

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 № 5
«Разработка интерфейса пользователя» по
дисциплине «Объектно - ориентированное
программирование на языке Java»

Выполнил Пасечный Л.В.

Факультет КТИ

Группа № 3311

Подпись преподавателя _____

Санкт-Петербург
2024 г

Цель работы

Знакомство с организацией обмена данными между объектами экранной формы и файлом.

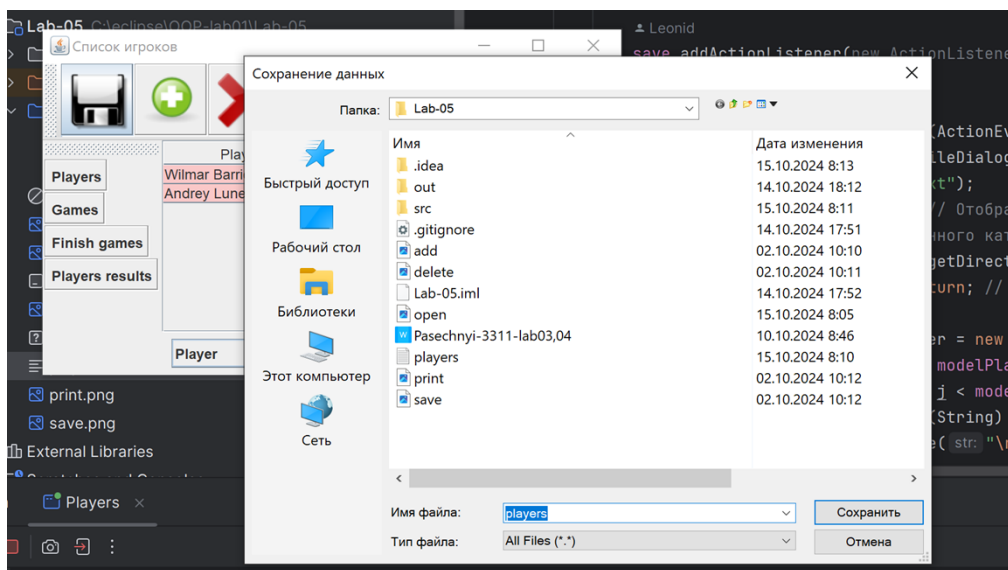
Описание задания

Добавить кнопкам save и open функцию открытия и сохранения в файл.

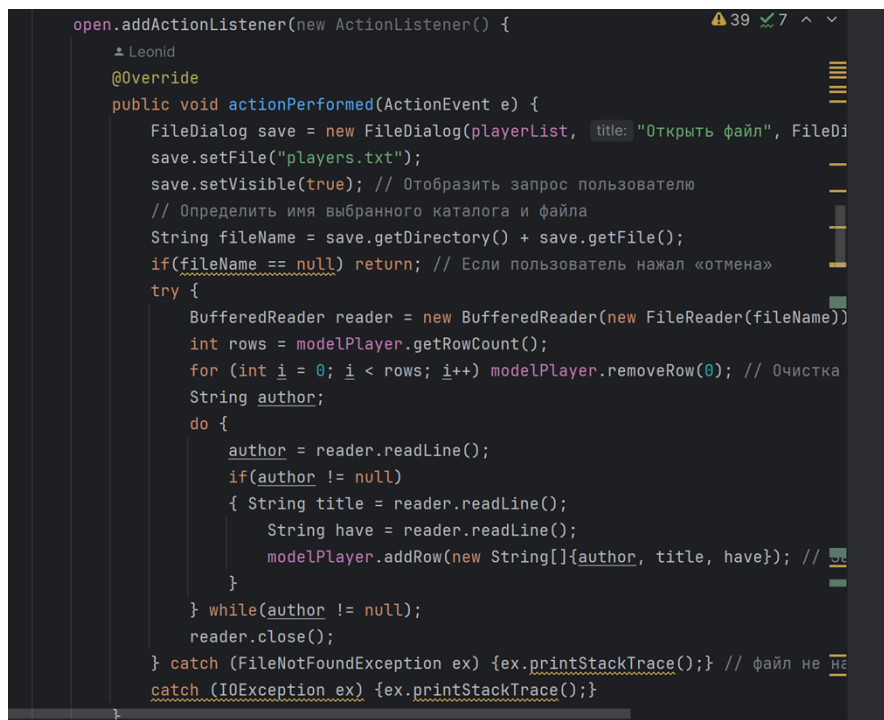
Описание проверки работоспособности приложения

Полную работоспособность приложения можно увидеть на примере 1 и 2.

Пример 1:



Пример 3:



Для функции save код аналогичен, но отличается в

FileDialog.SAVE.

