Обработка событий

Модель делегации событий: источник генерирует событие и посылает его одному или более слушателям.

Слушатель – это объект, уведомляемый о возникновении события. Он должен быть зарегистрирован одним или более источниками событий , чтобы получать извещения о событиях, а также должен реализовывать методы для получения и обработки этих событий.

Слушатель ожидает до тех пор, пока не получит событие. Если слушатель получил событие, то обрабатывает его и возвращает управление (делегирование обработки события отдельному фрагменту кода). Слушатели должны регистрироваться источником для того, чтобы получать извещения о событиях.

Событие – это объект, описывающий изменение состояния источника (м.б. сгенерирован в результате взаимодействия пользователя с элементом граф. польз. интерфейса – нажатие экранной кнопки, ввод символа, клик мышкой и др.).

Источник события - это объект , генерирующий событие. Источник может генерировать события разных типов. Источник должен регистрировать слушателей, чтобы слушатели могли получать извещения о событиях.

Классы событий

Суперкласс всех событий **EventObject** пакет **java.util.**

Суперкласс всех событий AWT **AWTEvent** пакет **java.awt** (использует модель делегации событий)**.**

События **AWT** пакет **java.awt.event**

ActionEvent – нажатие экранной кнопки, двойной щелчок на элементе списка, выбор пункта меню.

Константы:
ALT_MASK, CTRL_MASK, META_MASK, SHIFT_MASK ,
ACTION_PERFORMED – идентификация события.

Конструкторы:
ActionEvent(Object src, int type, String cmd)
ActionEvent(Object src, int type, String cmd, int mod)
ActionEvent(Object src, int type, String cmd, long when, int mod)

src – ссылка на объект, генерирующий событие;
type – тип события;
cmd – командная строка ;
mod – если было нажатие ALT, CTRL, SHIFT, META;
when – когда произошло.

Методы:
String getActionCommand() – возвращает имя команды источника события (для кнопки – название кнопки)

int getModifiers() – значение нажатого модификатора ALT, CTRL и др.

long getWhen() – возвращает время совершения события.

2. KeyEvent – клавиатурный ввод.
Типы событий (константы):
CHAR_UNDEFINED: KEY_PRESSED, KEY_RELEASED – генерируются при нажатии и отпускании клавиши,
KEY_TYPED – при вводе символа.

Другие константы: (коды всех клавиш, не завис. от модиф.)
VK_DOWN VK_LEFT VK_RIGHT VK_UP
VK_ALT VK_SHIFT VK_CONTROL
VK_PAGE_DOWN VK_PAGE_UP
VK_ESCAPE VK_CANCEL VK_ENTER

VK_0 … VK_9 VK_A ….VK_Z

Конструктор:

KeyEvent(Component src, int type, long when, int mod,
int code, char ch)

src – источник события;
type – тип события;
when – время, когда была нажата клавиша;
mod – модификаторы типа ALT, CTRL и др.;
code – код виртуальной клавиши;
ch – символ, если была нажата символьная клавиша

Если нажат не символ, то **ch** содержит **CHAR_UNDEFINED,**
если событие **KEY_TYPED**, то **code** содержит **VK_UNDEFINED**.

Методы:

char getKeyChar() – возвращает введенный символ (CHAR_UNDEFINED, если символа нет);

int getKeyCode() – возвращает код виртуальной клавиши.

3. MouseEvent – события мыши.

Типы событий:
MOUSE_CLICKED – кликнули мышью;
MOUSE_DRAGGED – перетащили мышь (с нажатием);
MOUSE_ENTERED – курсор мыши вошел в компонент;
MOUSE_EXITED – курсор мыши покинул компонент;
MOUSE_MOVED – мышь перемещена;
MOUSE_PRESSED – нажата кнопка мыши;
MOUSE_RELEASED – отпущена кнопка мыши;
MOUSE_WHEEL – прокрутка колесика мыши.

Конструктор:
MouseEvent(Component src, int type, long when, int mod, int x, int y, int clicks, boolean Popup)
x и **y** – координаты курсора мыши;
clicks – счетчик щелчков;
Popup – должно ли быть всплывающее меню.

Методы:

int getX() и **int getY()** - координаты курсора мыши.
(аналог Point getPoint() – объект, содержащий координаты)

int getClickCount() – количество щелчков мыши;

boolean isPopupTrigger() – будет ли всплывающее меню;

int getButton() – возвращает значение, представляющее кнопку, вызвавшую событие (NOBUTTON, BUTTON1, BUTTON2, BUTTON3).

Point getLocationOnScreen()

int getXOnScreen()

int getYOnScreen() - координаты мыши относительно экрана, а не компонента.

