Лабораторная работа № 3

Цель:

Изучить возможности Java Swing.

Постановка задачи:

Задание 1. Отобразить список. Элемент списка: название страны и флаг. Выбор элемента отображает в метке флаг, название страны и столицу. Столицы со странами хранятся в Map. Флаги можно взять в прилагаемом архиве.

Задание 2. В таблице JTable размещается информация о турпутевках (флаг страны – картинка, описание, цена). В четвертом столбце флажком (JCheckBox) можно выбрать тур. Нужно подсчитать стоимость заказа и отобразить **в таблице**. Реализовать возможности добавления и редактирования тура.

Решение задачи:

Задание 1.

Создадим класс для хранения информации о странах: название, столица, флаг. При добавлении данных будем проверять их на правильность, используя вспомогательный класс “MyException”.

```
Country(String name, String capital) throws MyException {
    if(name == null)
        throw new MyException("Name of country cannot be null!");
    if(capital == null)
        throw new MyException("Name of capital cannot be null!");
    this.name = name;
    this.capital = capital;
    this.flag = getImage();
}
Country(String name) {
    this.name = name;
    this.flag = getImage();
}
public int compareTo(Country o) { return this.name.compareTo(o.getName()); }
private ImageIcon getImage() {
    String str = this.name;
    str = str.toLowerCase();
    return new ImageIcon( filename: "src\\plain\\flag_" + str + ".png");
}
```

Рис.1. Конструкторы и некоторые методы класса “Country”.

Чтение XML-файла осуществим с помощью DOM, разбирая входящие данные на узлы и атрибуты и добавляя новые страны и соответствующие им столицы в “Map”.

```
public boolean open(String path) {
    try {
        model.clear();
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document document = builder.parse(new File(path));
        NodeList countryElements = document.getDocumentElement().getElementsByTagName("country");
        for (int i = 0; i < countryElements.getLength(); i++) {
            Node country = countryElements.item(i);
            // Получение атрибутов каждого элемента
            NamedNodeMap attributes = country.getAttributes();
            model.add(new Country(attributes.getNamedItem("name").getNodeValue(),
                    attributes.getNamedItem("capital").getNodeValue()));
            capMap.clear();
            for (Country c : model) {
                capMap.put(c.getName(), c.getCapital());
            }
        }
        return true;
    } catch (Exception e) {
        view.showError(e);
        return false;
    }
}
```

Рис.2. Метод “open” для чтения XML-файла.

```
<countries>
    <country name ="Estonia" capital ="Tallinn"/>
    <country name ="Belarus" capital ="Minsk"/>
    <country name ="Canada" capital ="Vancouver"/>
    <country name ="Norway" capital ="Oslo"/>
    <country name ="Egypt" capital ="Cairo"/>
    <country name ="France" capital ="Paris"/>
    <country name ="Germany" capital ="Berlin"/>
    <country name ="Latvia" capital ="Riga"/>
    <country name ="Lithuania" capital ="Vilnius"/>
</countries>
```

Рис.3. Пример XML-файла.

В “ComboBox” поместим список стран. Выбирая какую-либо страну по ключу в “Map” получаем доступ к столице, название которой отображается в текстовом поле.

```
comboBox.addActionListener(e ->{
    Map<String, String> map = controller.getCapMap();
    List<Country> value = controller.getModel();
    String key = Objects.requireNonNull(comboBox.getSelectedItem()).toString();
    for (Country c : value) {
        if(map.containsKey(key)){
            nameF.setText(key);
            nameCap.setText(map.get(key));
            if(c.getName().equals(key)){
                flag.setIcon(c.getFlagIcon());
            }

        }
    }
});
```

Рис.4.Отображение столицы и флага при выборе страны.

Задание 2.

Для выполнения следующего задания необходимо создать модель заполнения таблицы, добавив возможность редактировать ее ячейки.

```
DefaultTableModel tableModel = new DefaultTableModel(DATA, NAMES_OF_COLUMNS) {
    public Class getColumnClass(int column) { return getValueAt( row: 0, column).getClass(); }

    public boolean isCellEditable(int row, int column) {
        if (row == this.getRowCount() - 1)
            return false;
        else
            return super.isCellEditable(row, column);
    }
};
tableModel.addRow(EMPTY_COUNTRY);

table.setRowHeight(80);
table.setModel(tableModel);

table.addMouseListener((MouseAdapter) mouseClicked(e) → {
        JTable target = (JTable) e.getSource();
        for(int i =0; i <4; i++){
            if (e.getClickCount() == 1 && target.getSelectedRow() == i && target.getSelectedColumn() == 1) {
                JOptionPane.showMessageDialog( parentComponent: null, DATA[i][1]);
            }
        }

});
```

Рис.5. Модель таблицы.

