Лабораторная работа № 5

Цель:

Познакомиться с платформой JavaFX и научиться использовать регулярные выражения - шаблоны для поиска строки определенного формата в тексте.

Постановка задачи:

Задание 1. Из выпадающего списка можно выбрать тип данных: натуральное число, целое число, число с плавающей запятой (не забывать про запись 1e-5), дата, время, e-mail. Рядом со списком есть текстовое поле с проверяемыми данными. “Включать” зеленый кружок, если текстовое поле соответствует выбранному типу данных, и красный кружок в противном случае.

Задание 2. В многострочном текстовом поле находится текст. В список поместить все даты из этого поля.

Решение задачи:

Задание 1.

При создании проекта на платформе JavaFX по умолчанию создается класс “Main”, который наследуется от “Application”, и метод “start”, который необходимо переопределить. В данном методе и будем реализовывать основную часть программы. Для создания выпадающего списка используем конструктор “ComboBox(ObservableList<T>items)”, параметром которого служит список, в котором хранятся типы данных. Сами данные будем вводить через объект класса “TextField”.

```
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception{
        final List<String> list = new ArrayList<~>();
        list.add("natural");
        list.add("integer");
        list.add("fractional");
        list.add("date");
        list.add("time");
        list.add("e-mail");
        ObservableList<String> items = FXCollections.observableArrayList(list);
        final ComboBox<String> comboBox = new ComboBox<~>(items);
        comboBox.setValue(items.get(0));
        final TextField field = new TextField();
        Button ok = new Button( text: "ok");
```

Рис.1. Инициализация ключевых элементов интерфейса.

Чтобы после проверки введенных данных “включался” кружок зеленого или красного цвета, создаем объекты “Label”, в которые с помощью метода “setGraphic” устанавливаем изображения. Однако до момента нажатия пользователем кнопки “ok” оба кружка не будут видны (“setVisible(false)”).

```
File fileGreen = new File( pathname: "C:\\Users\\Лиза\\IdeaProjects\\Practice5.1\\src\\plain\\green.png");
File fileRed = new File( pathname: "C:\\Users\\Лиза\\IdeaProjects\\Practice5.1\\src\\plain\\red.png");
final String localUrl2 = fileGreen.toURI().toURL().toString();
final String localUrl3 = fileRed.toURI().toURL().toString();
Image green = new Image(localUrl2);
Image red = new Image(localUrl3);
final Label labelGreen = new Label();
labelGreen.setGraphic(new ImageView(green));
final Label labelRed = new Label();
labelRed.setGraphic(new ImageView(red));
labelGreen.setVisible(false);
labelRed.setVisible(false);
```

Рис.2. Работа с объектами классов “Image” и “Label”.

Реализуем булеву функцию “getChecked(String text, String item)”, в которую будем передавать текст, полученный из “field”, и выбранный элемент выпадающего списка. Необходимо знать, какой тип данных мы получаем, поэтому в функции присутствует конструкция “try-catch”. Следующим шагом будет проверка на элемент “comboBox”.

Если строка представляет собой набор букв, цифр и других символов, то рассматриваем элементы “date”, “time”, “e-mail”, если же строка может быть преобразована в число, то нас интересуют такие типы, как “natural” и “integer”. “Fractional” может относиться и к первому, и ко второму варианту, поэтому его прописываем в обоих блоках.

Для “date” используем класс “DateValidator”, метод “isValid” которого возвращает значение “true”, если введенная строка - дата, и “false” в другом случае.

Тип “time” проверяем с помощью регулярного выражения, в котором задаем 24-часовой формат. Часы и минуты разделяет двоеточие.