WindowEvent – закрытие, открытие, свертывание, активизация (и др.) окна

AdjustmentEvent – событие, связанное с линейкой прокрутки

ComponentEvent – сокрытие, визуализация, перемещение, изменение размера компонента;

ContainerEvent – добавление или исключение компонента из контейнера;

FocusEvent – компонент получает или теряет фокус клавиатурного ввода;

InputEvent – абстр. класс для всех классов ввода компонентов

ItemEvent – щелчок на флажке или элементе спика, отметка пункта меню;

MouseWheelEvent – прокрутка колесика мыши;

TextEvent – изменение значения текстовой области или поля.

Иерархия:

ComponentEvent → InputEvent → KeyEvent и MouseEvent

Источники событий:

Кнопка Флажок Переключатель Список Пункт меню Линейка прокрутки Текстовые компоненты Окно

Интерфейсы слушателей событий: (java.awt.event)

ActionListener - для обработки события ActionEvent
void actionPerformed(ActionEvent ae) – нажали кнопку …

MouseListener - для обработки события MouseEvent
void mouseClicked(MouseEvent me) - кликнули мышью;
void mouseEntered(MouseEvent me) - курсор мыши вошел в компонент;
void mouseExited(MouseEvent me) - курсор мыши покинул компонент;
void mousePressed(MouseEvent me) - нажата кнопка мыши;
void mouseReleased(MouseEvent me) - отпущена кнопка мыши.

MouseMotionListener - для обработки события MouseEvent
void mouseDragged(MouseEvent me) - перетащили мышь (с нажатием);
void mouseMoved(MouseEvent me) - мышь перемещена;

KeyListener – для обработки события KeyEvent
void keyPressed(KeyEvent ke) – клавиша нажата;
void keyReleased(KeyEvent ke) – клавиша отпущена;
void keyTyped(KeyEvent ke) – введен символ.

AdjustmentListener
WindowListener
TextListener и др.

Источник регистрирует слушателей с помощью метода

public void add*Type*Listener(*Type*Listener el)

addKeyListener(…) **addMouseListener(…)**
addMouseMotionListener(…) **addActionListener(…)**

Обработка событий мыши

```
import  java.awt.*;
import  java.awt.event.*;

public class MouseEvents extends Frame implements MouseListener,
MouseMotionListener{
  int  mX=0, mY=0;

  public MouseEvents(){              // окно будет источником и слушателем
        super(ms);
        setSize(800,500);
        addWindowListener(new WindowAdapter ( ) {
                public void windowClosing(WindowEvent e)   {
                        System.exit(0);  }  }  );
        addMouseListener(this);                 //this.addMouseListener(this)
        addMouseMotionListener(this);
        setVisible(true);  }

  public void mousePressed(MouseEvent e){    //нажали кнопку мыши
          mX=e.getX();
           mY=e.getY();     }        //получили координаты
```

```
public void mouseDragged(MouseEvent e){ //переместили мышь
        Graphics g = this.getGraphics();          // с нажатой кнопкой
        int x = e.getX();
        int y = e.getY();            //новые координаты
        g.drawLine(mX,mY,x,y);                    //нарисовали линию
        mX=x;
        mY=y;              }

public static void main(String args []) {
    new MouseEvents("Привет");
  }
public void mouseClicked(MouseEvent me) { }
public void mouseEntered(MouseEvent me) { }
public void mouseExited(MouseEvent me) { }
public void mouseReleased(MouseEvent me) { }
public void mouseMoved(MouseEvent me) { }
}
```

Обработка событий клавиатуры

```
import java.awt.*;
import java.awt.event.*;

public class KeyEvents extends Frame implements KeyListener{
        String msg = " ";
        int X=100, Y=100;

public KeyEvents(String ms){
      super(ms);
      setSize(800,500);
      addWindowListener(new WindowAdapter ( ) {
          public void windowClosing(WindowEvent e) {
                                System.exit(0);      } } );
     addKeyListener(this);    //подключаем слушателя
     setVisible(true);  }

public void keyReleased(KeyEvent ke){     //при отпускании клавиши
              setTitle("Key Up");   }

public void keyTyped(KeyEvent ke){     // ввод символа
        msg+=ke.getKeyChar();
        repaint();    }
```

```
public void paint(Graphics g){
        g.drawString(msg,X,Y);  }

public void keyPressed(KeyEvent ke){   //при нажатии клавиши
    setTitle ("Key Down");
    int key = ke.getKeyCode(); //код клавиши

    switch(key){
        case KeyEvent.VK_LEFT: //нажата клавиша «влево»
                msg+="left";
                break;
        case KeyEvent.VK_RIGHT: //нажата клавиша «вправо»
                msg+="right";
                break;
        case KeyEvent.VK_F1:  //нажата клавиша «F1»
                msg+="F1 !!!!!!";
                 break;       }
     repaint();  }

public static void main(String args []) {
          new KeyEvents("Привет");     }
}
```


Key Up
43 qesfgbfg vF1 !!!!!!F1 !!!!!!F1 !!!!!!leftleftleftleftleftrightrightrightrightrightbbnvbnxcvbnbvleftleftleftrightrightF1 !!!!!!F1 !!!!!!

