) {  
 for(int j = 0; j < fields.getSize(); ++j) {  
 cells[i][j] = new Cell(i, j, fields.getRow(i, j));  
 }  
 }  
  
 return cells;  
 }  
  
 private List<List<Cell>> getRows(Cell[][] cell) {  
 List<List<Cell>> listAll = new ArrayList();  
  
 for(int i = 0; i < cell.length; ++i) {  
 List<Cell> list = new ArrayList();  
  
 for(int j = 0; j < this.size; ++j) {  
 list.add(cell[i][j]);  
 }  
  
 listAll.add(list);  
 }  
  
 return listAll;  
 }  
  
 private List<List<Cell>> getColls(Cell[][] cell) {  
 List<List<Cell>> listAll = new ArrayList();  
  
 for(int i = 0; i < cell.length; ++i) {  
 List<Cell> list = new ArrayList();  
  
 for(int j = 0; j < this.size; ++j) {  
 list.add(cell[j][i]);  
 }  
  
 listAll.add(list);  
 }  
  
 return listAll;  
 }  
  
 private List<List<Cell>> getDiagSideTop(Cell[][] cells) {  
 List<List<Cell>> listAll = new ArrayList();  
  
 for(int k = 1; k < this.size - 1; ++k) {  
 int i = 0;  
 int j = k;  
  
 ArrayList list;  
 for(list = new ArrayList(); j >= 0; --j) {  
 list.add(cells[i][j]);  
 ++i;  
 }  
  
 listAll.add(list);  
 }  
  
 return listAll;  
 }  
  
 private List<List<Cell>> getDiagMainBottom(Cell[][] cells) {  
 List<List<Cell>> listAll = new ArrayList();  
  
 for(int k = 0; k < this.size - 2; ++k) {  
 List<Cell> list = new ArrayList();  
 int i = k + 1;  
  
 for(int j = 0; i < this.size && j < this.size; ++j) {  
 list.add(cells[i][j]);  
 ++i;  
 }  
  
 listAll.add(list);  
 }  
  
 return listAll;  
 }  
  
 private List<List<Cell>> getDiagMainTop(Cell[][] cells) {  
 List<List<Cell>> listAll = new ArrayList();  
  
 for(int k = 0; k < this.size - 2; ++k) {  
 int j = k + 1;  
 int i = 0;  
  
 ArrayList list;  
 for(list = new ArrayList(); i < this.size && j < this.size; ++j) {  
 list.add(cells[i][j]);  
 ++i;  
 }  
  
 listAll.add(list);  
 }  
  
 return listAll;  
 }  
  
 private List<List<Cell>> getDiagSideBottom(Cell[][] cells) {  
 List<List<Cell>> listAll = new ArrayList();  
  
 for(int k = 1; k < this.size - 1; ++k) {  
 int i = this.size - 1;  
 int j = k;  
  
 ArrayList list;  
 for(list = new ArrayList(); i < this.size && j < this.size; ++j) {  
 list.add(cells[i][j]);  
 --i;  
 }  
  
 listAll.add(list);  
 }  
  
 return listAll;  
 }  
  
 private List<List<Cell>> getDiagMain(Cell[][] cells) {  
 List<List<Cell>> listAll = new ArrayList();  
 List<Cell> list = new ArrayList();  
  
 for(int i = 0; i < cells.length; ++i) {  
 list.add(cells[i][i]);  
 }  
  
 listAll.add(list);  
 return listAll;  
 }  
  
 private List<List<Cell>> getDiagSide(Cell[][] cells) {  
 List<List<Cell>> listAll = new ArrayList();  
 List<Cell> list = new ArrayList();  
 int i = this.size;  
  
 for(int j = 0; j < this.size; ++j) {  
 --i;  
 list.add(cells[i][j]);  
 }  
  
 listAll.add(list);  
 return listAll;  
 }  
  
 public void computeMove() {  
 try {  
 Cell cell = this.service.nextMove(this.fields);  
 this.fields.setRow(cell.getX(), cell.getY(), State.O);  
 this.draw();  
 if (this.fields.isFull()) {  
 this.checkWin();  
 }  
  
 this.isComputer = false;  
 this.info.setText("YOUR TURN");  
 } catch (RemoteException var2) {  
 this.info.setText("service ERROR");  
 JOptionPane.showMessageDialog((Component)null, "service ERROR");  
 }  
  
