Лабораторная работа № 8

Цель:

Изучить паттерн проектирования “Observer”.

Постановка задачи:

В одной части окна отображать крупным планом нажимаемую клавишу, а в другой части вести лог нажатых клавиш.

Решение задачи:

Сначала создаем интерфейс “EventListener” с единственным методом “update(String keyName)”, отвечающий за уведомление наблюдателя.

```
public interface EventListener {
    void update(String keyName);
}
```

Рис.1. Интерфейс “EventListener”.

Для записи лога нажимаемых клавиш описываем класс “LogRecord”, реализующий интерфейс “EventListener”. Полями в данном классе будут служить объекты классов “List<String>”, “ObservableList<String>” и “ListView<String>”. Переопределяем функцию “update” таким образом, что при каждом вызове данного метода в список будет добавляться передаваемое значение.

```
public class LogRecord implements EventListener{
    private final List<String> list;
    private ObservableList<String> observable;
    private ListView<String> listView;

    LogRecord(){
        this.list  = new ArrayList<>();
        this.observable = FXCollections.observableArrayList(list);
        this.listView = new ListView<>(observable);
    }
```

Рис.2. Конструктор класса “LogRecord”.

```
ListView<String> getListView() { return listView; }

@Override
public void update(String keyName){
    observable.clear();
    list.add(keyName);
    observable.addAll(list);
    listView.setItems(observable);
}
```

Рис.3.Переопределение метода “update(String keyName)” в классе “LogRecord”.

Для отображения нажимаемых клавиш создадим класс “DisplayKey”, который также реализует “EventListener” (интерфейс паттерна “Observer”). При создании объекта класса инициализируется “Label”. Метод “update” в данном классе работает аналогично, однако теперь при вызове метода создается файл и, если путь к картинке найден, устанавливаем ее в “labelKey”.

```
public class DisplayKey implements EventListener {
    private Label labelKey;

    DisplayKey(){
        this.labelKey = new Label();
        labelKey.setVisible(false);
    }

    Label getLabelKey() { return labelKey; }

    @Override
    public void update(String keyName) {
        try {
            File file = new File( pathname: "C:\\Users\\Лиза\\IdeaProjects\\Practice8" +
                    "\\src\\standardKeyboardKeys\\" + keyName + ".png");
            String localUrl = file.toURI().toURL().toString();
            Image image = new Image(localUrl);
            labelKey.setGraphic(new ImageView(image));
            labelKey.setVisible(true);
        } catch (MalformedURLException exc){
            exc.getStackTrace();
        }
    }
}
```

Рис.4. Реализация класса “DisplayKey”.

В “Main” создаем объекты описанных выше классов, а для представления их составляющих в окне, вызываем методы “getListView()” и “getLabelKey()”.

После объявления всех переменных и компонентов пользовательского интерфейса назначаем слушателя, который при нажатии какой-либо клавиши вызывает метод “update”, передавая в него в качестве параметра название этой клавиши.

```
public void start(Stage primaryStage) throws Exception{

    Group group = new Group();
    LogRecord log = new LogRecord();
    ListView<String> listView = log.getListView();
    group.getChildren().add(listView);
    DisplayKey displayKey = new DisplayKey();

    FlowPane root = new FlowPane( hgap: 10,  vgap: 10, group, displayKey.getLabelKey());
    Scene scene = new Scene(root,  width: 500,  height: 500);

    scene.setOnKeyPressed(event -> {
        KeyCode key = event.getCode();
        log.update(key.getName());
        displayKey.update(key.getName());
    });

    primaryStage.setTitle("Practice 8");
    primaryStage.setScene(scene);
    primaryStage.show();
}

public static void main(String[] args) { launch(args); }
```

Рис.5. Класс “Main”.

