Практическая работа № 15. Вложенные и внутренние классы. Обработка событий в Джава программах с графическим интерфейсом пользователя

Цель данной практической работы - изучить использование анонимных и внутренних классов, научиться разрабатывать интерактивные программы на языке Джава с использованием графического интерфейса пользователя.

Теоретические сведения

В Java вложенные классы — это классы, которые определены внутри другого класса. Цель вложенного класса - четко сгруппировать вложенный класс с окружающим его классом, сигнализируя, что эти два класса должны использоваться вместе. Или, возможно, вложенный класс должен использоваться только изнутри его включающего (владеющего) класса. Разработчики на Джава часто ссылаются на вложенные классы в качестве внутренних классов, но внутренние классы (нестатические вложенные классы) только один из нескольких различных типов вложенных классов в Джава. В Джава вложенные классы считаются компонентами своего включающего класса. Таким образом, вложенный класс не может быть объявлен public, package (без модификатора доступа), protected и private (см. модификаторов доступа для получения дополнительной информации). Следовательно, вложенные классы в Джава также могут наследоваться подклассами. В Джава можно создать несколько различных типов вложенных классов.

Рассмотрим различные типы вложенных классов Java:

- Статические вложенные классы
- Нестатические вложенные классы
- Местные классы
- Анонимные классы

Статические вложенные классы

Статические вложенные классы объявляются в Java следующим образом:

```
public class Outer {
  public static class Nested {
  }}
```

Чтобы создать экземпляр Nested класса, вы должны сослаться на него, добавив к нему префикс имени Outer класса, например:

```
Outer.Nested instance = new Outer.Nested ();
```

Листинг 15.1 – Пример класса Question

```
public class Question {
  private Type type;
  public Type getType() { return type; }
  public void setType(Type type) { this.type = type; }

  public static enum Type {
  SINGLE_CHOICE, MULIT_CHOICE, TEXT
  }

}
```

В языке Джава статический вложенный класс — это, по сути, обычный класс, который только что был вложен в другой класс. Будучи статическим, статический вложенный класс может получить доступ только к переменным экземпляра включающего класса через ссылку на экземпляр включающего класса.

Нестатические вложенные классы

Нестатические вложенные классы в Java также называются внутренними классами. Внутренние классы связаны с экземпляром включающего класса. Таким образом, вы должны сначала создать экземпляр включающего класса, чтобы создать экземпляр внутреннего класса. Вот пример определения внутреннего класса:

```
public class Outer {
  public class Inner { //вложенный класс
  }
}
```

Вот как вы создаете экземпляр Inner класса:

```
Outer outer = new Outer();
Outer.Inner inner = outer.new Inner();
```

Обратите внимание, как вы ставите new после ссылки на внешний класс, чтобы создать экземпляр внутреннего класса. нестатические вложенные классы (внутренние классы) имеют доступ к полям включающего класса, даже если они объявлены закрытыми. Вот пример:

Листинг 15.2 – Пример внутреннего (иннер) класса

```
public class Outer {
    private String text = "I am private!";
    public class Inner {
```

```
            public void printText() {
                System.out.println(text);
            }
        }
    }
```

Вложенные классы делятся на статические (в примере выше StaticNestedClass — это именно такой класс) и нестатические (non-static).

Собственно нестатические вложенные классы имеют и другое название - *внутренние классы (inner classes). Внутренний класс (inner)* является подмножеством вложенного класса (nested classes). *Внешний класс (outer class)* мы иногда будем называть еще обрамляющим классом.