Число, введенное в “natural” должно быть целым и положительным, а в “integer” только целым. Данную проверку осуществляем с помощью условий оператора “if”.

```
private boolean getChecked (String text, String item){
    try {
        if(item.equals("date")){
            DateValidator dateValidator = DateValidator.getInstance();
            return dateValidator.isValid(text, datePattern: "dd/MM/yyyy", strict: false);
        }
        if(item.equals("time")){
            Pattern comma = Pattern.compile("^(([0-1][0-9])|(2*[0-3])):[0-5][0-9]$");
            Matcher matcher1 = comma.matcher(text);
            if(matcher1.find()){ return true; }
        }
        else{
            double number = Double.parseDouble(text);
            if (item.equals("natural") && (number > 0) && (number % 1 == 0)) {
                return true;
            }
            if (item.equals("integer") && (number % 1 == 0)) { return true; }
            if (item.equals("fractional")) {
                Pattern eps = Pattern.compile("^[-+]?[0-9]*[.][0-9]+(?:[eE][-+]?[0-9]+)?$");
                Matcher matcher = eps.matcher(text);
                Pattern eps2 = Pattern.compile("^[-+]?([1-9]*([eE][-][0-9]+))$");
                Matcher matcher2 = eps2.matcher(text);
                Pattern point = Pattern.compile("^(0|[-+]?[1-9]\\d*)([.]\\d+)$");
                Matcher matcher1 = point.matcher(text);
                return matcher1.find() || matcher.find() || matcher2.find();
            }
        }
    }
```

Рис.3.Первая часть функции “getChecked”, которая выполняет проверку.

Регулярное выражение для “e-mail” тестирует строку на совпадение со стандартным форматом адреса электронной почты “имя_пользователя@имя_домена”.

```
    catch (NumberFormatException e){
        if(item.equals("e-mail")){
            Pattern mail = Pattern.compile("^[_A-Za-z0-9-+]+(\\.[_A-Za-z0-9-]+)" +
                    "*@[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$");
            Matcher m = mail.matcher(text);
            if(m.find()){ return true; }
        }
        if (item.equals("fractional")) {
            Pattern eps = Pattern.compile("^[-+]?[0-9]*[,][0-9]+(?:[eE][-+]?[0-9]+)?$");
            Matcher matcher = eps.matcher(text);
            Pattern point = Pattern.compile("^(0|[-+]?[1-9]\\d*)([,]\\d+)$");
            Matcher matcher1 = point.matcher(text);
            return matcher1.find() || matcher.find();
        }
        else return false;
    }
    return false;
}
```

Рис.4. Вторая часть функции “getChecked”.

Так как дробное число может быть представлено различными способами, прописываем несколько регулярных выражений. Таким образом, числа вида “13,4”, “0.6003”, “1e-10”, “-8,43487E+5” пройдут проверку.

Обрабатываем нажатие кнопки “ok”. Считываем информацию, введенную пользователем в текстовое поле. Затем объявляем переменную “flag”, которая принимает значение “true/false” в зависимости от результата, который возвращает функция “getChecked”: “true” - видимым становится зеленый кружок, “false” - красный.

```
ok.setOnAction((event) → {
        String text1 = field.getText();
        System.out.println(text1);
        boolean flag = getChecked(text1, comboBox.getValue());
        if(flag){
            labelRed.setVisible(false);
            labelGreen.setVisible(true);
        }
        else {
            labelGreen.setVisible(false);
            labelRed.setVisible(true);
        }
        field.clear();
});
```

Рис.5.Обработка действий с кнопкой “ok”.

“FlowPane” располагает элементы интерфейса друг за другом в порядке их следования в конструкторе. Если элементы не помещаются в одной строке, то переносятся на следующую.

```
    FlowPane rt = new FlowPane( hgap: 10,  vgap: 30, comboBox, field, ok, labelGreen, labelRed);
    rt.setAlignment(Pos.CENTER);
    Scene scene = new Scene(rt,  width: 400,  height: 300);
    primaryStage.setTitle("Practice 5.1");
    primaryStage.setScene(scene);
    primaryStage.show();
}
public static void main(String[] args) { launch(args); }
```

Рис.6.Ключевые методы создания интерфейса и функция “main”, запускающая приложение.