Справа от таблицы вставим текстовые поля, заполняемые при добавлении в таблицу новой путевки. Проверку входных данных будем осуществлять с помощью условных операторов. В случае ошибки всплывает панель уведомления с подсказкой.

```
addTrip.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            int price = Integer.valueOf(priceField.getText());
            if(price < 0){
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Negative price!");
            }
            else{
                Country cntr = new Country(countryField.getText());
                Object[] obj = new Object[]{cntr.getFlagIcon(), descriptionField.getText(), price, false};
                tableModel.insertRow( row: tableModel.getRowCount() - 1, obj);
            }
        } catch (NumberFormatException exc) {
            JOptionPane.showMessageDialog( parentComponent: null, exc.getMessage());
        }
    }
});
```

Рис.6. Добавление путевок в таблицу.

Реализуем слушателя для таблицы, который будет реагировать на выставление галочки рядом с какой-либо путевкой: если тур отмечен, то суммируем его стоимость с текущей.

```
});
tableModel.addTableModelListener(new TableModelListener() {
    @Override
    public void tableChanged(TableModelEvent e) {
        if (!trigger) {
            int sum = 0;
            for (int i = 0; i < tableModel.getRowCount() - 1; i++) {
                if ((Boolean) tableModel.getValueAt(i,  column: 3)) {
                    sum += (int) tableModel.getValueAt(i,  column: 2);
                }
            }
            trigger = true;
            tableModel.setValueAt(sum,  row: tableModel.getRowCount() - 1,  column: 2);
            trigger = false;

        }
```

Рис.7.Слушатель таблицы, считающий общую стоимость.

Результат:

Задание 1.

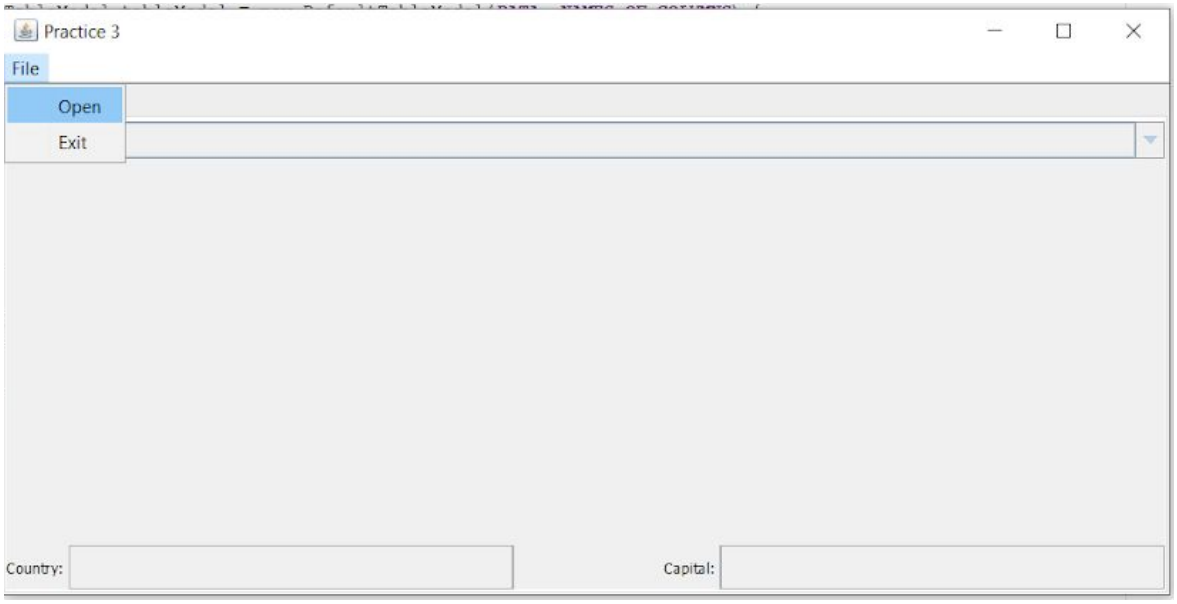

Рис.8.Вид оконного приложения после запуска (вкладка 1).