Локальные классы (local inner classes)

Локальные классы (local classes) определяются в блоке кода в программе на языке Джаве. На практике чаще всего объявление происходит в методе некоторого другого класса. Хотя объявлять локальный класс можно внутри статических и нестатических блоков инициализации. Пример использования локального класса:

Листинг 15.3 – Пример локального класса

```
Public class Handler {
Public void handle(String requestPath){
class Path{
List<String> parts = new ArrayList<String>();
String path = "/";
Path(String path) {
if (path == null) return;
this.path = path;
for (String s : path.split("/"))
if(s.trim().length() > 0) this.parts.add(s);
}
int size(){return parts.size();}
String get(int i){
return i > this.parts.size() - 1 ? null :
this.parts.get(i);
}
boolean startsWith(String s){
return path.startsWith(s);
}
```

```
}
Path path = new Path(requestPath);
if(path.startsWith("/page")){
String pageId = path.get(1);
... }
if(path.startsWith("/post")){
   categoryId = path.get(1);
   String postId = path.get(2);
...
 }
...
}}
```

Данный код с некоторыми изменениями взят из реального проекта и используется для обработки get запросов к веб-серверу. Он вводит новую абстракцию, с которой удобно работать в пределах метода и которая не нужна за его пределами.

Как и обычный поля классы, локальные классы ассоциируются с экземпляром обрамляющего класса и имеют доступ к его полям и методам. Кроме этого, локальный класс может обращаться к локальным переменным и параметрам метода, если они объявлены с модификатором final.

У локальных классов есть множество ограничений:

- они видны только в пределах блока, в котором объявлены;
- они не могут быть объявлены как private, public, protected или static;
- они не могут иметь внутри себя статических объявлений (полей, методов, классов); исключением являются константы (static final);

Кстати, интерфейсы тоже нельзя объявлять локально по тем же причинам, по каким их нельзя объявлять внутренними.

Анонимные классы (anonymous inner classes)

Анонимные классы являются важным подспорьем в повседневной жизни программистов на языке Джава. Анонимный класс (anonymous class) — это локальный класс без имени. Классический пример анонимного класса:

```
new Thread(new Runnable() {
     public void run() {
          ...
     }
 }).start();
```

На основании анонимного класса создается поток и запускается с помощью метода start класса Thread. Синтаксис создания анонимного класса базируется на использовании оператора new с именем класса (интерфейса) и телом вновь создаваемого анонимного класса.

Основное ограничение при использовании анонимных классов — это невозможность описания конструктора, так как класс не имеет имени. Аргументы, указанные в скобках, автоматически используются для вызова конструктора базового класса с теми же параметрами. Вот пример:

Листинг 15.4 – Пример анонимного класса

```
Class Clazz {
Clazz(int param) { }
Public static void main(String[] args){
// правильное создание анонимного класса
New Clazz(1) { };
// неправильное создание анонимного класса
New Clazz() { };
}}
```

Так как анонимный класс является локальным классом, он имеет все те же ограничения, что и локальный класс.

Использование анонимных классов оправдано во многих случаях, в частности когда:

- тело класса является очень коротким;
- нужен только один экземпляр класса;
- класс используется в месте его создания или сразу после него;
- имя класса не важно и не облегчает понимание кода.

Остается только сказать, что нужно использовать с умом возможности, которые предоставляют нам вложенные классы, это сделает ваш код чище, красивее и понятнее.

Преимущества использования вложенного класса

Преимущества вложенных классов в языке Джава заключаются в том, что вы можете группировать классы вместе, которые принадлежат друг другу. Вы, конечно, можете группировать их, например поместив в один пакет, но размещение одного класса внутри другого делает такую группировку более сильной.

Вложенный класс, как правило, используется только через класс, в который он вложен. Иногда вложенный класс виден только классу, в который он вложен, используется только внутри и, следовательно, никогда не виден за

пределами этого класса. В других случаях вложенный класс виден за пределами класса, в который он вложен, но не может использоваться без него

Обработка событий в программах с графическим интерфейсом пользователя

Что означает термин обработка событий? В Джава это механизм, который контролирует состояние объекта на предмет наступления события и при его появлении запускает процедуру обработки события. В качестве объектов выступают различные компоненты графического интерфейса, например кнопки или текстовые поля. Это также может быть курсор мыши, нажатие кнопки мыши или нажатие клавиши клавиатуры. То есть, объекты gui это источники событий. Для каждого объекта создаётся слушатель события с помощью специального класса. Слушатель при наступлении события запускает процедуру обработки события.