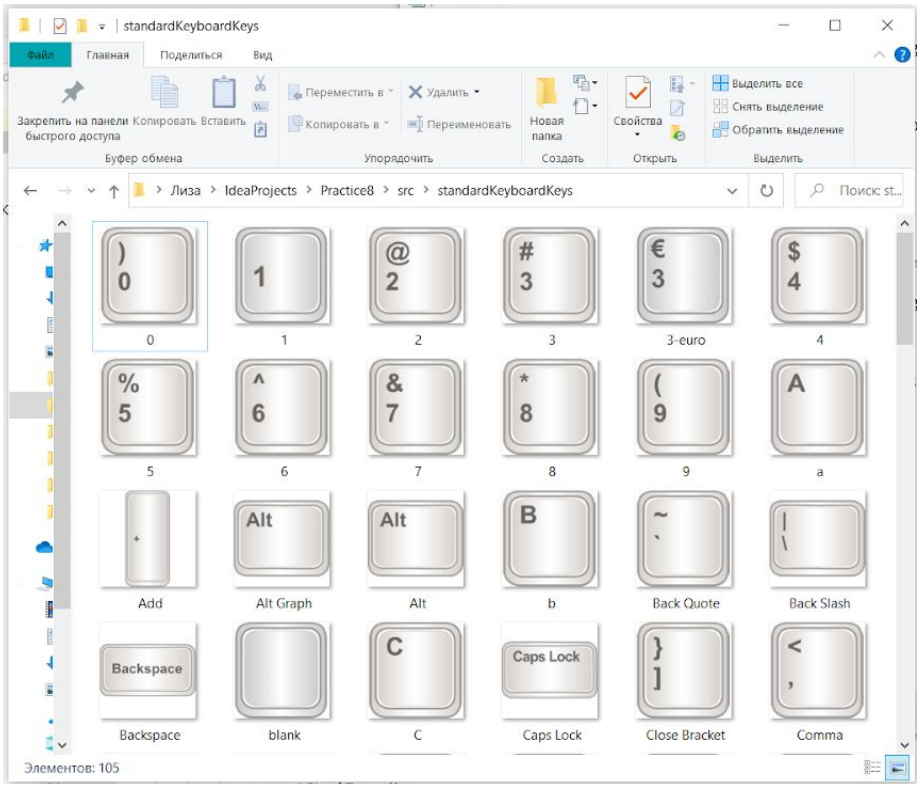

Рис.6.Папка, в которой хранятся изображения кнопок.

Результат:

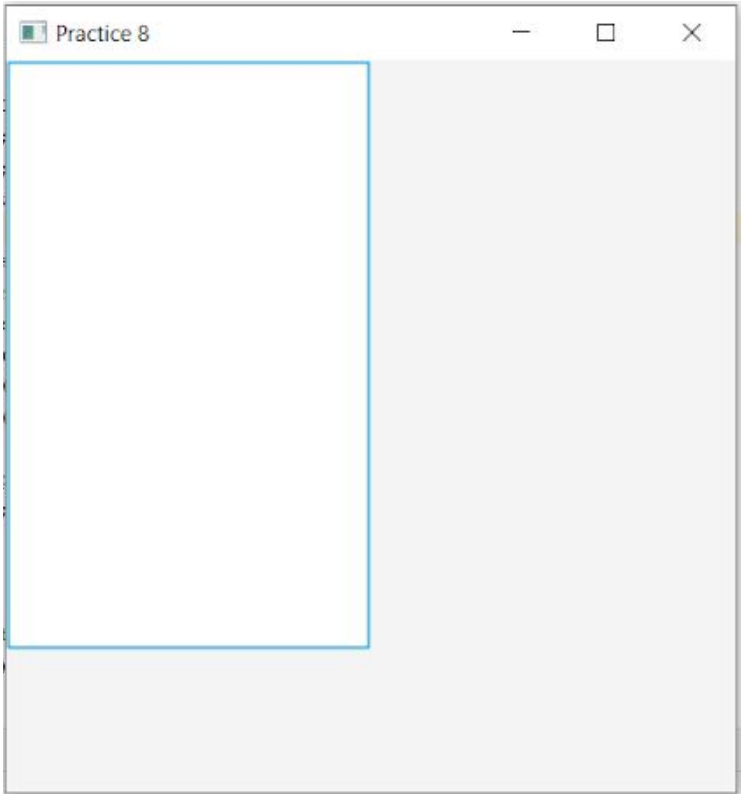

Рис.7.Вид оконного приложения после запуска.