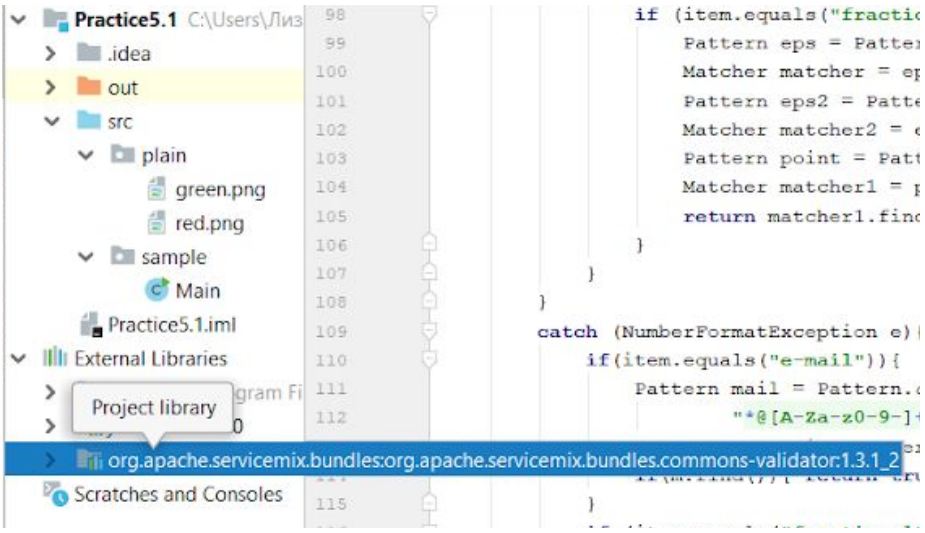

Рис.7.Специальная библиотека для работы с классом “DateValidator”.

Задание 2.

Во втором задании структура проекта аналогичная.

Создаем многострочное текстовое поле и список, куда будем помещать найденные в тексте даты.

```
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
            final List<String> list = new ArrayList<>();
            String TEXT = "Volcanic eruptions:\nVesuvius 24-08-79 \nMon Pele 08.05.1902 \n" +
                    "Ruiz 13.11?1985 \nKrakatau 26-08-1883 \nEtna 11/03/1669 \nMayon 23-10-1766 " +
                    "\nWrongExample 31.02.1900 \nCorrectExample 28.02.1900";
            TextArea textArea = new TextArea(TEXT);
            textArea.setPrefColumnCount(15);
            textArea.setPrefRowCount(15);
            ObservableList<String> observable = FXCollections.observableArrayList(list);
            ListView<String> listView = new ListView<>(observable);
```

Рис.8.Создание элементов интерфейса.

Затем реализуем функцию “isChecked(String str)”, в которую будет передаваться строка, проверяемая на принадлежность шаблону “date”. Сам шаблон описывается регулярным выражением, в котором учитываем количество месяцев в году и дней в каждом месяце. День, месяц и год могут разделяться точкой, дефисом или слешем.

```
private boolean isChecked(String str){
    Pattern date = Pattern.compile("^(?:(?:31([/\\-.])(?:0?[13578]|1[02]))\\1|" +
            "(?:(?:29|30)([/\\-.])(?:0?[1,3-9]|1[0-2])\\2))" +
            "(?:(?:1[6-9]|[2-9]\\d)?\\d{2})$|^(?:29([/\\-.])0?2\\3(?:(?:(?:1[6-9]|[2-9]\\d)?" +
            "(?:0[48]|[2468][048]|[13579][26])|(?:(?:16|[2468][048]|[3579][26])00))))$|^" +
            "(?:0?[1-9]|1\\d|2[0-8])(\\/|-|\\.)(?:(?:0?[1-9])|(?:1[0-2]))\\4" +
            "(?:(?:1[6-9]|[2-9]\\d)?\\d{2})$");
    Matcher matcher = date.matcher(str);
    return matcher.find();
}
```

Рис.9.Реализация функции “isChecked”.

Обрабатываем нажатие кнопки “ok”. Сперва очищаем список. Затем текст, полученный из поля “textArea”, разбиваем с помощью метода “split” на слова. Каждую подстроку (элемент массива “subStr”) передаем в функцию “isChecked”, возвращаемое значение которой определяет, попадет подстрока в список или нет.

```
Button ok = new Button( text: "ok");
ok.setOnAction(event -> {
    list.clear();
    observable.clear();
    String text = textArea.getText();
    String[] subStr;
    String del = " ";
    subStr = text.split(del);
    for (String s : subStr) {
        if (isChecked(s)) {
            list.add(s);
        }
    }
    observable.addAll(list);
    listView.setItems(observable);
});
```

Рис.10.Обработка нажатия кнопки “ok”, вызов функции “isChecked”.

```
        FlowPane root = new FlowPane( hgap: 10,  vgap: 10, textArea, listView, ok);
        root.setAlignment(Pos.CENTER);
        primaryStage.setTitle("Practice 5.2");
        Scene scene = new Scene(root,  width: 800,  height: 605);
        primaryStage.setScene(scene);
        primaryStage.show();
}
public static void main(String[] args) { launch(args); }
```

Рис.11.Настройки интерфейса приложения и функция “main”.

