**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе № 4**

**по дисциплине «Технология объектно-ориентированного программирования и проектирования»**

Тема: Обработка исключений

|  |  |
| --- | --- |
| Студент гр. 4312 | Устинов В. А. |
| Преподаватель | Лазарев Э. А. |

Санкт-Петербург

2025

**Цель работы**

Знакомство с механизмом обработки исключений в языке Java.

**Перечень ситуаций, контролируемых с помощью исключений**

1. Проверка пустых полей

* Пользователь оставляет поле ввода пустым
* Вид исключения: InputException.ErrorType.*EMPTY\_FIELD*
* Метод: InputException.validEmptyField(JTextField[] textFields, String[] fieldNames)
* Пустое поле ввода => "Field “FieldName” cannot be empty"

2. Проверка номеров строк

* Ввод некорректного номера строки
* Виды исключений: InputException.ErrorType.(*EMPTY\_FIELD*, *INVALID\_NUMBER*, *OUT\_OF\_RANGE*)
* Метод: InputException.validRowNumber(String rowNumberStr, int maxRows)
* Пустой ввод => "Row number cannot be empty"
* Нечисловой ввод => "Row number must be a number"
* Вне диапазона => "Invalid row number: X. Valid range: 1-Y"

3. Проверка формата данных

* Неправильный формат массива данных
* Вид исключения: InputException.ErrorType.*INVALID\_FORMAT*
* Метод: InputException.validDataArray(String[] data, int expectedLength)
* Null массив => "Data cannot be null"
* Недостаточное количество полей => "Invalid data format. Expected X fields, got Y"

4. Проверка текстовых полей (только буквы)

* Ввод небуквенных символов в текстовые поля
* Вид исключения: InputException.ErrorType.*INVALID\_FORMAT*
* Метод: InputException.validLettersOnly(String text, String fieldName)
* Ввод цифр или спецсимволов => " “fieldName” should contain only letters"

5. Проверка числовых полей (0 или 1)

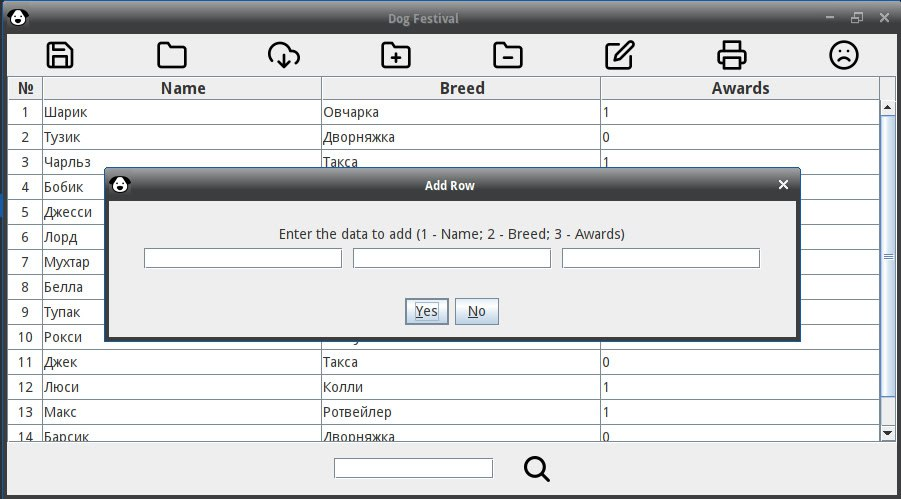
* Ввод значений отличных от 0 или 1 в особое числовое поле
* Вид исключения: InputException.ErrorType.*INVALID\_FORMAT*
* Метод: InputException.validZeroOrOne(String text, String fieldName)
* Некорректный ввод => " “fieldName” should be only 0 or 1"

6. Обработка ошибок инициализации данных

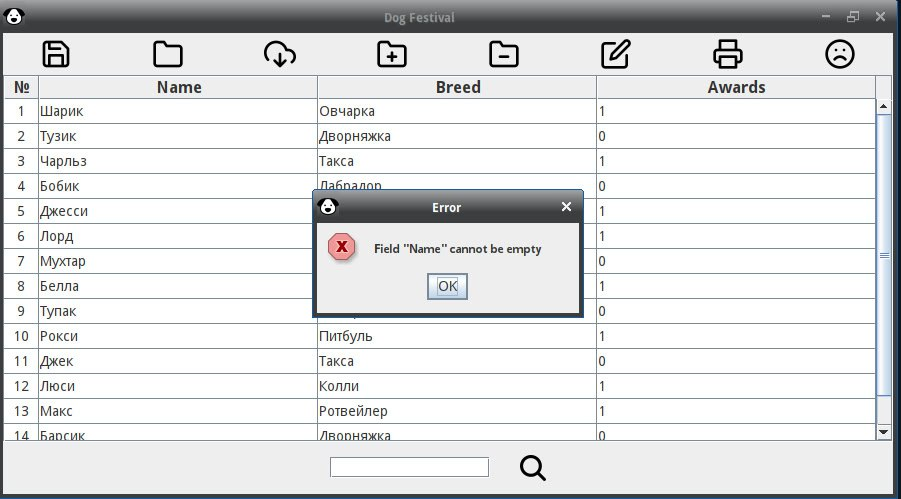
* Ошибки при загрузке данных из CSV файла
* Вид исключения: InputException.ErrorType.*INVALID\_FORMAT*
* Метод: MainWindow.initData()
* Ошибка инициализации => "Error initializing data: [детали ошибки]"