 }  
  
 public void draw() {  
 for(int i = 0; i < this.size; ++i) {  
 for(int j = 0; j < this.size; ++j) {  
 if (this.fields.getRow(i, j) == State.EMPTY) {  
 this.buttons[i][j].setIcon(new ImageIcon(this.white));  
 }  
  
 if (this.fields.getRow(i, j) == State.X) {  
 this.buttons[i][j].setIcon(new ImageIcon(this.cross));  
 }  
  
 if (this.fields.getRow(i, j) == State.O) {  
 this.buttons[i][j].setIcon(new ImageIcon(this.zerro));  
 }  
 }  
 }  
  
 SwingUtilities.invokeLater(new Runnable() {  
 public void run() {  
 Game.this.gridPanel.updateUI();  
 }  
 });  
 }  
}

# Листинг Cell.java

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.game;  
  
import java.io.Serializable;  
  
public class Cell implements Serializable {  
 private static final long serialVersionUID = 1112221L;  
 private int x;  
 private int y;  
 State state;  
  
 public Cell(int x, int y) {  
 this.x = x;  
 this.y = y;  
 this.state = State.O;  
 }  
  
 public Cell(int x, int y, State state) {  
 this.x = x;  
 this.y = y;  
 this.state = state;  
 }  
  
 public int getX() {  
 return this.x;  
 }  
  
 public int getY() {  
 return this.y;  
 }  
  
 public State getState() {  
 return this.state;  
 }  
  
 public int hashCode() {  
 return super.hashCode();  
 }  
  
 public String toString() {  
 return String.format("CellP{%d,%d}", this.x, this.y);  
 }  
}

# Листинг InfoDialog.java

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.game;  
  
import java.awt.BorderLayout;  
import java.awt.FlowLayout;  
import java.awt.Font;  
import java.awt.LayoutManager;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import javax.swing.JButton;  
import javax.swing.JDialog;  
import javax.swing.JLabel;  
import javax.swing.JPanel;  
import javax.swing.border.EmptyBorder;  
  
public class InfoDialog extends JDialog {  
 private final JPanel contentPanel = new JPanel();  
  
 public InfoDialog() {  
 this.setResizable(false);  
 this.setModal(true);  
 this.setBounds(100, 100, 450, 300);  
 this.getContentPane().setLayout(new BorderLayout());  
 this.contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));  
 this.getContentPane().add(this.contentPanel, "Center");  
 this.contentPanel.setLayout((LayoutManager)null);  
 JLabel lblNewLabel\_1 = new JLabel("Справка");  
 lblNewLabel\_1.setHorizontalAlignment(0);  
 lblNewLabel\_1.setHorizontalTextPosition(0);  
 lblNewLabel\_1.setFont(new Font("Tahoma", 1, 18));  
 lblNewLabel\_1.setBounds(161, 45, 121, 16);  
 this.contentPanel.add(lblNewLabel\_1);  
 lblNewLabel\_1 = new JLabel("Инфа");  
 lblNewLabel\_1.setHorizontalAlignment(0);  
 lblNewLabel\_1.setHorizontalTextPosition(0);  
 lblNewLabel\_1.setBounds(194, 107, 56, 16);  
 this.contentPanel.add(lblNewLabel\_1);  
 JPanel buttonPane = new JPanel();  
 buttonPane.setLayout(new FlowLayout(1));  
 this.getContentPane().add(buttonPane, "South");  
 JButton okButton = new JButton("Назад");  
 okButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 InfoDialog.this.dispose();  
 }  
 });  
 okButton.setActionCommand("OK");  
 buttonPane.add(okButton);  
 this.getRootPane().setDefaultButton(okButton);  
 }  
}

# Листинг MainFrame.java

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.game;  
  
import com.game.rmi.GameService;  
import java.awt.Color;  
import java.awt.Component;  
import java.awt.Dimension;  
import java.awt.EventQueue;  
import java.awt.Font;  
import java.awt.FontFormatException;  
import java.awt.LayoutManager;  
import java.awt.SystemColor;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.IOException;  
import java.net.MalformedURLException;  
import java.rmi.Naming;  
import java.rmi.NotBoundException;  
import java.rmi.RemoteException;  
import javax.swing.DefaultComboBoxModel;  
import javax.swing.DefaultListCellRenderer;  
import javax.swing.JButton;  
import javax.swing.JComboBox;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JOptionPane;  
import javax.swing.JPanel;  
import javax.swing.border.EmptyBorder;  
  
public class MainFrame extends JFrame {  
 private JPanel contentPane;  
  
 public static void main(String[] args) {  
 EventQueue.invokeLater(new Runnable() {  
 public void run() {  
 try {  
 MainFrame frame = new MainFrame();  
 frame.setVisible(true);  
 } catch (Exception var2) {  
 var2.printStackTrace();  
 }  
  