Результат:

Задание 1.

Тестируем программу на корректность.

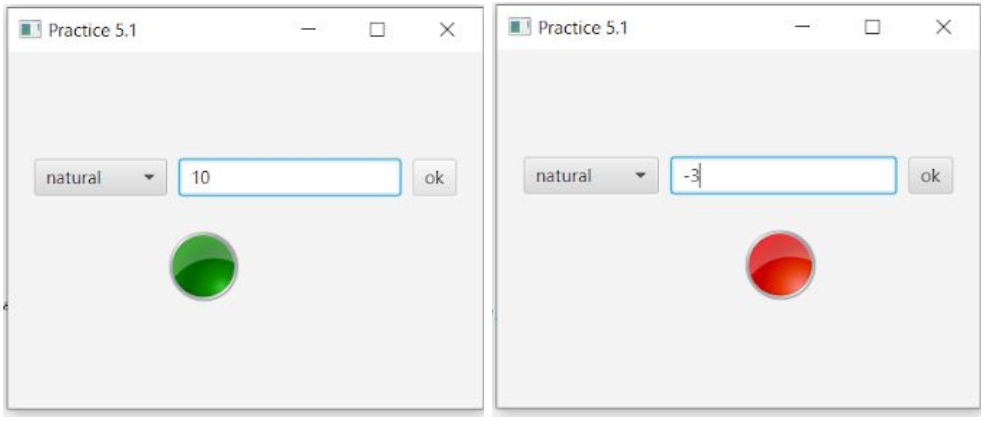

Рис.12.Некоторые тесты для типа данных “натуральное число”.

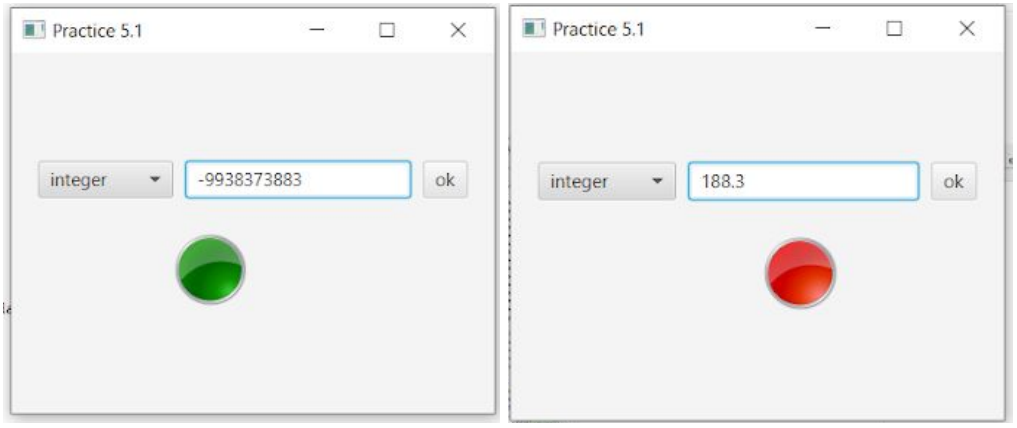

Рис.13.Некоторые тесты для типа данных “целое число”.

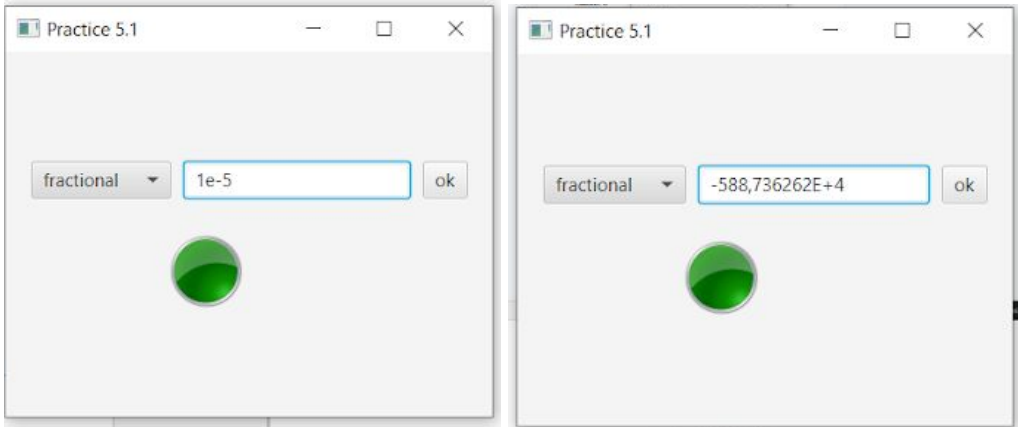

Рис.14.Некоторые тесты для типа данных “число с плавающей запятой”.