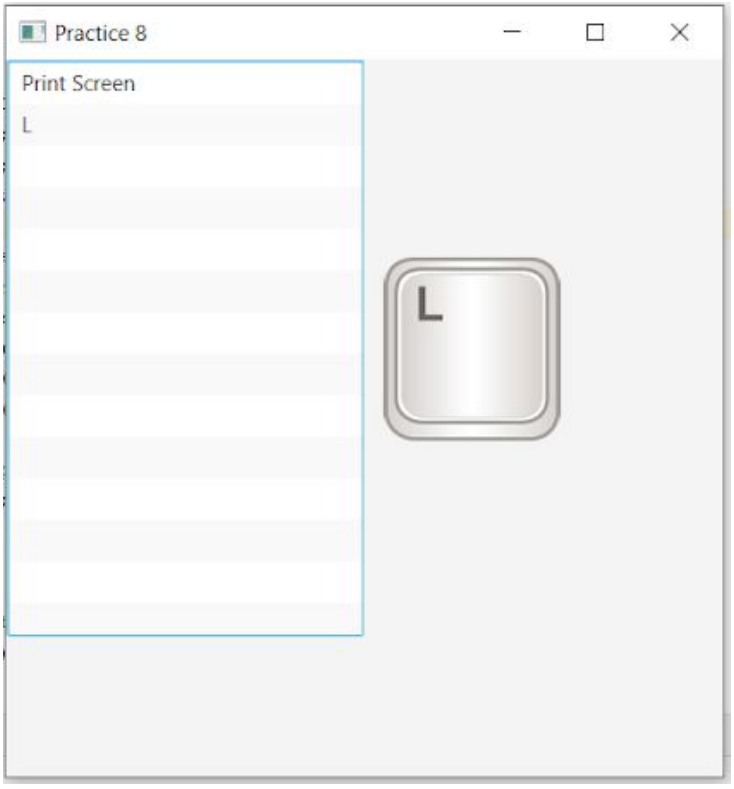

Рис.8.Демонстрация работы программы.

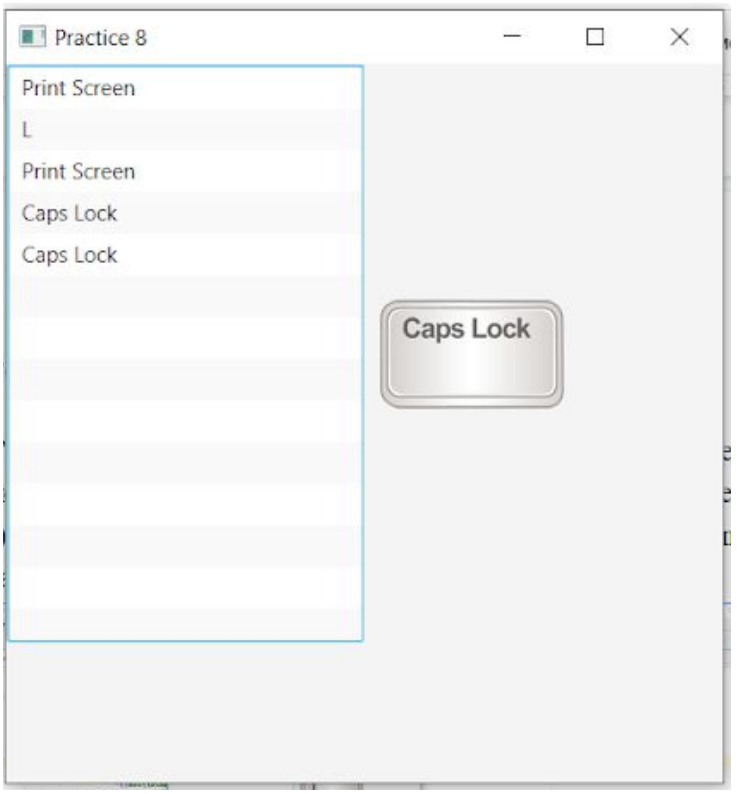

Рис.9.Демонстрация работы программы.