**Скриншоты, иллюстрирующие работу обработчиков ситуаций**

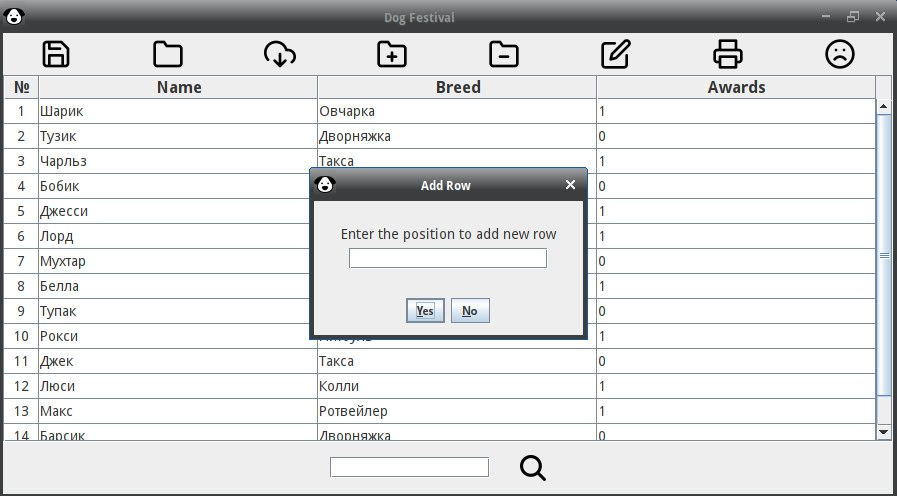
1) Обработка пустых полей



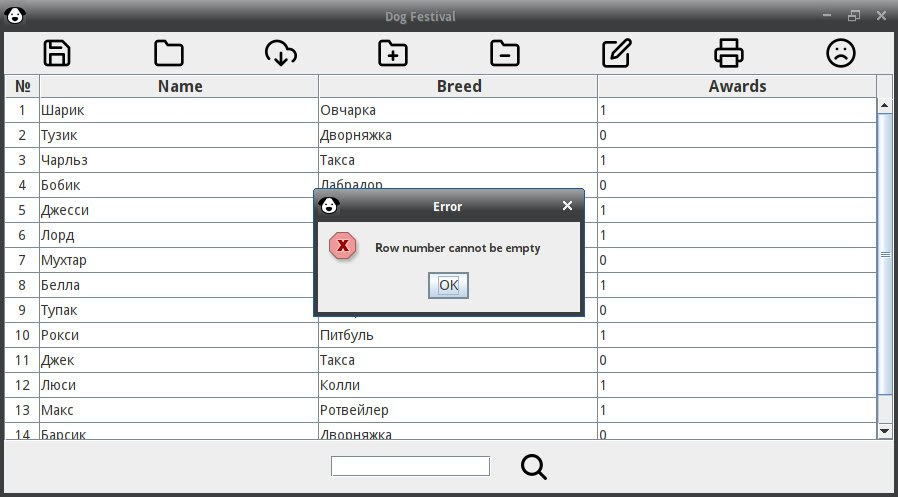
*Рис. 1 — Ввод пустой строки в поле ввода*



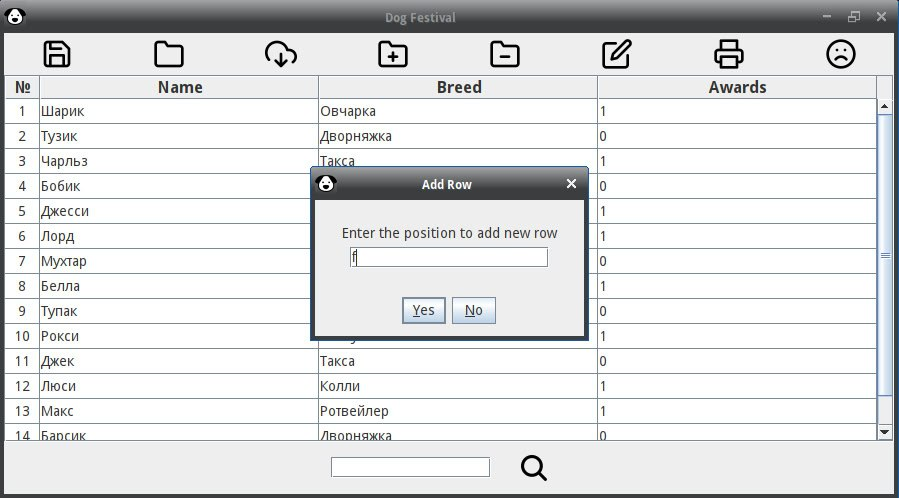
*Рис. 2 — Всплывающее окно соответствующей ошибки*