 }  
 });  
 }  
  
 public MainFrame() throws FontFormatException, IOException {  
 this.setBackground(Color.BLACK);  
 this.setForeground(SystemColor.desktop);  
 this.setResizable(false);  
 this.setDefaultCloseOperation(3);  
 this.setBounds(100, 100, 450, 272);  
 this.contentPane = new JPanel();  
 this.contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));  
 this.setContentPane(this.contentPane);  
 this.contentPane.setLayout((LayoutManager)null);  
 new Font("Arial Black", 1, 22);  
 JLabel lblNewLabel = new JLabel("TIC TAC TOE");  
 lblNewLabel.setBounds(132, 11, 180, 51);  
 lblNewLabel.setHorizontalAlignment(0);  
 lblNewLabel.setFont(new Font("Showcard Gothic", 1, 25));  
 this.contentPane.add(lblNewLabel);  
 final JComboBox<Size> chSize = new JComboBox();  
 chSize.setPreferredSize(new Dimension(100, 25));  
 chSize.setBackground(new Color(246, 174, 50));  
 chSize.setBounds(144, 75, 155, 35);  
 chSize.setFont(new Font("Showcard Gothic", 0, 22));  
 chSize.setRenderer(new DefaultListCellRenderer() {  
 {  
 this.setHorizontalAlignment(0);  
 this.setEnabled(false);  
 }  
 });  
 this.contentPane.add(chSize);  
 chSize.setModel(new DefaultComboBoxModel(new Size[]{new Size(3, "3x3"), new Size(4, "4x4"), new Size(5, "5x5"), new Size(6, "6x6"), new Size(7, "7x7"), new Size(8, "8x8"), new Size(9, "9x9"), new Size(10, "10x10")}));  
 JButton btStart = new JButton("NEW GAME");  
 btStart.setBackground(new Color(246, 174, 50));  
 btStart.setPreferredSize(new Dimension(100, 25));  
 btStart.setFont(new Font("Showcard Gothic", 0, 18));  
 btStart.setBounds(147, 123, 150, 35);  
 this.contentPane.add(btStart);  
 this.contentPane.setBackground(new Color(250, 217, 14));  
 JButton btInfo = new JButton("i");  
 btInfo.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 try {  
 InfoDialog dialog = new InfoDialog();  
 dialog.setDefaultCloseOperation(2);  
 dialog.setVisible(true);  
 } catch (Exception var3) {  
 var3.printStackTrace();  
 }  
  
 }  
 });  
 btInfo.setFont(new Font("Tahoma", 3, 18));  
 btInfo.setBounds(335, 130, 57, 25);  
 JButton btExit = new JButton("EXIT");  
 btExit.setBackground(new Color(246, 174, 50));  
 btExit.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 System.exit(1);  
 }  
 });  
 btExit.setPreferredSize(new Dimension(100, 25));  
 btExit.setFont(new Font("Showcard Gothic", 0, 18));  
 btExit.setBounds(147, 180, 150, 35);  
 this.contentPane.add(btExit);  
 btStart.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 Size choose = (Size)chSize.getSelectedItem();  
 int size = choose.getSize();  
  
 try {  
 GameService server = (GameService)Naming.lookup("//localhost/GameService");  
 PlayingField fields = server.newGame(size);  
 Game dialog = new Game(fields);  
 dialog.setDefaultCloseOperation(2);  
 dialog.setVisible(true);  
 } catch (RemoteException | NotBoundException | MalformedURLException var7) {  
 JOptionPane.showMessageDialog((Component)null, "Ошибка подключения к серверу!");  
 }  
  
 }  
 });  
 this.setLocationRelativeTo((Component)null);  
 }  
  
 private class Size {  
 int size;  
 String info;  
  
 public Size(int size, String info) {  
 this.size = size;  
 this.info = info;  
 }  
  
 public String getInfo() {  
 return this.info;  
 }  
  
 public int getSize() {  
 return this.size;  
 }  
  
 public String toString() {  
 return this.info;  
 }  
 }  
}

# Листинг PlayingField.java

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.game;  
  
import java.io.Serializable;  
  
public class PlayingField implements Serializable {  
 private static final long serialVersionUID = 1112221L;  
 private State[][] rows;  
 private int size;  
  
 public PlayingField(int size) {  
 this.size = size;  
 this.rows = new State[size][size];  
  
 for(int i = 0; i < size; ++i) {  
 for(int j = 0; j < size; ++j) {  
 this.rows[i][j] = State.EMPTY;  
 }  
 }  
  