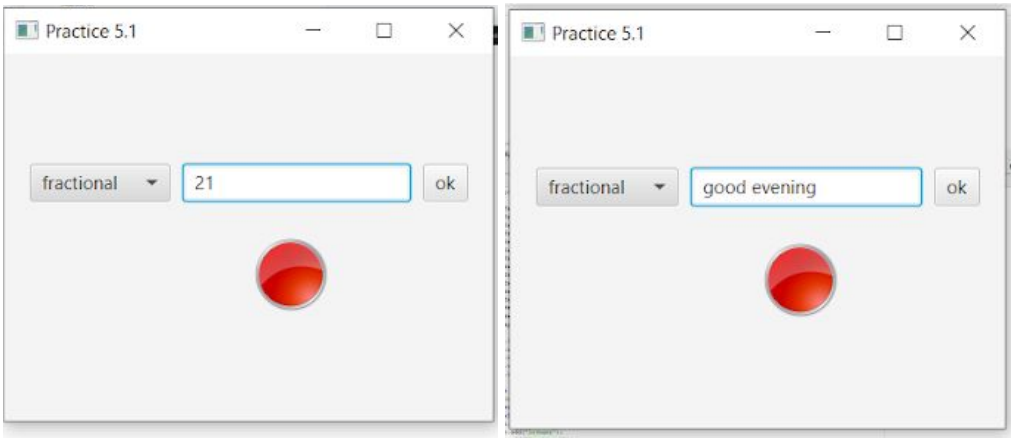

Рис.15.Некоторые тесты для типа данных “число с плавающей запятой”.

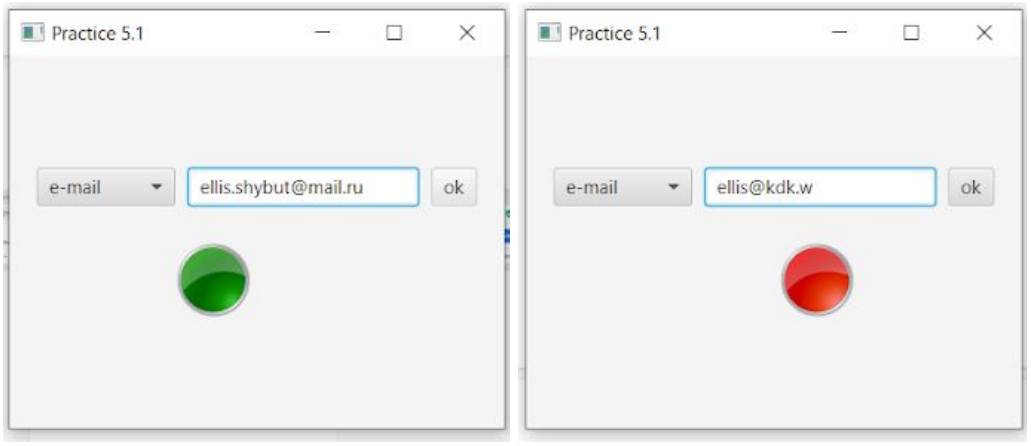

Рис.16.Некоторые тесты для типа данных “электронная почта”.