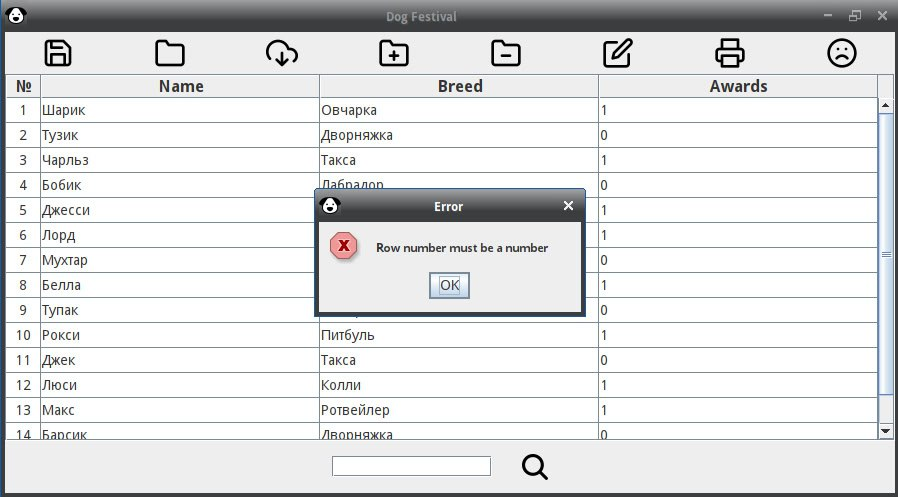
2) Обработка номеров строк

*Рис. 3 — Ввод пустой строки в поле ввода номера строки*

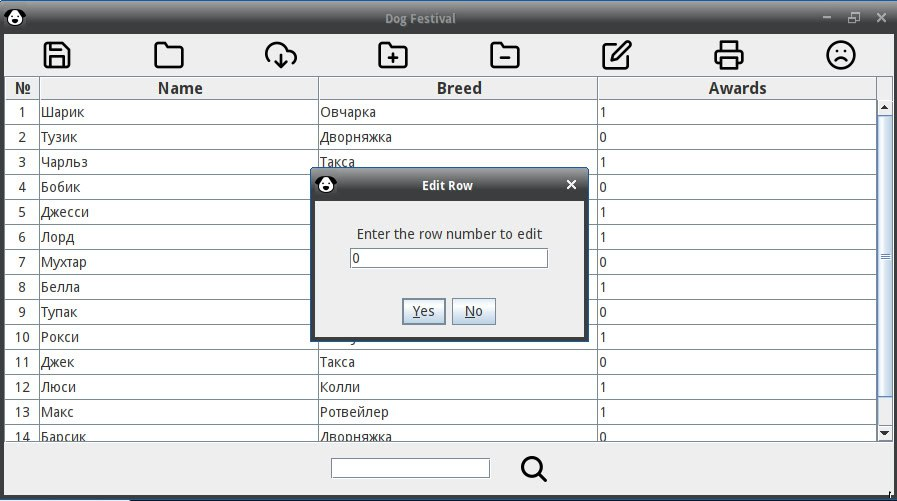


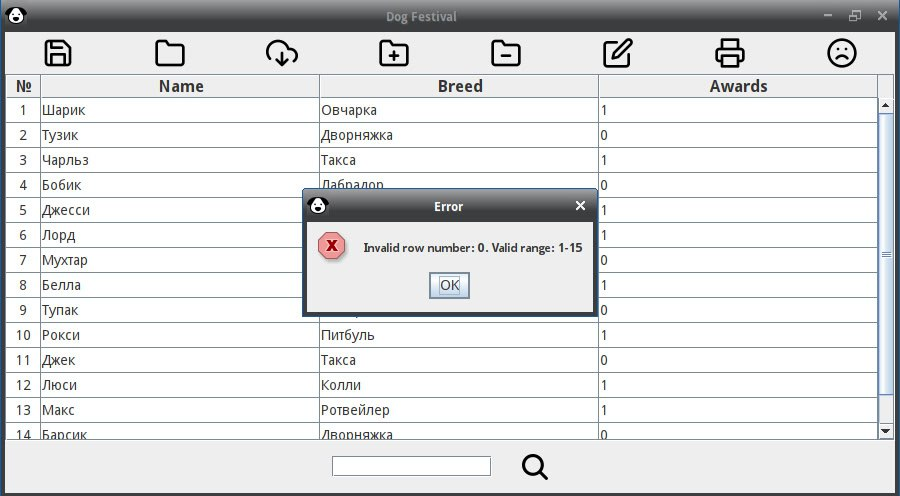
*Рис. 4 — Всплывающее окно соответствующей ошибки*



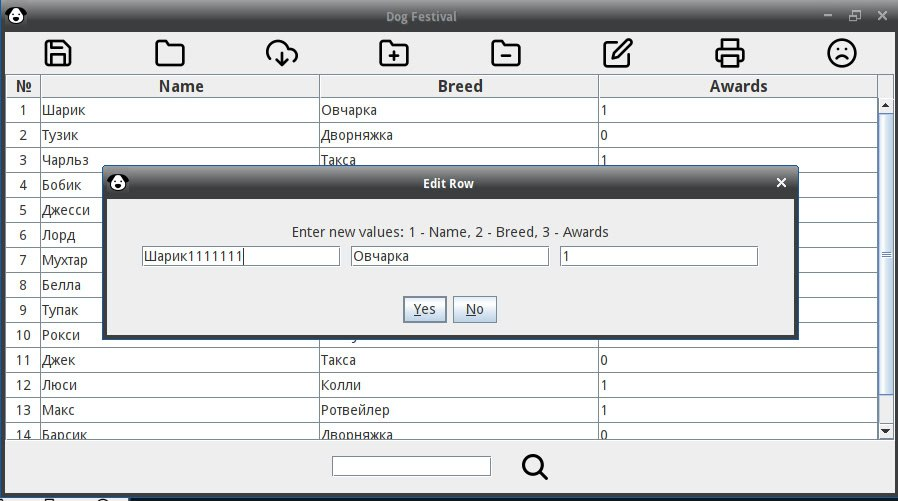
*Рис. 5 — Ввод буквы или спецсимвола в поле ввода номера строки*