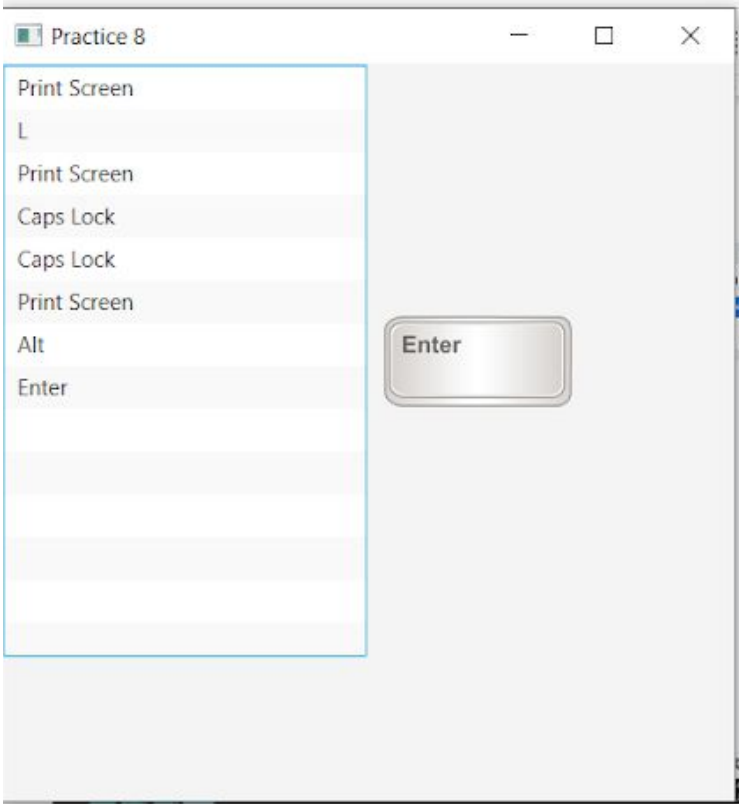

Рис.10.Демонстрация работы программы.

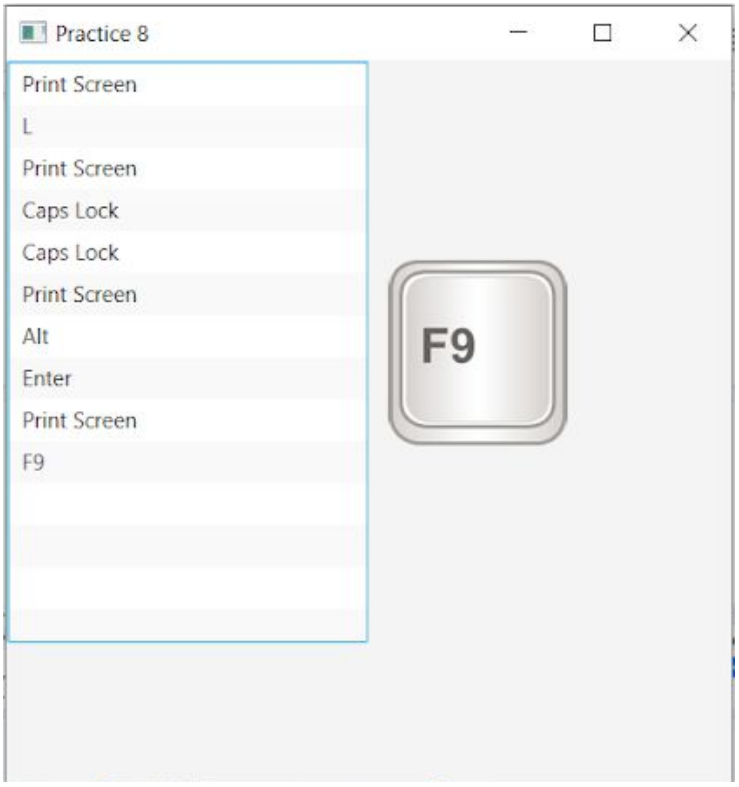

Рис.11.Демонстрация работы программы.

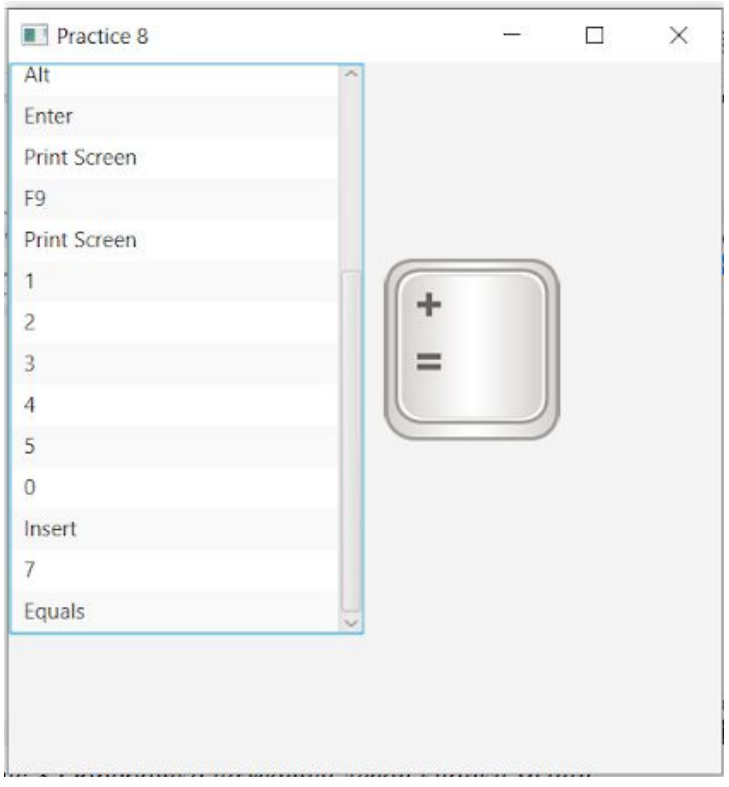

Рис.12.Демонстрация работы программы.