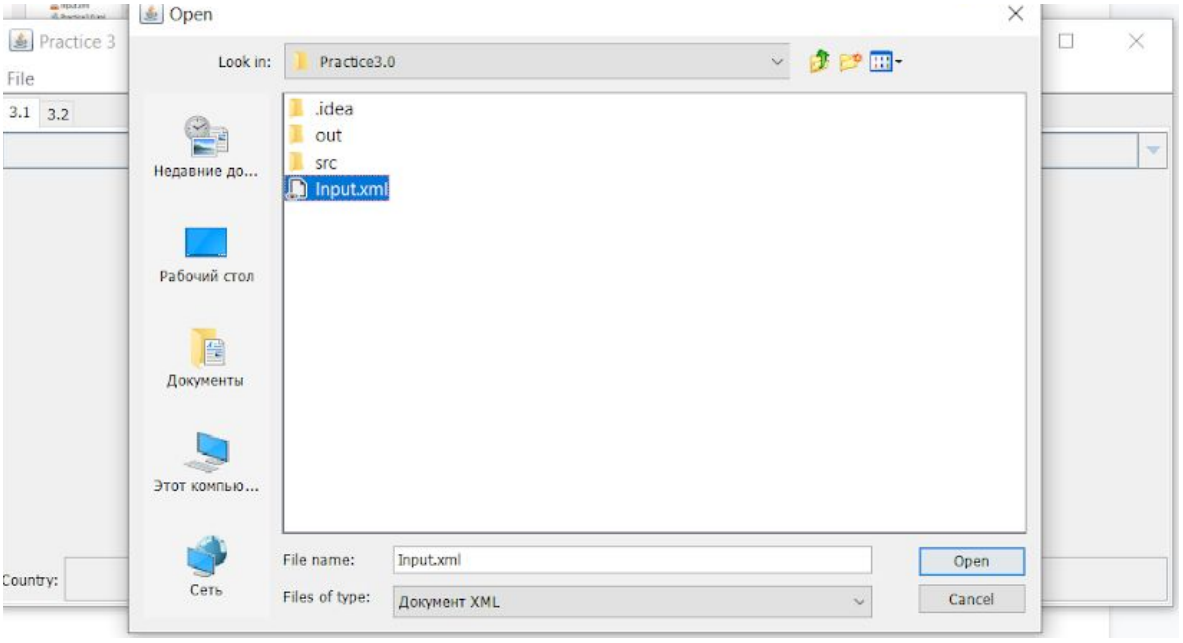

Рис.9. Выбор XML-файла.