*Рис. 6 — Всплывающее окно соответствующей ошибки*

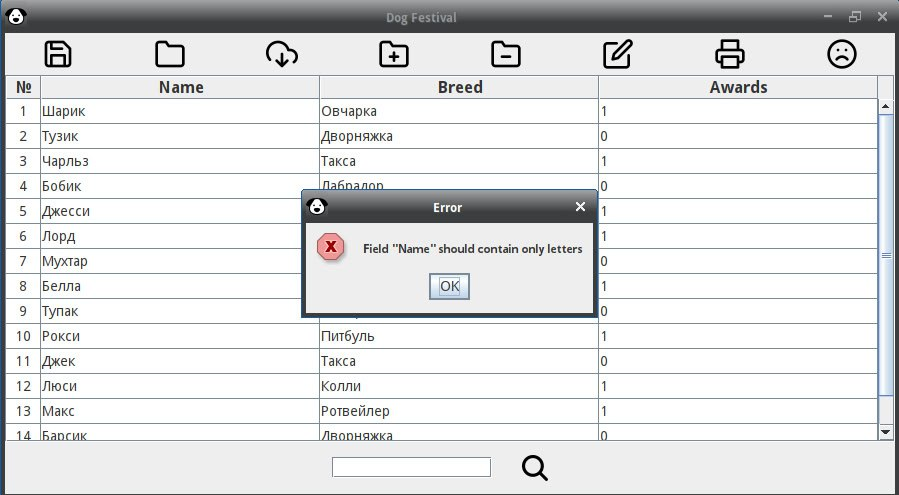


*Рис. 7 — Ввод номера строки за пределами таблицы*

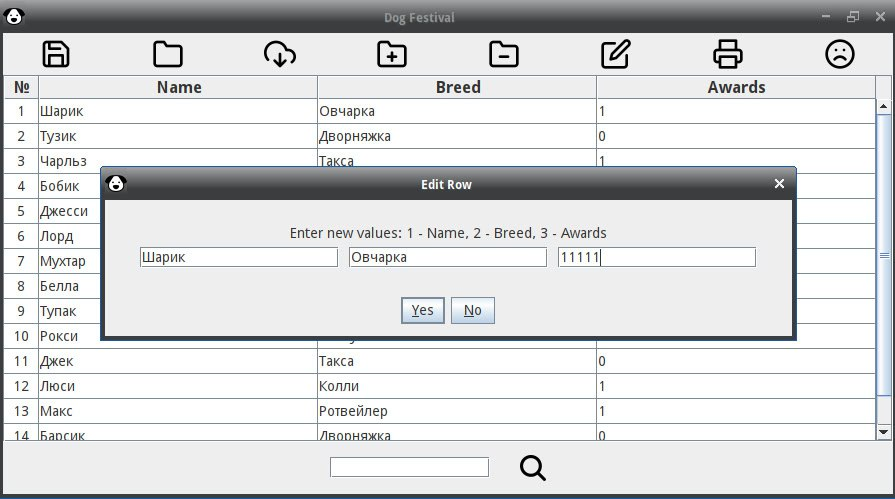
*Рис. 8 — Всплывающее окно соответствующей ошибки*

3) Обработка текстовых полей

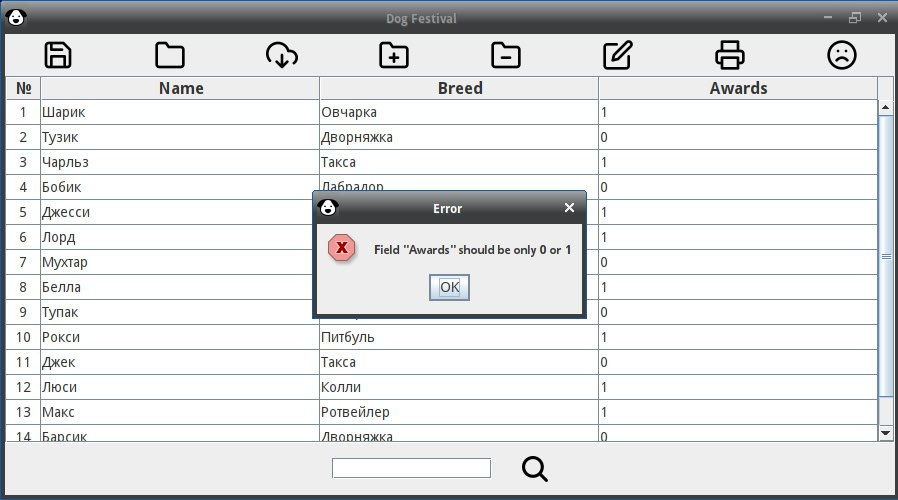
*Рис. 9 — Ввод цифр или спецсимволов в текстовые поля ввода*



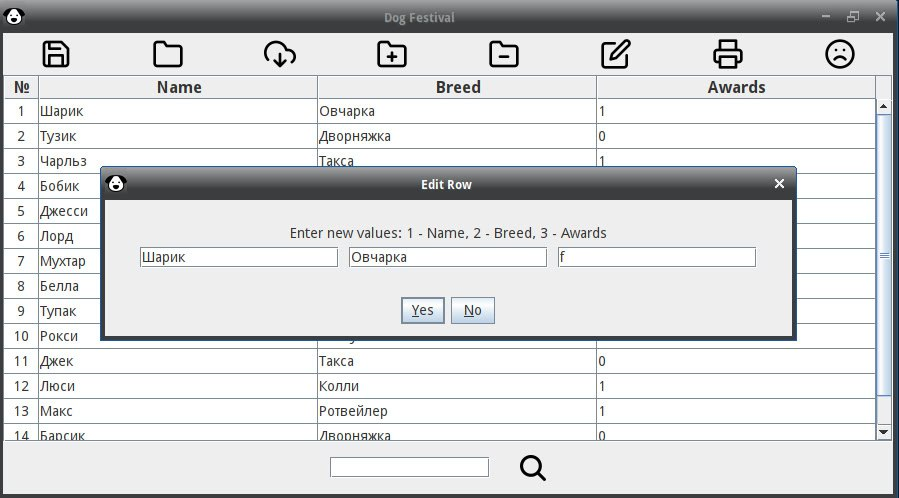
*Рис. 10 — Всплывающее окно соответствующей ошибки*