Интерфейс ActionListener

В Джава есть интерфейс ActionListener. Он получает уведомление всякий раз, когда вы нажимаете кнопку или выбираете пункт меню. Он уведомлен о наступлении события ActionEvent. Интерфейс ActionListener находится в пакете java.awt.event . У него только один метод: actionPerformed (). При создании собственного обработчика вам нужно добавить в свой класс реализацию этого метода.

Метод actionPerformed ()

Синтаксис метода:

```
public abstract void actionPerformed (ActionEvent e);
```

Метод actionPerformed () вызывается автоматически всякий раз, когда вы изменяете состояние объекта, например щелкаете мышкой по кнопке. Как написать ActionListener в своем классе. Для этого вам необходимо выполнить 3 шага:

1. Реализуйте интерфейс ActionListener в классе:

```
public class ActionListenerExample implements
ActionListener
```

2. Зарегистрируйте компонент в Listener (component здесь это тот компонент для которого пишете):

```
component.addActionListener(instanceOfListenerclass);
```

3. Переопределите метод actionPerformed ()

```
public void actionPerformed(ActionEvent e){
//Здесь пишете свой код
}
```

Листинг 15.5 – Пример обработки нажатия на кнопку

```
import java.awt.*;
import java.awt.event.*;
//первый шаг
public class ActionListenerExample implements
ActionListener{
public static void main(String[] args) {
    Frame f=new Frame("ActionListener Example");
    final TextField tf=new TextField();
    tf.setBounds(50,50, 150,20);
    Button b=new Button("Click Here");
    b.setBounds(50,100,60,30);
    //второй шаг
    b.addActionListener(this);
    f.add(b);f.add(tf);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
//третий шаг
public void actionPerformed(ActionEvent e){
            tf.setText("Добро пожаловать в мир
Джава.");
}
}
```

Результат выполнения программы на листинге 15.5 вы можете увидеть на рис. 15.1.

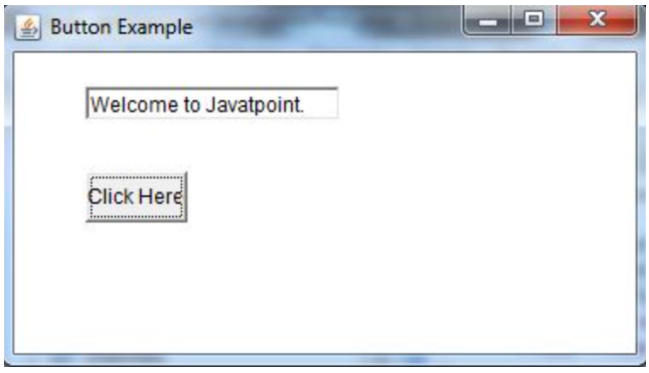

Рисунок 15.1. Пример работы программы на листинге 15.4

Использование анонимных классов

Рассмотрим пример реализации ActionListener через анонимный класс:

Мы также можем использовать анонимный класс для реализации

ActionListener, вместо способа, рассмотренного в листинге 15.1 Это сокращенный способ, поэтому вам теперь не нужно выполнять все три шага описанные выше. Добавляем слушателя к кнопке таким образом:

```
b.addActionListener ( новый  ActionListener () {
public void  actionPerformed (ActionEvent e) {
    tf.setText ( «Добро пожаловать в мир Java.» );
 }
});
```