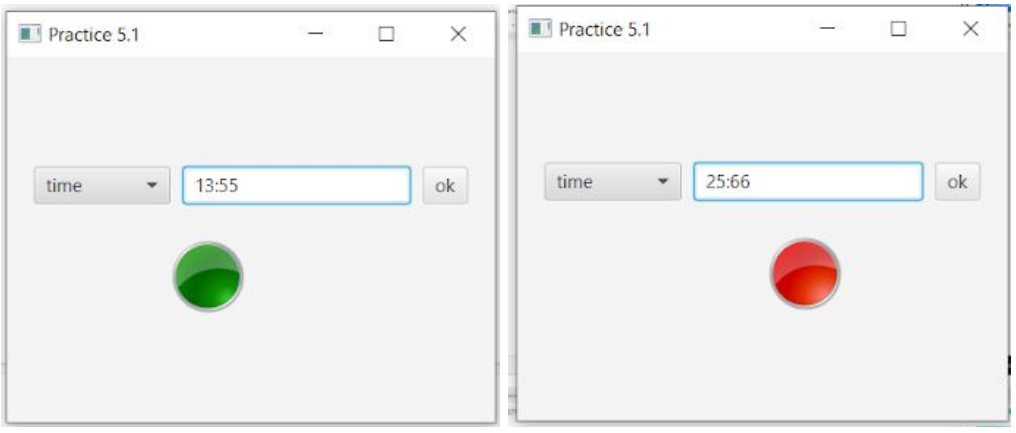

Рис.17.Некоторые тесты для типа данных “время”.

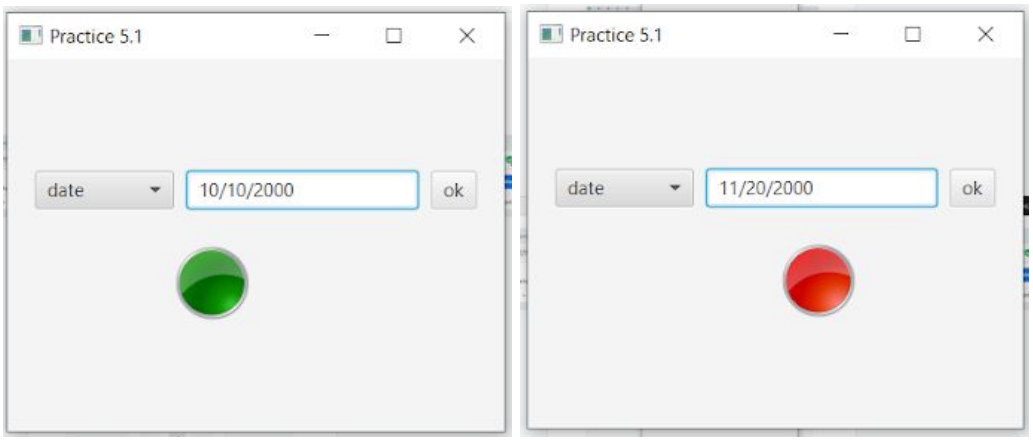

Рис.18.Некоторые тесты для типа данных “дата”.

Задание 2.