4) Обработка особых числовых полей

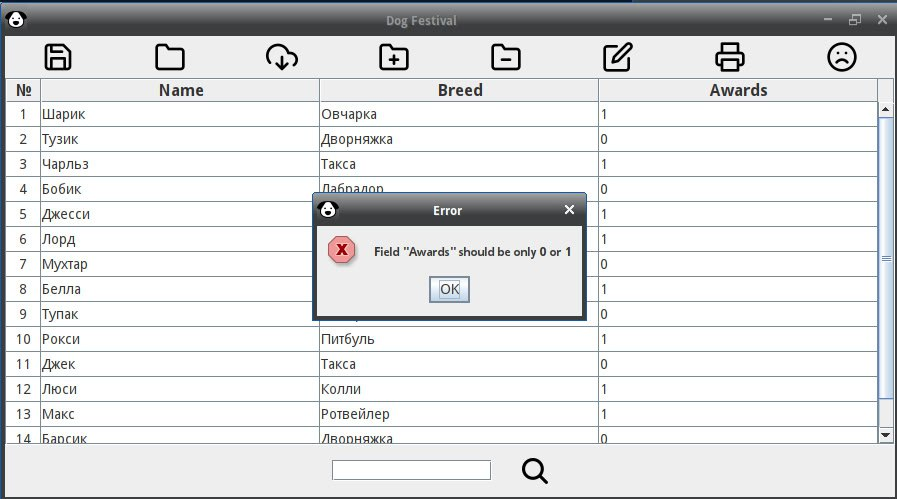
*Рис. 11 — Ввод не 1 или 0*



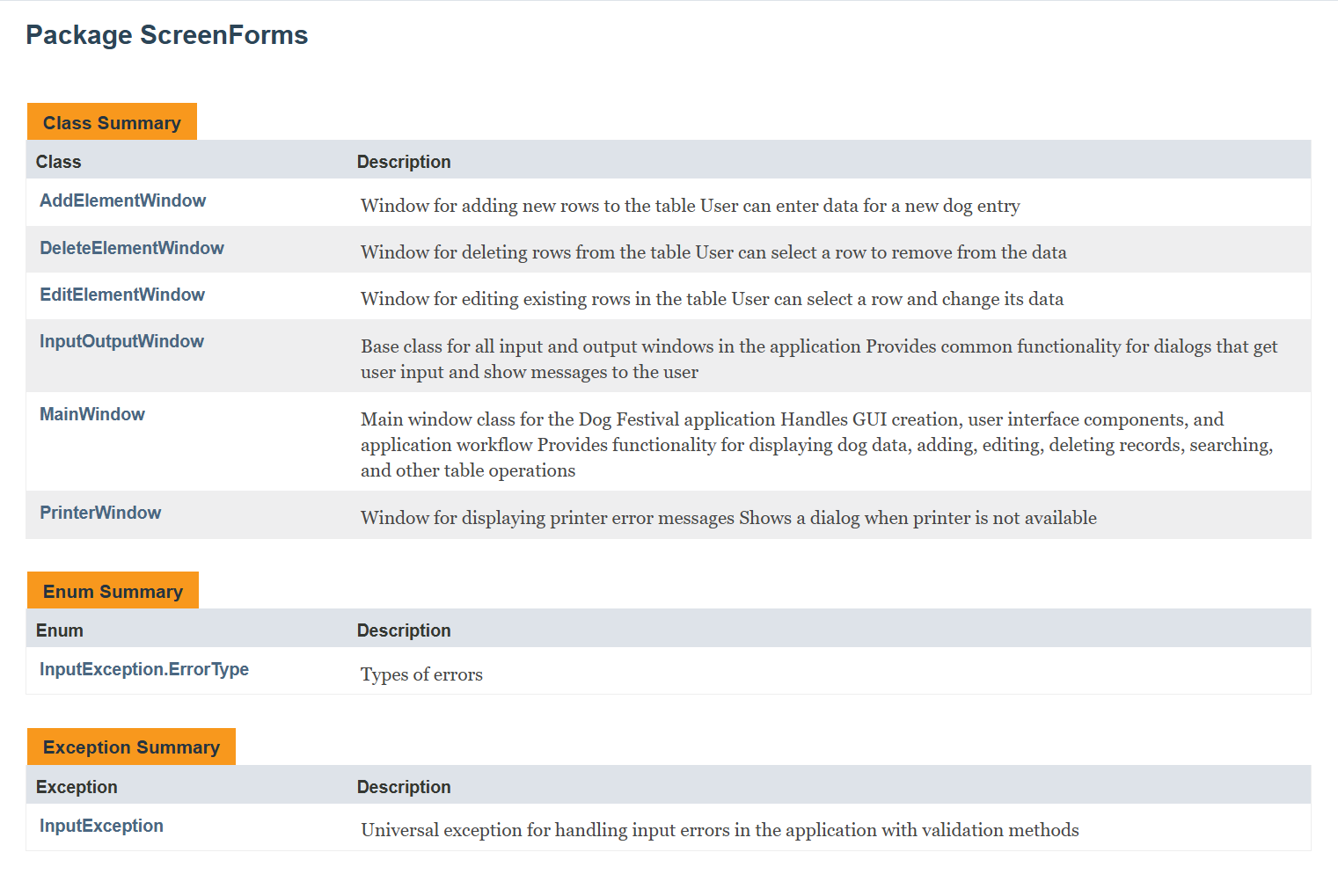
*Рис. 12 — Всплывающее окно соответствующей ошибки*