Листинг 15.6 – Пример программы с анонимным классом

```
import java.awt.*;
import java.awt.event.*;
public class ActionListenerExample {
    public static void main(String[] args) {
        Frame f=new Frame("ActionListener Example");
        final TextField tf=new TextField();
        tf.setBounds(50,50, 250,20);
        Button b=new Button("Click Here");
        b.setBounds(50,100,60,30);
        b.setSize(100,50);

        b.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent e){
                tf.setText("Добро пожаловать в мир
Джава.");
            }
        });
        f.add(b);f.add(tf);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

Результат работы программы на листинге 15.6 представлен на рис.15.2.

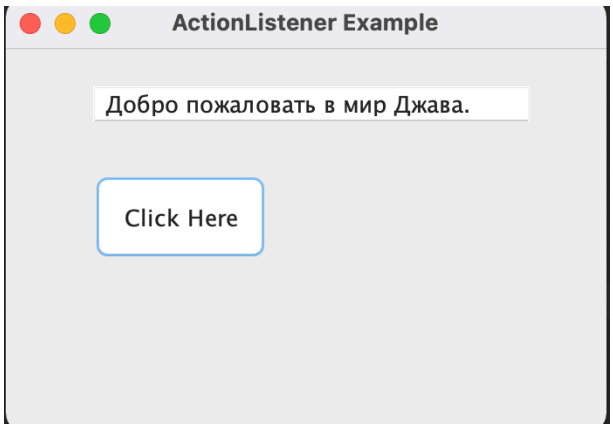

Рисунок 15.2 Пример работы программы на листинге 15.5

Компонент JTextArea

Компонент TextAreas похож на TextFields, но в него можно вводить более одной строки. В качестве примере TextArea можно рассмотреть поле ввода текста, который мы набираем в теле сообщения электронной почты.

Листинг 15.7 – Пример с полем ввода текста

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class TextAreaExample extends JFrame{
JTextArea jta1 = new JTextArea(10,25);
JButton button = new JButton("Add some Text");
public TextAreaExample(){

super("Example");
setSize(300,300);
setLayout(new FlowLayout());
add(jta1);
add(button);
button.addActionListener(
 new ActionListener(){
 public void actionPerformed(ActionEventae){
 String txt = JOptionPane.showInputDialog (
null,"Insert some text");
jta1.append(txt);
}
});
}
public static void main(String[]args){
new TextAreaExample().setVisible(true);
  }
}
```

На рис. 15.3 представлен пример выполнения программы, представленной на листинге 15.7.

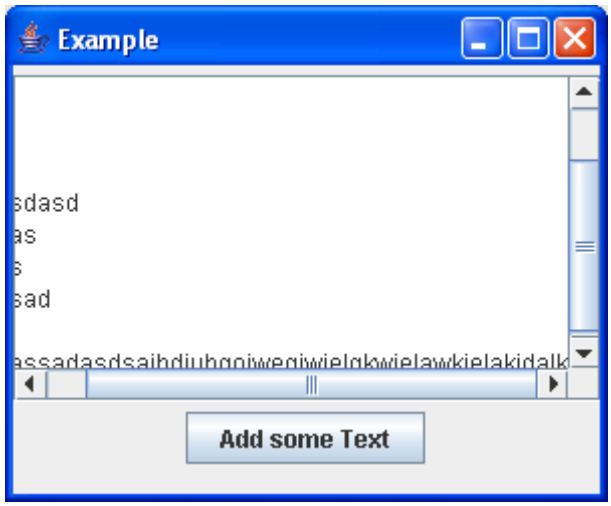

Рисунок 15.3. Пример работы программы на листинге 15.7

Рассмотрим еще один пример программы, для сложения двух чисел.