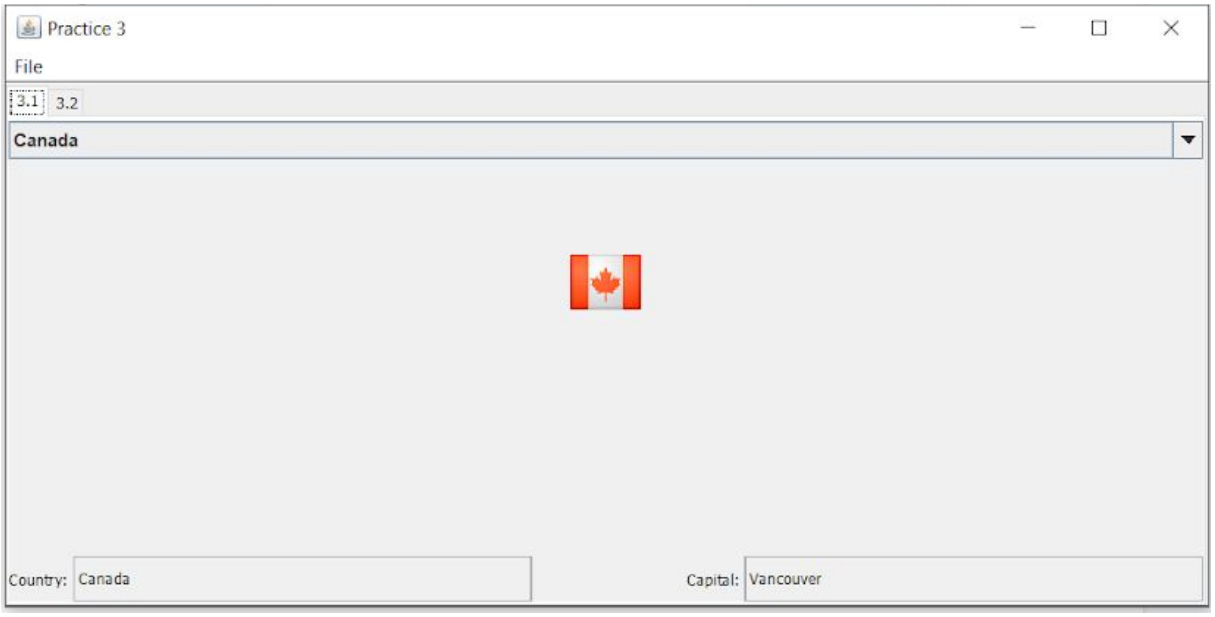

Рис.10. Демонстрация работы программы.

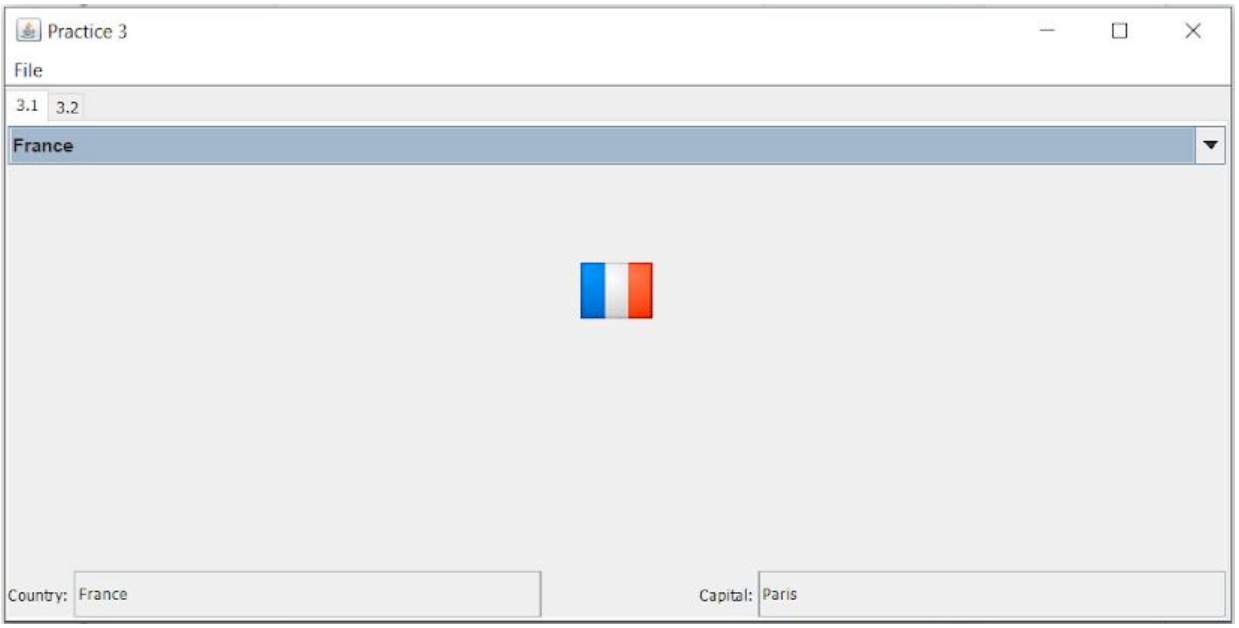

Рис.11. Демонстрация работы программы.

Задание 2.