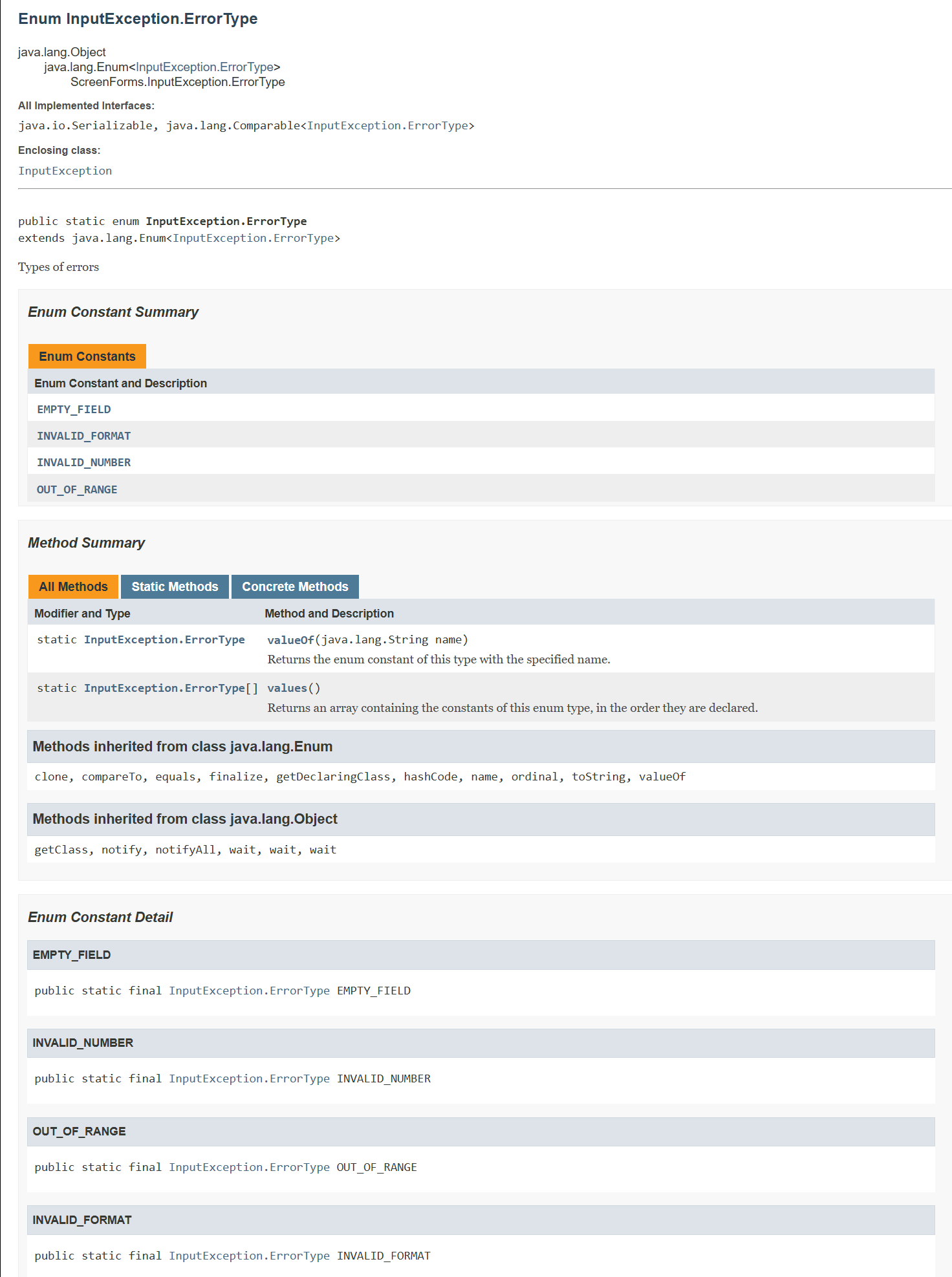


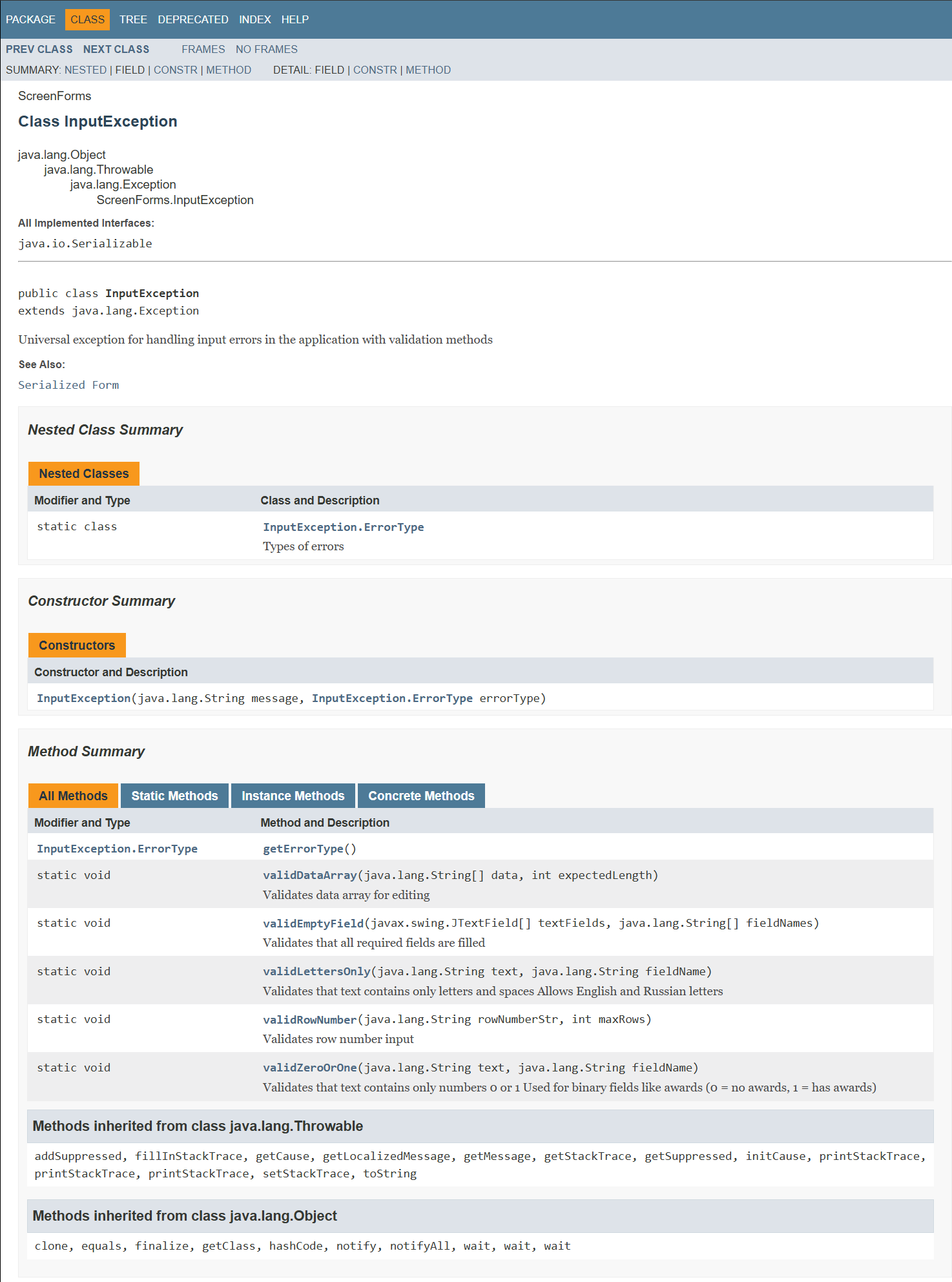
*Рис. 13 — Ввод буквы в числовое поле*



*Рис. 14 — Всплывающее окно соответствующей ошибки*

**Документация javadoc**





**Исходные тексты классов собственных исключений, методов, где генерируются и обрабатываются исключительные ситуации**

1) Класс InputException

package ScreenForms;  
  
import javax.swing.JTextField;  
  
public class InputException extends Exception {  
    private final ErrorType errorType;  
  
    public enum ErrorType {  
        EMPTY\_FIELD,  
        INVALID\_NUMBER,  
        OUT\_OF\_RANGE,  
        INVALID\_FORMAT  
    }  
  
    public InputException(String message, ErrorType errorType)  
    {  
        super(message);  
        this.errorType = errorType;  
    }  
  
    public ErrorType getErrorType()  
    {  
        return errorType;  
    }  
  
    public static void validEmptyField(JTextField[] textFields, String[] fieldNames) throws InputException  
    {  
        for (int i = 0; i < textFields.length; i++)  
        {  
            if (textFields[i].getText().trim().isEmpty())  
            {  
                String fieldName = (fieldNames != null && i < fieldNames.length) ? fieldNames[i] : "Field " + (i + 1);  
                throw new InputException("Field ''" + fieldName + "'' cannot be empty", ErrorType.EMPTY\_FIELD);  
            }  
        }  
    }  
  
    public static void validRowNumber(String rowNumberStr, int maxRows) throws InputException  
    {  
        if (rowNumberStr.trim().isEmpty())  
        {  