Ссылка на репозиторий

<https://github.com/Mazer1234/OOP-lab01.git>

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

Players.java – основной код

Players.html – документация, сгенерированная JavaDoc

Также есть видеоотчет 2024-10-10 08-13-43.mp4 в репозитории или по ссылке <https://disk.yandex.ru/i/mJ7-4O3yf-nX8g>

Текст программы

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class Players {
    private JFrame playerList;
    private DefaultTableModel modelPlayer, modelGames, modelEndGames, modelResult;
    private JButton save, print, add, delete, open;
    private JButton buttonPlayer, buttonGames, buttonEndGames, buttonResult;
    private JToolBar toolBar, choosePanel;
    private JScrollPane scroll, scrollGames, scrollEndGames, scrollResult;
    private JTable players, games, endGames, result;
    private JComboBox player;
    private JTextField playerName;
    private JButton filter;
    public void show() {
        //Создание окна
        playerList = new JFrame("Список игроков");
        playerList.setSize(500, 300);
        playerList.setLocation(100, 100);
        playerList.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

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

        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("players.txt"));
        for (int i = 0; i < modelPlayer.getRowCount(); i++) // Для всех строк
            for (int j = 0; j < modelPlayer.getColumnCount(); j++) // Для всех столбцов
                {writer.write ((String) modelPlayer.getValueAt(i, j)); // Записать значение из ячейки
                writer.write("\n"); // Записать символ перевода каретки
            }
        writer.close();
    }
    catch(IOException ex) // Ошибка записи в файл
    { ex.printStackTrace(); }
}
});

```

```

open.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        FileDialog save = new FileDialog(playerList, "Открыть файл", FileDialog.LOAD);
        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 {
            BufferedReader reader = new BufferedReader(new FileReader(fileName));
            int rows = modelPlayer.getRowCount();
            for (int i = 0; i < rows; i++) modelPlayer.removeRow(0); // Очистка таблицы
            String author;
            do {
                author = reader.readLine();
                if(author != null)
                { String title = reader.readLine();
                  String have = reader.readLine();
                  modelPlayer.addRow(new String[] {author, title, have}); // Запись строки в таблицу
                }
            } while(author != null);
            reader.close();
        } catch (FileNotFoundException ex) {ex.printStackTrace();} // файл не найден
        catch (IOException ex) {ex.printStackTrace();}
    }
});

```

```

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

```

```

//Добавление кнопок на панель инструментов
toolBar = new JToolBar("Tools");
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 игроков с данными
```

```
String[] columns = { "Player", "Number", "Position"};
String[][] data = {{ "Wilmar Barrios", "35", "midfielder"},
    { "Andrey Lunev", "99", "goalkeeper"} };
modelPlayer = new DefaultTableModel(data, columns);
players = new JTable(modelPlayer);
scroll = new JScrollPane(players);
```

```
//Создание календаря игр.
```

```
String[] ColumnsGame = {"Date", "Command opponent"};
String[][] DataGames = {{ "09.10.2023", "CSKA"},
    { "09.11.2023", "LOKO"} };
```

```
modelGames = new DefaultTableModel(DataGames, ColumnsGame);
games = new JTable(modelGames);
scrollGames = new JScrollPane(games);
```

```
String[] ColumnsEndGames = {"Date", "Scores", "Command opponent"};
String[][] DataEndGames = {{ "09.10.2023", "5:0", "CSKA"},
    { "09.11.2023", "3:0", "LOKO"} };
modelEndGames = new DefaultTableModel(DataEndGames, ColumnsEndGames);
endGames = new JTable(modelEndGames);
scrollEndGames = new JScrollPane(endGames);
```

```
String[] ColumnsResult = {"Date", "Player", "Scores"};
String[][] DataResult = {{ "09.10.2023", "Wilmar Barrios", "3"},
    { "09.11.2023", "Andrey Lunev", "0"} };
modelResult = new DefaultTableModel(DataResult, ColumnsResult);
result = new JTable(modelResult);
scrollResult = new JScrollPane(result);
```

```
//Размещение таблицы с данными.
```

```
playerList.add(scroll, BorderLayout.CENTER);
```

```
//Создание кнопок выбора таблицы
```

```
buttonPlayer = new JButton("Players");
buttonPlayer.setToolTipText("Show players table");
```

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

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

```
buttonResult = new JButton("Players results");
buttonResult.setToolTipText("Show players results");
```

```
//Обработка кнопок включения таблиц
```

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

```

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

```

```

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

```

```

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

```

```

//Добавление компонентов выбора таблицы
choosePanel = new JToolBar();
choosePanel.setOrientation(SwingConstants.VERTICAL);
choosePanel.add(buttonPlayer);
choosePanel.add(buttonGames);
choosePanel.add(buttonEndGames);
choosePanel.add(buttonResult);

```

```

playerList.add(choosePanel, BorderLayout.WEST);
//Подготовка компонентов поиска
player = new JComboBox(new String[] {"Player", "Wilmar Barrios", "Andrey Lunev"});
playerName = new JTextField("Player name");
filter = new JButton("Search");

```

```

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

```

```

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

```

```

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

```

```

//Добавление действия
filter.addActionListener(new ActionListener()
{
    public void actionPerformed (ActionEvent event)
    {
        try { checkName(playerName);
        }import javax.swing.*;

```