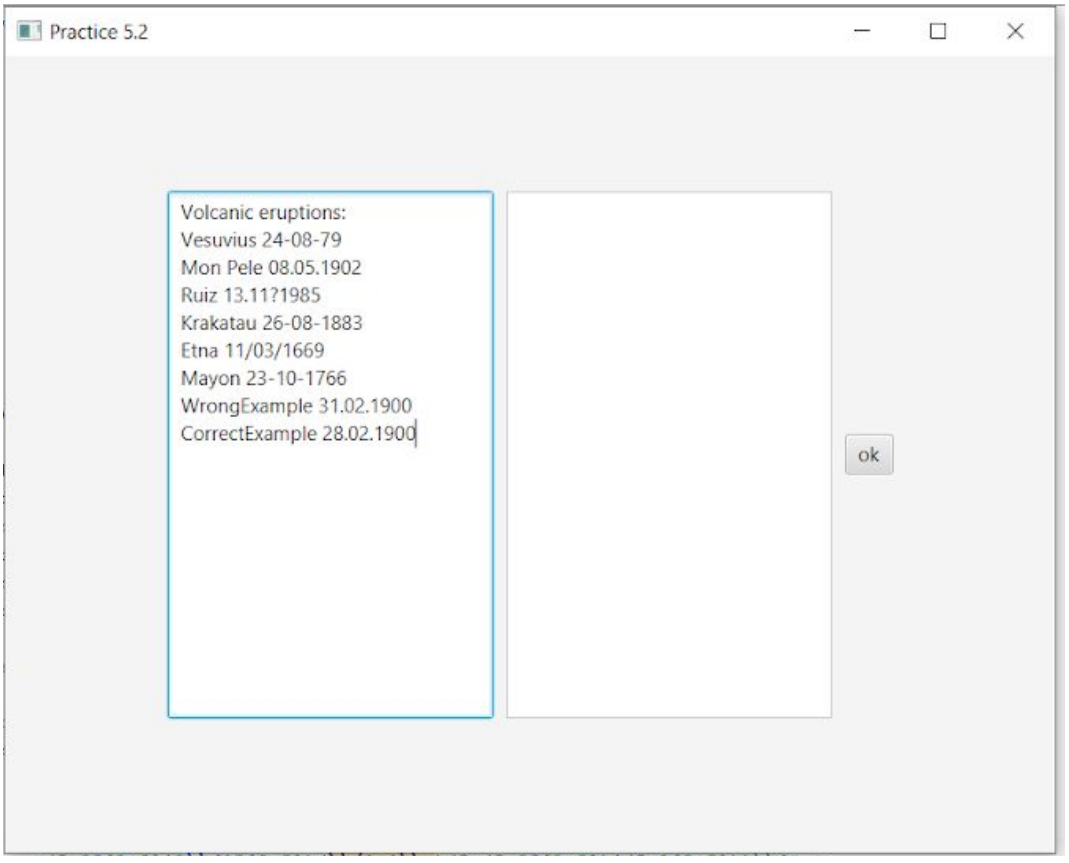

Рис.19.Вид программы после запуска.

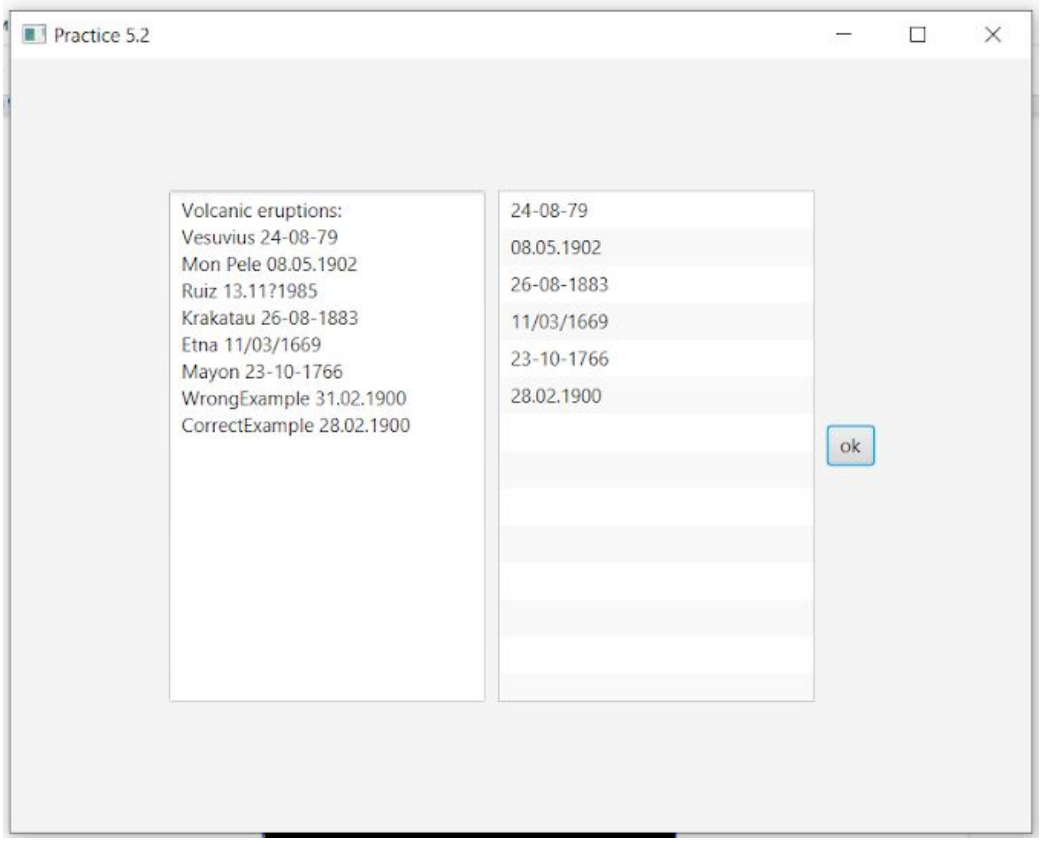

Рис.20.Результат работы оконного приложения.