            throw new InputException("Row number cannot be empty", ErrorType.EMPTY\_FIELD);  
        }  
        try {  
            int rowNumber = Integer.parseInt(rowNumberStr);  
            if (rowNumber < 1 || rowNumber > maxRows)  
            {  
                throw new InputException("Invalid row number: " + rowNumber + ". Valid range: 1-" + maxRows, ErrorType.OUT\_OF\_RANGE);  
            }  
        } catch (NumberFormatException e) {  
            throw new InputException("Row number must be a number", ErrorType.INVALID\_NUMBER);  
        }  
    }  
  
    public static void validDataArray(String[] data, int expectedLength) throws InputException  
    {  
        if (data == null)  
        {  
            throw new InputException("Data cannot be null",ErrorType.INVALID\_FORMAT);  
        }  
        if (data.length < expectedLength)  
        {  
            throw new InputException("Invalid data format. Expected " + expectedLength + " fields, got " + data.length, ErrorType.INVALID\_FORMAT);  
        }  
    }  
  
    public static void validLettersOnly(String text, String fieldName) throws InputException  
    {  
        if (!text.matches("[a-zA-Zа-яА-Я\\s]+"))  
        {  
            throw new InputException("Field ''" + fieldName + "'' should contain only letters", ErrorType.INVALID\_FORMAT);  
        }  
    }

    public static void validZeroOrOne(String text, String fieldName) throws InputException  
    {  
        if (!text.matches("[01]"))  
        {  
            throw new InputException("Field ''" + fieldName + "'' should be only 0 or 1", ErrorType.INVALID\_FORMAT);  
        }  
    }  
}