Листинг 15.8 – Пример программы сложения 2 чисел

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class LabExample extends JFrame{
JTextField jta1 = newJTextField(10);
JTextField jta2 = new JTextField(10);
JButton button = new JButton(" Add themup");
Font fnt = new Font("Times new roman",Font.BOLD,20);
//начало конструктора класса LabExample
LabExample(){
super("Example");
setLayout(new FlowLayout());
setSize(250,150);
// создаем надпись 1
add(new JLabel("1st Number"));
// добавляем текстовое поле jta1
add(jta1);
// создаем надпись 2

add(new JLabel("2nd Number"));
// добавляем текстовое поле jta2
add(jta2);
// добавляем кнопку
add(button);
```

```
    //добавляем слушателя к кнопке
    button.addActionListener(new ActionListener(){
    // добавляем реализацию actionPerformed
    public void actionPerformed(ActionEventae){
    try{
    //переменная для хранения ввода в текстовое поле 1
    double x1 = Double.parseDouble(jta1.getText().trim());
    //переменная для хранения ввода в текстовое поле 2
    double x2 = Double.parseDouble(jta2.getText().trim());
    //создаем всплывающее окно INFORMATION_MESSAGE
    JOptionPane.showMessageDialog(null, "Result =
    "+(x1+x2),"Alert",JOptionPane.INFORMATION_MESSAGE);
    }catch(Exception e){
    //создаем всплывающее окно ERROR_MESSAGE
    JOptionPane.showMessageDialog( null, "Error in Numbers
!","alert" , JOptionPane.ERROR_MESSAGE);
        }
     }
    });// конец button.addActionListener()
    } // конец конструктора
    setVisible(true);
    }
    public static void main(String[]args){
    new LabExample();
     } // конец main()
    } // конец класса LabExample
```

На рис. 15.4, 15.5, 15.6, 15.7 мы видим результат работы программы, представленной на листинге 15.8

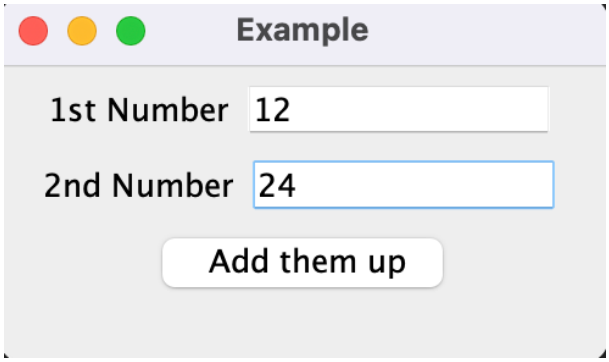

Рисунок 15.4. Пример работы программы на листинге 15.8

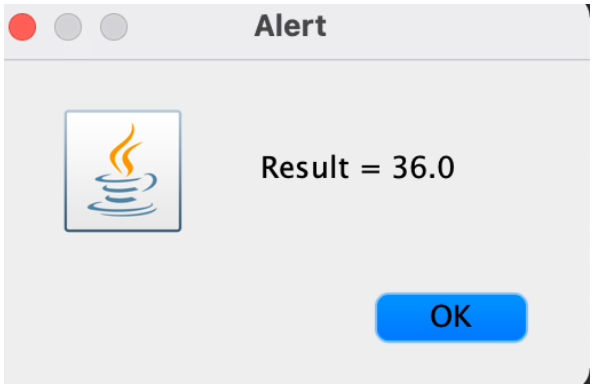

Рисунок 15.5. Пример работы программы на листинге 15.6

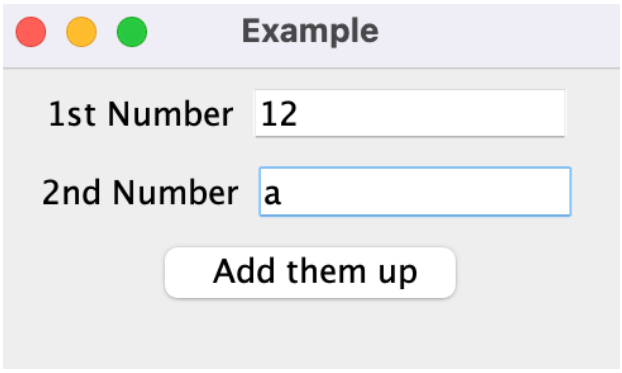

Рисунок 15.6. Пример работы программы на листинге 15.6

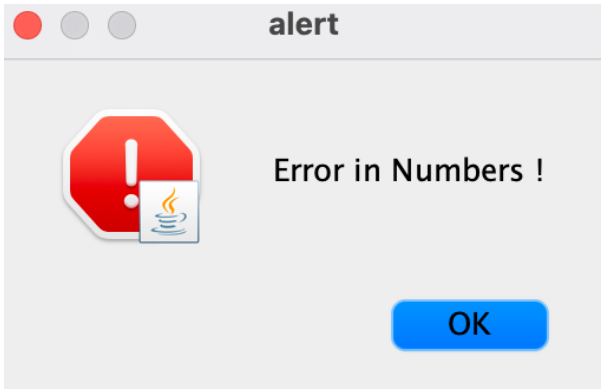

Рисунок 15.7. Пример работы программы на листинге 15.6

Замечание.

Мы можем легко добавить возможность прокрутки к текстовому полю, добавив его в контейнер с именем JScrollPane следующимобразом:

