Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Санкт-Петербургский государственный электротехнический университет
“ЛЭТИ” им.В.И.Ульянова (Ленина) »

Кафедра МОЭВМ

ОТЧЕТ
по лабораторно-практической работе № 3, 4, 5 «Разработка интерфейса пользователя» по дисциплине «Объектно - ориентированное программирование на языке Java»

Выполнил Загуменнов И.М.

Факультет КТИ

Группа № 3311

Подпись преподавателя _____

Санкт-Петербург

2024 г

Цель работы

Знакомство с правилами построения экранной формы.

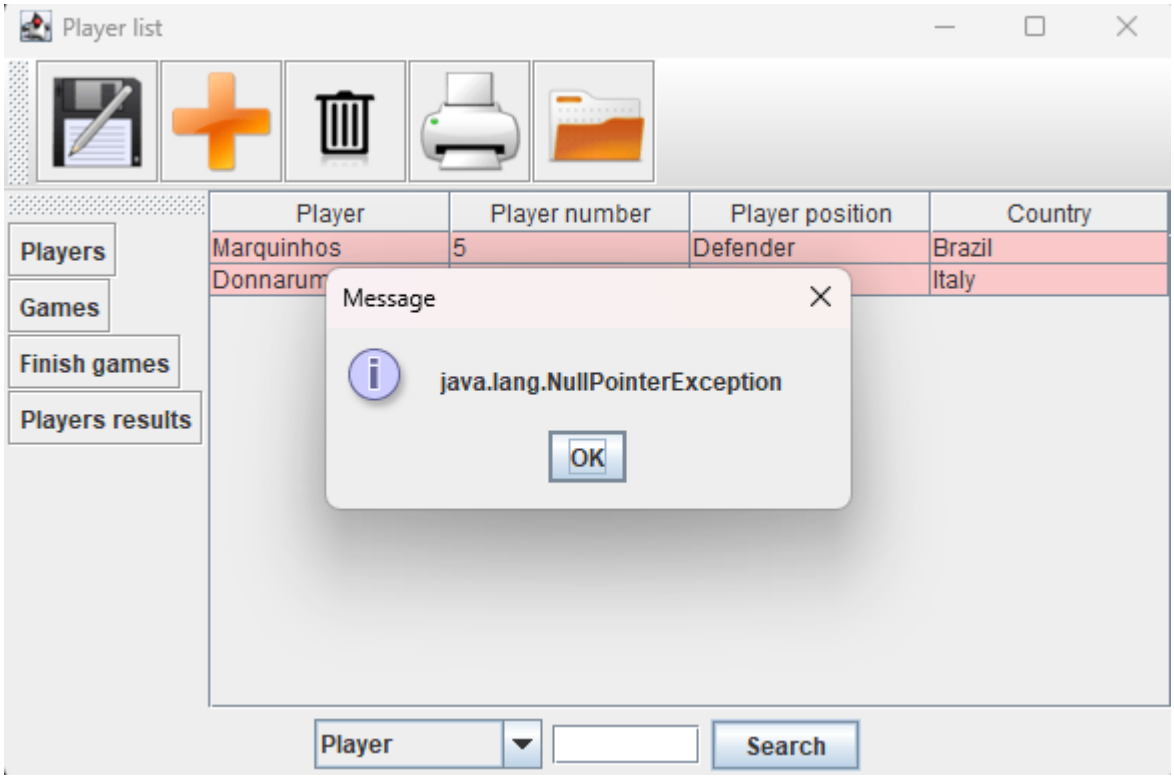
Описание задания

Добавить ActionListener в программу, для работы поиска и для переключения между таблицами. Также для некоторых моментов добавить выбросы исключений(exception). Добавить кнопкам save и open функцию открытия и сохранения в файл.

Описание проверки работоспособности приложения

Полную работоспособность приложения можно увидеть на примере 1 и 2.

Пример 1:



Пример 2:

```
buttonP.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        playerList.add(scroll, BorderLayout.CENTER);
        playerList.setVisible(true);
    }
});
```

Приимер 3:

```
save.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e){
        FileDialog save = new FileDialog(playerList, title: "Сохранение данных", FileDialog.SAVE);
        save.setFile("players.txt");
        save.setVisible(true); // Отобразить запрос пользователю
        // Определить имя выбранного каталога и файла
        String fileName = save.getDirectory() + save.getFile();
        if(fileName == null) return; // Если пользователь нажал «отмена»
        try {
            BufferedWriter writer = new BufferedWriter (new FileWriter(fileName));
            for (int i = 0; i < model.getRowCount(); i++)     // Для всех строк
                for (int j = 0; j < model.getColumnCount(); j++)  // Для всех столбцов
                {writer.write ((String) model.getValueAt(i, j)); // Записать значение из ячейки
                    writer.write( str: "\n"); // Записать символ перевода каретки
                }
            writer.close();
        }
        catch(IOException ec)    // Ошибка записи в файл
        { ec.printStackTrace();  }


    }
```

