Обработка событий в Java

Модель обработки событий

- Модель обработки событий построена на основе стандартного шаблона проектирования ООП Observer/Observable.
- В качестве наблюдаемого объекта выступает тот или иной компонент AWT или Swing. Он является источником события.
- Для источника события можно задать один или несколько классов-наблюдателей. Они называются слушателями (listener) и описываются специальными интерфейсами, название которых оканчивается на слово Listener.
- Когда с наблюдаемым объектом что-то происходит, создается объект "событие" (event), который "посылается" всем слушателям этого события. Так слушатель узнает, например, о действии пользователя и может на него отреагировать.

- Каждое событие является подклассом класса java.util.EventObject.
- События пакета AWT, являются подклассами java.awt.AWTEvent.
- Для удобства классы различных событий и интерфейсы слушателей помещены в отдельный пакет java.awt.event

ActionEvent

- Появляются в компонентах
  - Button
  - JButton
  - List
  - TextField
  - JComboBox
  - JTextField
  - …
- Обработчик события должен реализовывать интерфейс ActionListener

Пример

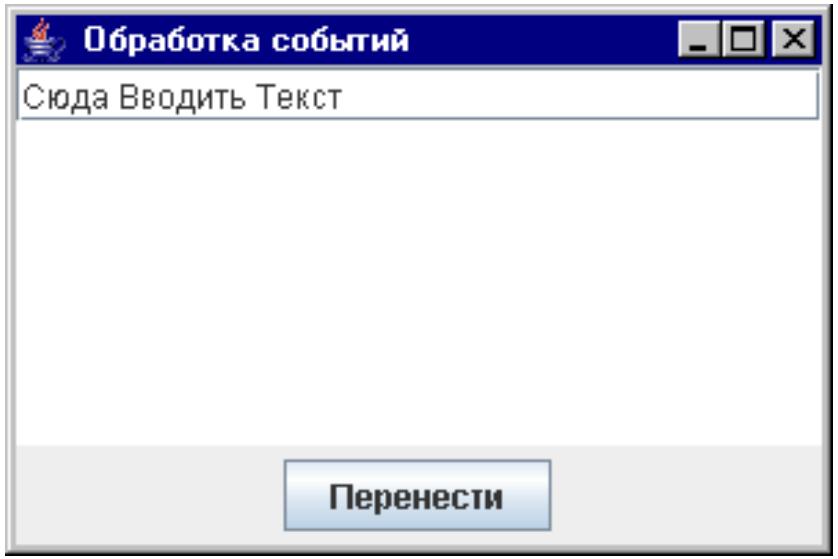
- Опишем обработчик событий, который будет перемещать строку из области ввода типа JTextField и присоединять ее к области вывода текста типа JTextArea

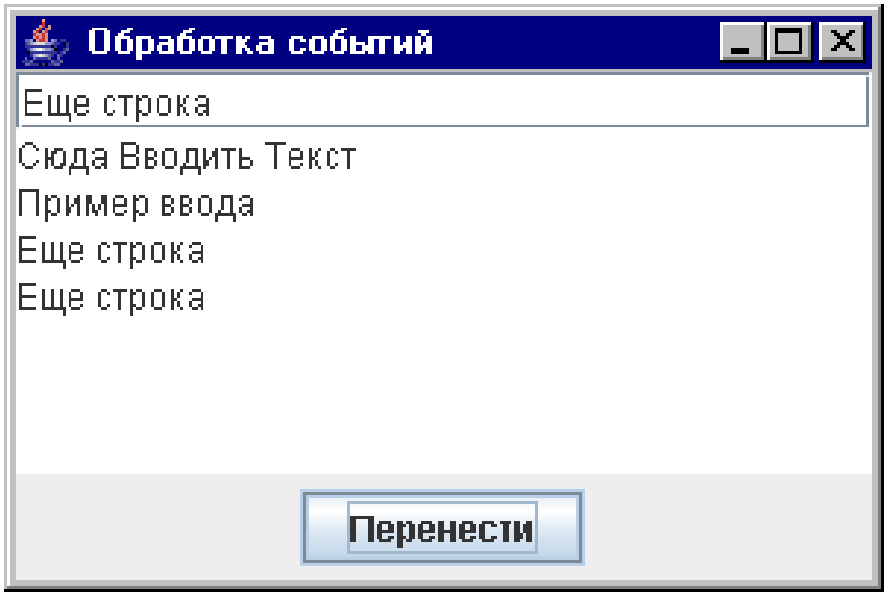
```
class TextMove implements ActionListener {
  private JTextField tf;
  private JTextArea ta;
  TextMove (JTextField f, JTextArea a) {
        tf=f; ta=a;
  }

  public void actionPerformed(ActionEvent e) {
    ta.append(tf.getText()+"\n");
}}
```

В контейнер JFrame поместим два текстовых поля – для ввода JTextField и для вывода JTextArea и кнопку JButton.

После ввода в поле JTextField строки и нажатия клавиши <Enter>, строка переносится в область JTextArea.

Такой же эффект имеет нажатие на кнопку JButton.


Обработка событий
Сюда Вводить Текст
Перенести


Обработка событий
Еще строка
Сюда Вводить Текст
Пример ввода
Еще строка
Еще строка
Перенести

```
class MyNoteBook extends JFrame {
.....
   Container c  = getContentPane();
   JTextField tf = new JTextField ("Сюда Вводить Текст",50);
   c.add(tf,BorderLayout.NORTH);
   JTextArea ta = new JTextArea();
   ta.setEditable(false);
   c.add(ta);
```

```
JPanel p = new JPanel();
 c.add(p,BorderLayout.SOUTH);
 JButton b = new JButton("Перенести");
 p.add(b);

 TextMove tm = new TextMove(tf,ta);
 tf.addActionListener(tm);
 b.addActionListener(tm);
```

Самообработка событий

- Класс, содержащий источник события, может сам его обрабатывать.
- Для этого он должен реализовывать соответствующий интерфейс, например, ActionListener

class MyNoteBook extends JFrame

implements ActionListener {

…..

tf.addActionListener(this);

b.addActionListener(this);

}

Добавить метод

```
public void actionPerformed(ActionEvent e) {
    ta.append(tf.getText()+”\n”);
}
```

Обработка внутренним анонимным классом

```
JButton b = new JButton(“Перенести”);
b.addActionListener(new ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    processMove(); / / метод внешнего класса
  }
});
```

Классы-адаптеры

- Представляют пустую реализацию интерфейсов-слушателей, имеющих более одного метода.
- Например, для действий с мышью
  - MouseAdapter implements MouseListener
  - MouseMotionAdapter implements MouseMotionListener