```
JTextArea txtArea = new JTextArea(20,20) ;
JScrollPane jScroll = new JScrollPane(txtArea);
// …
/* мы добавим полосу прокрутки панели (scrollPane) а не объект TextArea*/
add(Scroll);
```

Попробуйте выполнить сами!

Задания на практическую работу № 15

1. Напишите программу калькулятор, используя пример в листинге 15.6. Реализуйте помимо сложения вычитание, деление и умножение для двух чисел, которые вводятся с клавиатуры.

2. Разработайте программу выбора меню как на рис. 15.8 ниже. Вам понадобится JComboBox.

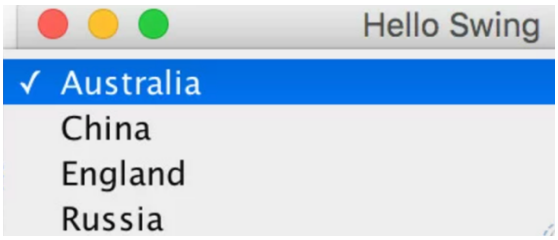

Рисунок 15.8. Окно выбора меню

При выборе пункта меню должна выводится информация о стране

3. Разработайте программу с меню, двумя кнопками и текстовым полем ввода. В этой программе у вас должны быть разные настройки в меню. Должно быть меню «Файл», которое включает в себя подменю «Сохранить», «Выйти» и «Правка», включая подменю «Копировать, вырезать, вставить» и меню «Справка». Вид окна программы должен иметь вид как на рис.15.9 ниже.

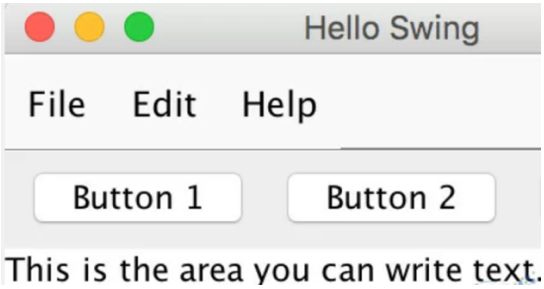

Рисунок 15.9. Окно выбора меню

Замечание. Для выполнения задания вам понадобятся следующие классы: GridLayout, JButton, JFrame, JMenu, JMenuBar, JMenuItem, JPanel, JTextArea;

4. Разработайте программу калькулятора вида как представлено на рис. 15.10 ниже

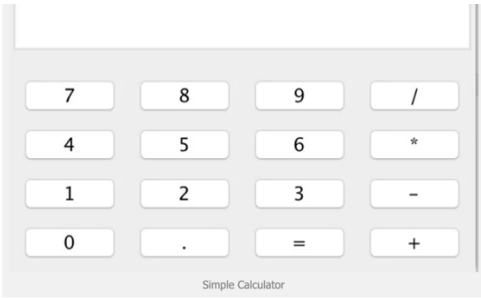

Рисунок 15.10. Окно программы калькулятор