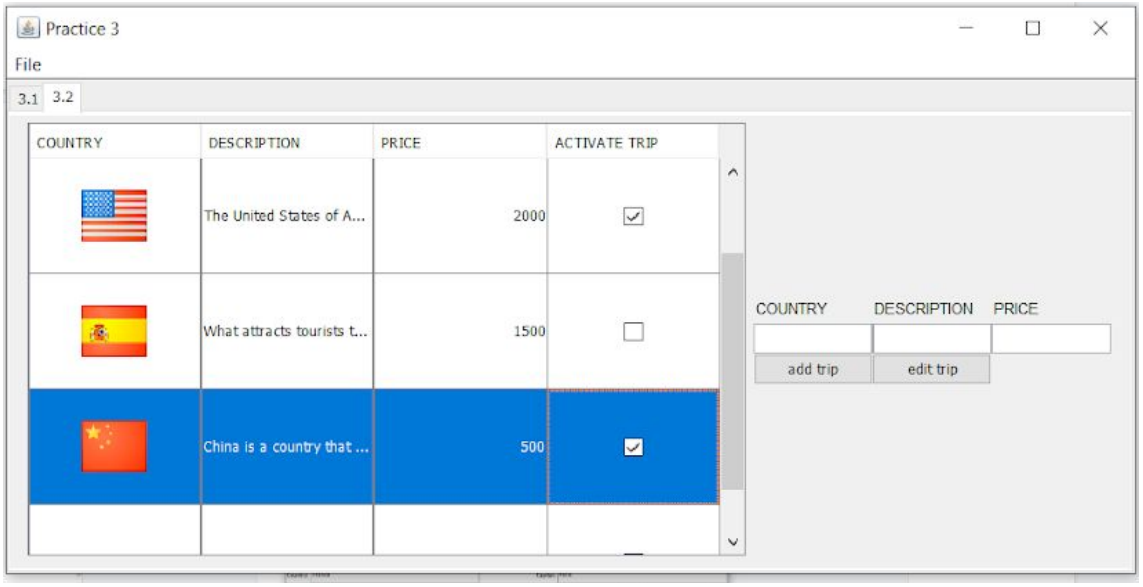

Рис.12. Вид оконного приложения после запуска (вкладка 2).

.

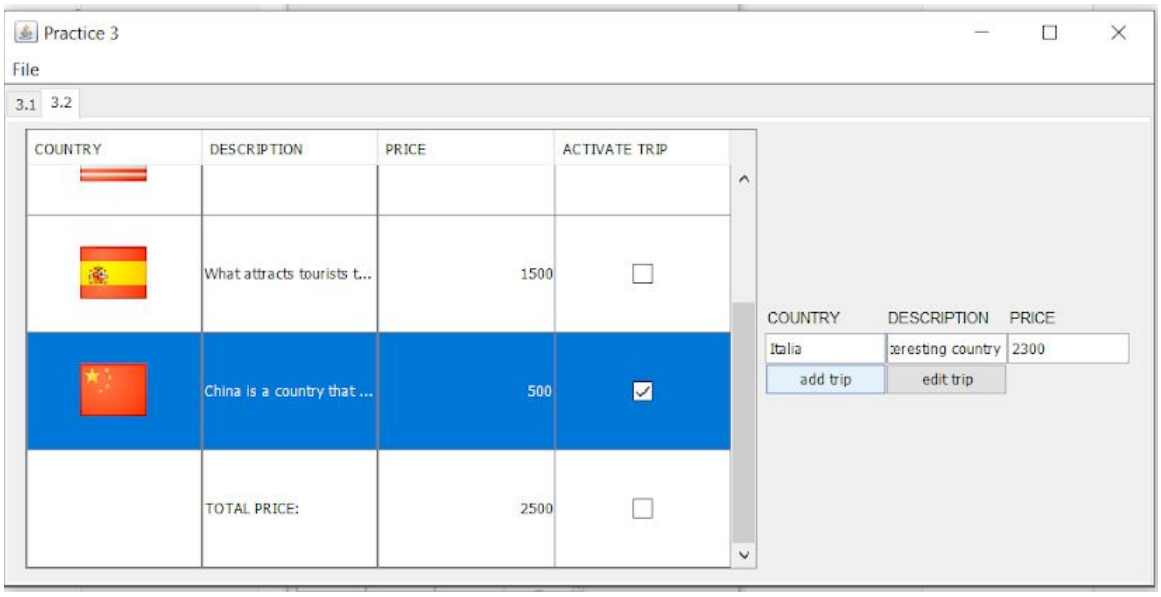

Рис.13. Демонстрация подсчета суммарной стоимости.