 }  
  
 public State[][] getRows() {  
 return this.rows;  
 }  
  
 public int getSize() {  
 return this.size;  
 }  
  
 public boolean isFull() {  
 for(int i = 0; i < this.size; ++i) {  
 for(int j = 0; j < this.size; ++j) {  
 if (this.rows[i][j] == State.EMPTY) {  
 return false;  
 }  
 }  
 }  
  
 return true;  
 }  
  
 public void setRow(int i, int j, State state) {  
 this.rows[i][j] = state;  
 }  
  
 public State getRow(int i, int j) {  
 return this.rows[i][j];  
 }  
  
 public boolean isFree(int i, int j) {  
 return this.rows[i][j] == State.EMPTY;  
 }  
  
 public int hashCode() {  
 return super.hashCode();  
 }  
}

# Листинг State.java

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.game;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public enum State {  
 X(1),  
 O(-1),  
 EMPTY(0);  
  
 public int value;  
 private static Map map = new HashMap();  
  
 private State(int value) {  
 this.value = value;  
 }  
  
 public int getValue() {  
 return this.value;  
 }  
  
 public static State valueOf(int cellState) {  
 return (State)map.get(cellState);  
 }  
  
 static {  
 State[] var0 = values();  
 int var1 = var0.length;  
  
 for(int var2 = 0; var2 < var1; ++var2) {  
 State cellState = var0[var2];  
 map.put(cellState.value, cellState);  
 }  
  
 }  
}

# Сервер

# Листинг Cell.java

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.game;  
  
import java.io.Serializable;  
  
public class Cell implements Serializable {  
 private static final long serialVersionUID = 1112221L;  
 private int x;  
 private int y;  
 State state;  
  
 public Cell(int x, int y) {  
 this.x = x;  
 this.y = y;  
 this.state = State.O;  
 }  
  
 public Cell(int x, int y, State state) {  
 this.x = x;  
 this.y = y;  
 this.state = state;  
 }  
  
 public int getX() {  
 return this.x;  
 }  
  
 public int getY() {  
 return this.y;  
 }  
  
 public State getState() {  
 return this.state;  
 }  
  
 public int hashCode() {  
 return super.hashCode();  
 }  
  
 public String toString() {  
 return String.format("CellP{%d,%d}", this.x, this.y);  
 }  
}

# Листинг PlayingField.java

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.game;  
  
import java.io.Serializable;  
  
public class PlayingField implements Serializable {  
 private static final long serialVersionUID = 1112221L;  
 private State[][] rows;  
 private int size;  
  
 public PlayingField(int size) {  
 this.size = size;  
 this.rows = new State[size][size];  
  
 for(int i = 0; i < size; ++i) {  
 for(int j = 0; j < size; ++j) {  
 this.rows[i][j] = State.EMPTY;  
 }  
 }  
  
 }  
  
 public State[][] getRows() {  
 return this.rows;  
 }  
  
 public int getSize() {  
 return this.size;  
 }  
  
 public boolean isFull() {  
 for(int i = 0; i < this.size; ++i) {  
 for(int j = 0; j < this.size; ++j) {  
 if (this.rows[i][j] == State.EMPTY) {  
 return false;  
 }  
 }  
 }  
  
 return true;  
 }  
  
 public void setRow(int i, int j, State state) {  
 this.rows[i][j] = state;  
 }  
  
 public State getRow(int i, int j) {  
 return this.rows[i][j];  
 }  
  
 public boolean isFree(int i, int j) {  
 return this.rows[i][j] == State.EMPTY;  
 }  
  
 public int hashCode() {  
 return super.hashCode();  
 }  
}

# Листинг State.java

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.game;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public enum State {  
 X(1),  
 O(-1),  
 EMPTY(0);  
  
 public int value;  
 private static Map map = new HashMap();  
  
 private State(int value) {  
 this.value = value;  
 }  
  
 public int getValue() {  
 return this.value;  
 }  
  
 public static State valueOf(int cellState) {  
 return (State)map.get(cellState);  
 }  
  
 static {  
 State[] var0 = values();  
 int var1 = var0.length;  
  
 for(int var2 = 0; var2 < var1; ++var2) {  
 State cellState = var0[var2];  
 map.put(cellState.value, cellState);  
 }  
  
 }  
}