```

import javax.swing.table.DefaultTableModel;
import java.awt.*;

```

```

import java.awt.event.*;
import java.io.*;

public class Players{
    private JFrame playerList;
    private DefaultTableModel modelPlayer, modelGames, modelEndGames, modelResult;
    private JButton save, print, add, delete, open;
    private JButton buttonPlayer, buttonGames, buttonEndGames, buttonResult;
    private JToolBar toolBar, choosePanel;
    private JScrollPane scroll, scrollGames, scrollEndGames, scrollResult;
    private JTable players, games, endGames, result;
    private JComboBox player;
    private JTextField playerName;
    private JButton filter;
    public void show(){
        //Создание окна
        playerList = new JFrame("Список игроков");
        playerList.setSize(500, 300);
        playerList.setLocation(100, 100);
        playerList.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

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

        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("players.txt"));
                    for (int i = 0; i < modelPlayer.getRowCount(); i++) // Для всех строк
                        for (int j = 0; j < modelPlayer.getColumnCount(); j++) // Для всех столбцов
                            {writer.write ((String) modelPlayer.getValueAt(i, j)); // Записать значение из ячейки
                                writer.write("\n"); // Записать символ перевода каретки
                            }
                    writer.close();
                }
                catch(IOException ex) // Ошибка записи в файл
                { ex.printStackTrace(); }

            }
        });

        open.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                FileDialog save = new FileDialog(playerList, "Открыть файл", FileDialog.LOAD);
                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 {
    BufferedReader reader = new BufferedReader(new FileReader(fileName));
    int rows = modelPlayer.getRowCount();
    for (int i = 0; i < rows; i++) modelPlayer.removeRow(0); // Очистка таблицы
    String author;
    do {
        author = reader.readLine();
        if(author != null)
        { String title = reader.readLine();
          String have = reader.readLine();
          modelPlayer.addRow(new String[]{author, title, have}); // Запись строки в таблицу
        }
    } while(author != null);
    reader.close();
} catch (FileNotFoundException ex) {ex.printStackTrace();} // файл не найден
catch (IOException ex) {ex.printStackTrace();}
}
});

```

```

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

```

```

//Добавление кнопок на панель инструментов
toolBar = new JToolBar("Tools");
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 игроков с данными

```

```

String[] columns = { "Player", "Number", "Position" };
String[][] data = {{ "Wilmar Barrios", "35", "midfielder"},
    { "Andrey Lunev", "99", "goalkeeper" }};
modelPlayer = new DefaultTableModel(data, columns);
players = new JTable(modelPlayer);
scroll = new JScrollPane(players);

```

```

//Создание календаря игр.
String[] ColumnsGame = {"Date", "Command opponent"};
String[][] DataGames = {{ "09.10.2023", "CSKA"},

```



```

        {"09.11.2023" , "LOKO"}});

modelGames = new DefaultTableModel(DataGames, ColumnsGame);
games = new JTable(modelGames);
scrollGames = new JScrollPane(games);

String[] ColumnsEndGames = {"Date", "Scores", "Command opponent"};
String[][] DataEndGames = {{ "09.10.2023", "5:0" , "CSKA"},
        {"09.11.2023", "3:0" , "LOKO"}};
modelEndGames = new DefaultTableModel(DataEndGames, ColumnsEndGames);
endGames = new JTable(modelEndGames);
scrollEndGames = new JScrollPane(endGames);

String[] ColumnsResult = {"Date", "Player", "Scores"};
String[][] DataResult = {{ "09.10.2023", "Wilmar Barrios" , "3"},
        {"09.11.2023", "Andrey Lunev" , "0"}};
modelResult = new DefaultTableModel(DataResult, ColumnsResult);
result = new JTable(modelResult);
scrollResult = new JScrollPane(result);

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

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

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

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

buttonResult = new JButton("Players results");
buttonResult.setToolTipText("Show players results");

//Обработка кнопок включения таблиц
buttonPlayer.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        playerList.add(scroll, BorderLayout.CENTER);
        playerList.setVisible(true);
    }
});

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

buttonEndGames.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        playerList.add(scrollEndGames, BorderLayout.CENTER);

```

```

        playerList.setVisible(true);
    }
});

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

//Добавление компонентов выбора таблицы
choosePanel = new JToolBar();
choosePanel.setOrientation(SwingConstants.VERTICAL);
choosePanel.add(buttonPlayer);
choosePanel.add(buttonGames);
choosePanel.add(buttonEndGames);
choosePanel.add(buttonResult);

playerList.add(choosePanel, BorderLayout.WEST);
//Подготовка компонентов поиска
player = new JComboBox(new String[]{"Player", "Wilmar Barrios", "Andrey Lunev"});
playerName = new JTextField("Player name");
filter = new JButton("Search");

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

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

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

//Добавление действия
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 Players().show();
};
}
```