Ссылка на репозиторий

https://github.com/hazlmar/OOP1

В этом репозитории находятся исходные файлы лабораторных, данная работа хранится в папке Lab-02:

playerlist.java – основной код

playerlist.html – документация, сгенерированная JavaDoc

Также есть видеоотчет в репозитории 2024-10-20 23-03-09.mkv или по ссылке https://disk.yandex.ru/i/qk-edxehezNJ_Q

Текст программы

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
public class playerlist {
```

```
    // Объявления графических компонентов
    private JFrame playerList;
    private DefaultTableModel model, modelGames, modelEnd, modelRes;
    private JButton save, print, add, delete, open;
    private JButton buttonP, buttonG, buttonEg, buttonRes;
    private JToolBar toolBar, panel;
    private JScrollPane scroll, scrollGame, scrollEnd, scrollRes;
    private JTable players, games, endG, gameRes;
    private JComboBox player;
    private JTextField playerName;
    private JButton filter;
    public void show() {
        // Создание окна
        playerList = new JFrame("Player list");
        playerList.setSize(600, 400);
        playerList.setLocation(1200, 200);
        playerList.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Создание кнопок и прикрепление иконок
        save = new JButton(new ImageIcon("save.png"));
        add = new JButton(new ImageIcon("add.png"));
        delete = new JButton(new ImageIcon("delete.png"));
        print = new JButton(new ImageIcon("print.png"));
        open = new JButton(new ImageIcon("open.png"));

        // Настройка подсказок для кнопок
        save.setToolTipText("Save");
        add.setToolTipText("Add player");
        delete.setToolTipText("Delete player");
        print.setToolTipText("Print");
        open.setToolTipText("Open");

        save.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e){
                FileDialog save = new FileDialog(playerList, "Сохранение данных", FileDialog.SAVE);
                save.setFile("players.txt");
                save.setVisible(true); // Отобразить запрос пользователю
                // Определить имя выбранного каталога и файла
                String fileName = save.getDirectory() + save.getFile();
                if(fileName == null) return; // Если пользователь нажал «отмена»
                try {
                    BufferedWriter writer = new BufferedWriter (new FileWriter(fileName));
                    for (int i = 0; i < model.getRowCount(); i++)     // Для всех строк
                        for (int j = 0; j < model.getColumnCount(); j++)  // Для всех столбцов
                        {writer.write ((String) model.getValueAt(i, j)); // Записать значение из ячейки
                            writer.write("\n"); // Записать символ перевода каретки
                        }
                    writer.close();
                }
                catch(IOException ec)    // Ошибка записи в файл
                { ec.printStackTrace();  }


            }
        });open.addActionListener(new ActionListener() {
```

```
        @Override
        public void actionPerformed(ActionEvent e){
            FileDialog open = new FileDialog(playerList, "Открыть файл", FileDialog.LOAD);
            open.setFile("players.txt");
            open.setVisible(true); // Отобразить запрос пользователю
            // Определить имя выбранного каталога и файла
            String fileName = open.getDirectory() + open.getFile();
            if(fileName == null) return; // Если пользователь нажал «отмена»
            try {
                BufferedReader reader = new BufferedReader(new FileReader(fileName));
                int rows = model.getRowCount();
                for (int i = 0; i < rows; i++) model.removeRow(0); // Очистка таблицы
                String author;
                do {
                    author = reader.readLine();
                    if(author != null)
                    { String title = reader.readLine();
                        String have = reader.readLine();
                        String country = reader.readLine();
                        model.addRow(new String[]{author, title, have, country}); // Запись строки в таблицу
                    }
                } while(author != null);
                reader.close();
            } catch (FileNotFoundException ex) {ex.printStackTrace();} // файл не найден
            catch (IOException ec) {ec.printStackTrace();}

        }
    });
    // Добавление кнопок на панель инструментов
    toolBar = new JToolBar("Toolbar");
    toolBar.add(save);
    toolBar.add(add);
    toolBar.add(delete);
    toolBar.add(print);
    toolBar.add(open);

    // Размещение панели инструментов
    playerList.setLayout(new BorderLayout());
    playerList.add(toolBar, BorderLayout.NORTH);