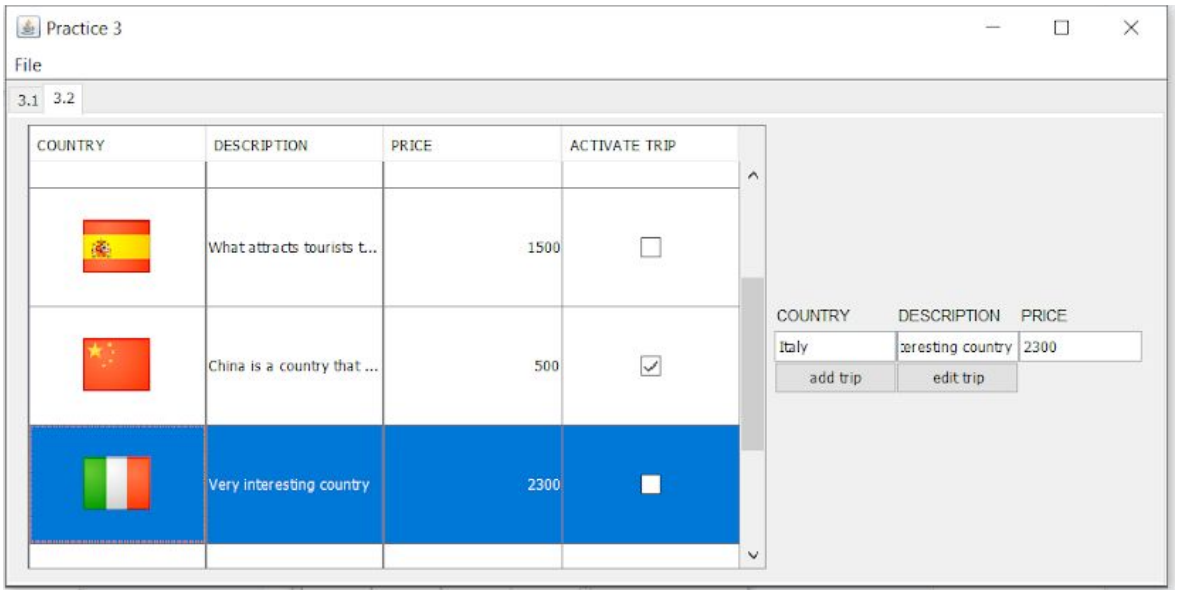

Рис.14. Демонстрация добавления путевки в таблицу.

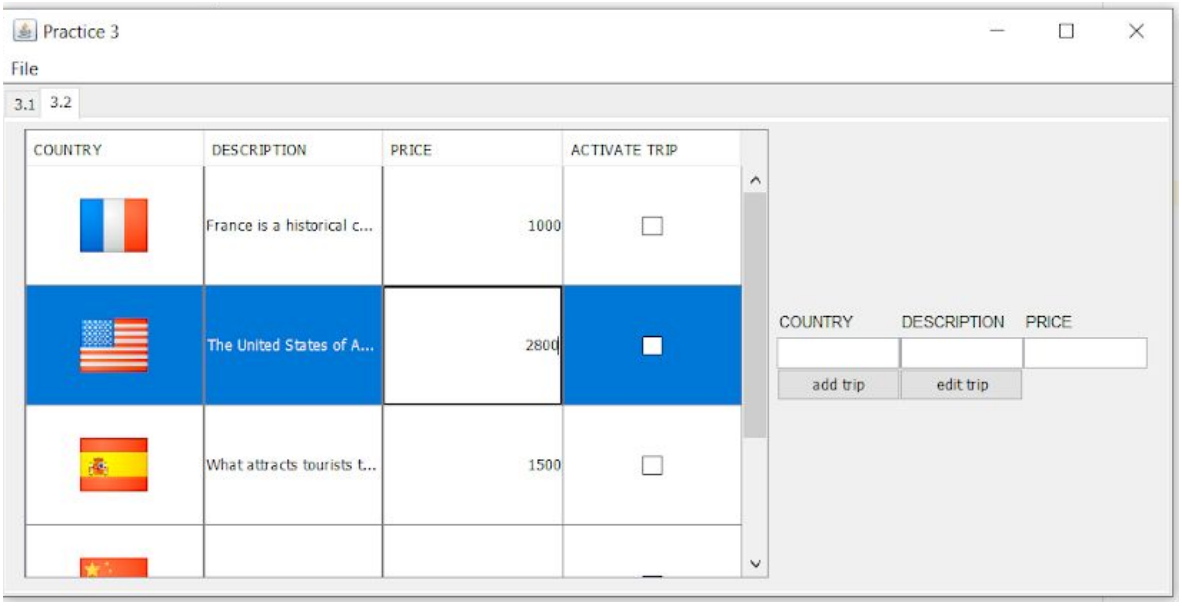

Рис.15. Демонстрация редактирования выбранной ячейки.