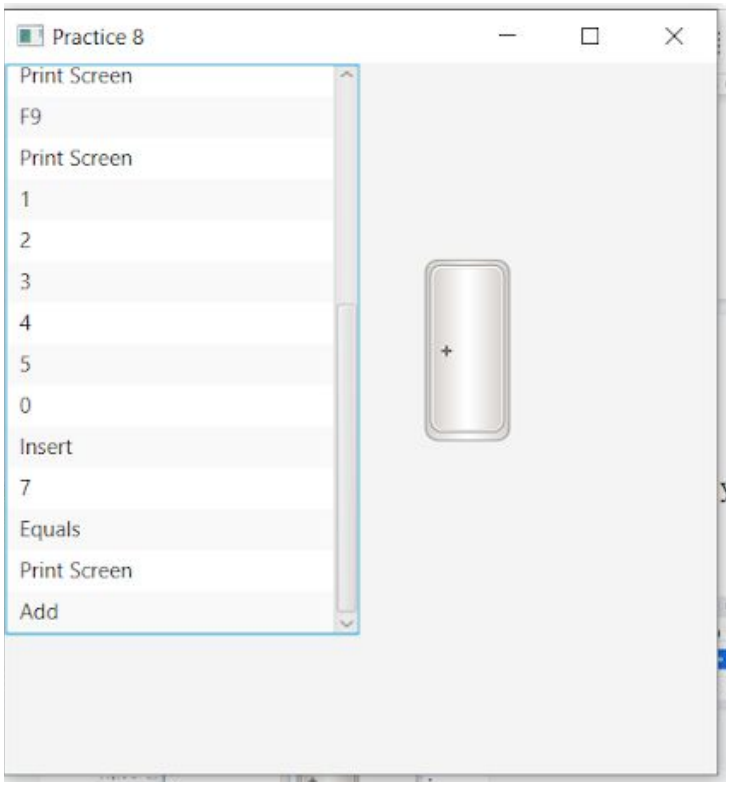

Рис.13.Демонстрация работы программы.

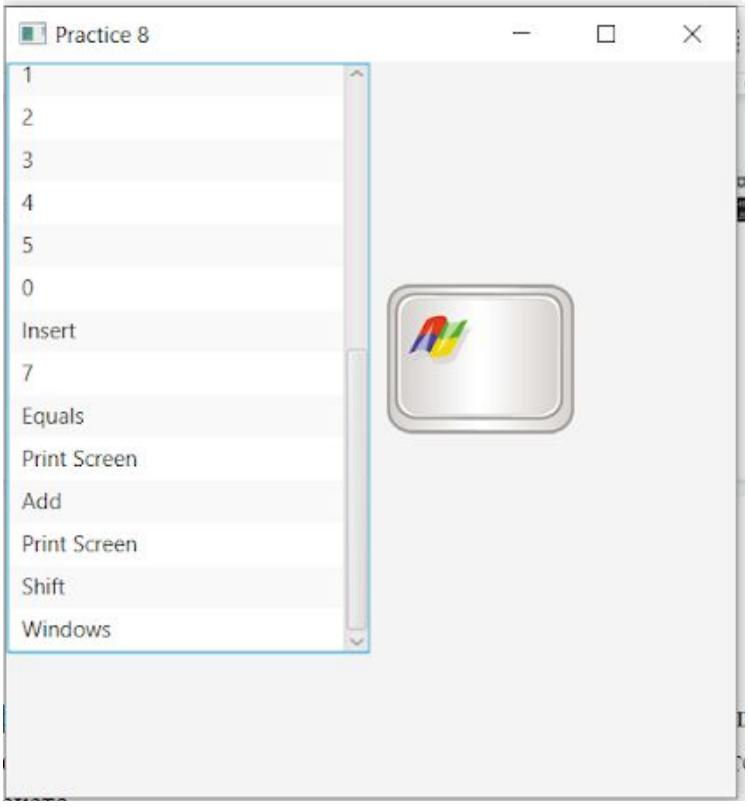

Рис.14.Демонстрация работы программы.