    // Создание таблицы с данными
    String [] columns = {"Player", "Player number", "Player position", "Country"};
    String [][] data = {{"Marquinhos", "5", "Defender", "Brazil"},
            {"Donnarumma", "1", "Goalkeeper", "Italy"}};
    model=  new DefaultTableModel(data, columns);
    players = new JTable(model);
    scroll = new JScrollPane(players);

    // Размещение таблицы с данными
    playerList.add(scroll, BorderLayout.CENTER);

    // Визуализация экранной формы
    players.setBackground(new Color(150, 200, 100));
    playerList.setVisible(true);

    //Подготовка таблицы
```

```
        String[] columnsGame = {"Date", "Command opponent"};
        String[][] dataGame = {{"09.10.2023", "CSKA"},
            {"09.11.2023" , "LOKO"}};
        modelGames =  new DefaultTableModel(dataGame, columnsGame);
        games = new JTable(modelGames);
        scrollGame = new JScrollPane(games);

        String[] columnsGameEnd = {"Date", "Scores", "Command opponent"};
        String[][] dataGameEnd = {{"09.10.2023", "5:0" , "CSKA"},
            {"09.11.2023", "3:0" , "LOKO"}};
        modelEnd =  new DefaultTableModel(dataGameEnd, columnsGameEnd);
        endG = new JTable(modelEnd);
        scrollEnd = new JScrollPane(endG);

        String[] columnsRes = {"Date", "Player", "Scores"};
        String[][] dataRes = {{"09.10.2023", "Wilmar Barrios" , "3"},
            {"09.11.2023", "Andrey Lunev" , "0"}};
        modelRes =  new DefaultTableModel(dataRes, columnsRes);
        gameRes = new JTable(modelRes);
        scrollRes = new JScrollPane(gameRes);

        playerList.add(scroll, BorderLayout.CENTER);

        //Создание кнопок выбора таблицы
        buttonP = new JButton("Players");
        buttonP.setToolTipText("Show players table");

        buttonG = new JButton("Games");
        buttonG.setToolTipText("Show games calendar");

        buttonEg = new JButton("Finish games");
        buttonEg.setToolTipText("Show finish games");

        buttonRes = new JButton("Players results");
        buttonRes.setToolTipText("Show players results");
        buttonP.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                playerList.add(scroll, BorderLayout.CENTER);
                playerList.setVisible(true);
            }
        });

        buttonG.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                playerList.add(scrollGame, BorderLayout.CENTER);
                playerList.setVisible(true);
            }
        });

        buttonEg.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                playerList.add(scrollEnd, BorderLayout.CENTER);
                playerList.setVisible(true);
```

```
        }
    });

    buttonRes.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            playerList.add(scrollRes, BorderLayout.CENTER);
            playerList.setVisible(true);
        }
    });

    panel = new JToolBar("Tabels");
    panel.setOrientation(SwingConstants.VERTICAL);
    panel.add(buttonP);
    panel.add(buttonG);
    panel.add(buttonEg);
    panel.add(buttonRes);
    playerList.add(panel, BorderLayout.WEST);


    player = new JComboBox(new String[]{"Player", "Wilmar Barrios", "Andrey Lunev"});
    playerName = new JTextField("Player name");
    filter = new JButton("Search");

    //Добавление компонентов на панель
    JPanel filterPane = new JPanel();
    filterPane.add(player);
    filterPane.add(playerName);
    filterPane.add(filter);

    //Размещение панели поиска внизу окна.
    playerList.add(filterPane, BorderLayout.SOUTH);

    //Визуализация экранной формы
    players.setBackground(new Color(250,200,200));
    playerList.setVisible(true);


    filter.addActionListener (new ActionListener()
    {
        public void actionPerformed (ActionEvent event)
        {
            try{ checkName(playerName);
            }
            catch(NullPointerException ex){
                JOptionPane.showMessageDialog(playerList, ex.toString());
            }
            catch(MyException myEx){
                JOptionPane.showMessageDialog(null, myEx.getMessage());
            }
        }});
}
private class MyException extends Exception{
    public MyException(){
        super("You didn't enter the name of player");
    }
```

```
    }
    private void checkName(JTextField pName) throws MyException, NullPointerException{
        String sName = pName.getText();
        if(sName.contains("Player name")) throw new MyException();
        if (sName.length() == 0) throw new NullPointerException();
    }

    public static void main(String[] args) {
        // Создание и отображение экранной формы
        new playerlist().show();
    }
}
```