2) AddElementWindow.java

private void showDataAdding() throws InputException  
{  
    boolean validInput = false;  
    operationCancelled = false;  
    while (!validInput && !operationCancelled)  
    {  
        InputException.validRowNumber(String.valueOf(rowToAdd), tableModel.getRowCount() + 1);  
  
        AddElementWindow dataInputWindow = new AddElementWindow(addTable, 3);  
  
        dataInputWindow.IODialog.setVisible(true);  
        String[] addData = dataInputWindow.getData();  
  
        if (addData != null)  
        {  
            InputException.validEmptyField(dataInputWindow.textFields);  
            InputException.validDataArray(addData, 3);  
  
            InputException.validLettersOnly(addData[0], "Name");  
            InputException.validLettersOnly(addData[1], "Breed");  
            InputException.validZeroOrOne(addData[2], "Awards");  
  
            dataInputWindow.addRowToTable(rowToAdd, addData);  
  
            if (dataInputWindow.SCSDialog != null)  
            {  
                dataInputWindow.SCSDialog.setVisible(true);  
            }  
  
            validInput = true;  
        }  
        else  
        {  
            operationCancelled = true;  
        }  
    }  
}

3) EditElementWindow.java

private void showRowSelection() throws InputException  
{  
    boolean validInput = false;  
    operationCancelled = false;  
    while (!validInput && !operationCancelled)  
    {  
        IODialog.setVisible(true);  
        String[] rowData = getData();  
  
        if (rowData != null)  
        {  
            InputException.validRowNumber(rowData[0],                     tableModel.getRowCount());  
            rowToEdit = Integer.parseInt(rowData[0]);  
  
            currentData = getCurrentRowData(rowToEdit);  
            InputException.validDataArray(currentData, 3);  
  
            validInput = true;  
        }  
        else  
        {  
            operationCancelled = true;  
        }  
    }  
}

private void showDataEditing() throws InputException  
{  
    boolean validInput = false;  
    operationCancelled = false;  
    while (!validInput && !operationCancelled)  
    {  
        InputException.validRowNumber(String.valueOf(rowToEdit), tableModel.getRowCount());  
        InputException.validDataArray(currentData, 3);  
  
        EditElementWindow editDataWindow = new EditElementWindow(editTable, 3, currentData);  
        editDataWindow.IODialog.setVisible(true);  
  
        String[] editData = editDataWindow.getData();  
  
        if (editData != null)  
        {  
            InputException.validEmptyField(editDataWindow.textFields);  
            InputException.validDataArray(editData, 3);  
  
            InputException.validLettersOnly(editData[0], "Name");  
            InputException.validLettersOnly(editData[1], "Breed");  
            InputException.validZeroOrOne(editData[2], "Awards");  
  
            editDataWindow.EditRowByNumber(rowToEdit, editData);  
  
            if (editDataWindow.SCSDialog != null)  
            {  
                editDataWindow.SCSDialog.setVisible(true);  
            }  
  
            validInput = true;  
        }  
        else  
        {  
            operationCancelled = true;  
        }  
    }  
}

4) DeleteElementWindow.java

private void showDeleteRow() throws InputException  
{  
    boolean validInput = false;  
    boolean operationCancelled = false;  
    while (!validInput && !operationCancelled)  
    {  
        IODialog.setVisible(true);  
        String[] rowData = getData();  
  
        if (rowData != null)  
        {  
            InputException.validRowNumber(rowData[0], tableModel.getRowCount());  
            rowToDelete = Integer.parseInt(rowData[0]);  
  
            deleteRowByNumber(rowToDelete);  
            validInput = true;  
        }  
        else  
        {  
            operationCancelled = true;  
        }  
    }  
}

5) MainWindow.java

private void initData() throws InputException  
{  
    try {  
        fileMngr = new FileManager();*//init FileManager object*  
        dogs = new List<>();*//init List for dogs data*  
        dogs = fileMngr.inputFromCSV("src/data/dogs3.csv"); *//writes data from dogs3.csv*  
    } catch (Exception e) {  
        throw new InputException("Error initializing data: " + e.getMessage(),  
                               InputException.ErrorType.INVALID\_FORMAT);  
    }  
}

private static void handleButtonClick(int buttonIndex)  
{  
    if (buttonIndex < tooltips.length)  
    {  
        try {  
            switch(buttonIndex)  
            {  
                case 3:  
                    AddElementWindow addElem = new AddElementWindow(dogsTable);  
                    addElem.show();  
InputException  
                    break;  
                case 4:  
                    DeleteElementWindow deleteWindow = new DeleteElementWindow(dogsTable);  
                    deleteWindow.show();  
                    break;  
                case 5:  
                    EditElementWindow editWindow = new EditElementWindow(dogsTable);  
                    editWindow.show();  
                    break;  
                case 6:  
                    PrinterWindow printWindow = new PrinterWindow();  
                    printWindow.show();  
                    break;  
                case 7:  
                    exitApplication();  
                    break;  
                case 8:  
                    String text = searchTextField.getText();  
                    searchElement(text);  
                    break;  
            }  
        } catch (InputException e) {  
            JOptionPane.showMessageDialog(  
                mainFrame,  
                e.getMessage(),  
                "Error",  
                JOptionPane.ERROR\_MESSAGE  
            );  
        }  
    }  
}

6) InputOutputWindow.java

public void showErrorDialog(String errorMessage) {  
    JOptionPane.showMessageDialog(  
        IODialog,  
        errorMessage,  
        "Error",  
        JOptionPane.ERROR\_MESSAGE  
    );  
}