Классы адаптеры – обеспечивают пустую реализацию всех методов в интерфейсах слушателей событий. **MouseMotionAdapter** для **MouseMotionListener, MouseAdapter, KeyAdapter** и др.

```
import  java.awt.event.*;
import  java.awt.*;

public class Adapter extends Frame{
    public Adapter( ) {      //слушатель – объект класса MyAdapter
        super();
        setSize(800,500);
        addMouseListener(new MyAdapter(this));
        addWindowListener(new WindowAdapter (){
            public void windowClosing(WindowEvent e)    {
                                    System.exit(0);   }   }   );
         setVisible(true);   }

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

```
class MyAdapter extends MouseAdapter{
        Adapter adp;
 public MyAdapter(Adapter A) {
                    adp=A;  }

 public void mouseClicked(MouseEvent me){
        adp.setTitle("Mouse clicked");    }
}
```

Анонимные внутренние классы – классы без имени.

```
import  java.awt.event.*;
import  java.awt.*;

public class NoName extends Frame{

public NoName( ){
    super();
    setSize(800,500);
    addMouseListener( new MouseAdapter( ) {
            public void mousePressed(MouseEvent me) {
                        setTitle("Mouse pressed");   } } );
    setVisible(true);
}

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

Элементы управления

Текстовые метки Кнопки Флажки Списки
Списки с выбором Полосы прокрутки Текстовые поля
Текстовые области

Для добавления и удаления компонента в окно или апплет методы класса Container:

Component add(Component comp) – добавить;
void remove(Component comp) - удалить;
void removeAll() – удалить все.

1. **Label** – метка

Конструкторы:
Label() **Label(String str)** **Label(String str, int how)**
str – текст,
how = Label.LEFT или **Label.RIGHT** или **Label.CENTER**
void setText(String text) – установить текст в метке;
String getText() – получить текст метки.

2. Button – кнопка
Содержит текстовую метку и генерирует событие ActionEvent при нажатии.

Конструкторы: **Button()** **Button(String text)**

Методы:
void setLabel(String text) – установить название кнопки;
String getLabel() – получить название кнопки.

3. TextField – текстовое поле
Конструкторы:
TextField() **TextField (int num)**
TextField(String str) **TextField(String str, int num)**

str – строка, кот. будет отображаться в текст. поле;
num – количество символов – ширина текст. поля.
String getText() **void setText(String str)**
void setEchoChar(char ch) – символ для отображения (например, «*» для пароля)

```
import java.awt.*;    import java.awt.event.*;      //кнопки с обработкой событий

public class ButtonEvents extends Frame implements   ActionListener{
    Button   yes,    no;

 public ButtonEvents(String ms) {
     super(ms);          setLayout(null);         this.setSize(800,500);
     yes=new Button("YES");          no=new Button("NO");     //создаем кнопки
     yes.setSize(40,20);                  no.setSize(40,20);
     yes.setVisible(true);                 no.setVisible(true);
     yes.setLocation(100,100);         no.setLocation(100,150);
     this.add(yes);                         this.add(no);
    yes.addActionListener(this);    no.addActionListener(this); //кнопка - источник
    this.setVisible(true);     }

 public void actionPerformed(ActionEvent e){
        Graphics g = this.getGraphics();
        String text=e.getActionCommand();     //название кнопки
        if(text.equals("YES"))     g.drawString(text,90 ,90);
        else                                 g.drawString(text,90 ,180);      }

  public static void main(String args []) {
                                      new ButtonEvents("Привет");     }     }
```


Привет
YES
NO

Пример окна с обработкой событий мыши ClickMe

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;

public class ClickMe extends Frame implements   Runnable,
MouseListener {
    Thread t;          // поток для перемещения лягушки
    int   x,  y;
    BufferedImage img, img2;   //изображения лягушки и взрыва
    int period = 3000;
    int points = 0;
    int imgNumber = 0;    // номер картинки для отображения
```

```
public ClickMe(String ms)  {    // конструктор окна
   super(ms);
   setSize(800,500);
   setBackground (Color.green);
   addMouseListener(this);
   try{                              //загрузка изображений
         img  = ImageIO.read(new File("img0.jpg"));
         img2 = ImageIO.read(new File("img1.jpg"));}
         catch(IOException e){  }
         t = new Thread(this);
         t.start();
   setVisible(true);    }

public void paint ( Graphics  g)   {  //перерисовка
  if (imgNumber == 0)   g.drawImage(img, x - 50, y - 50, this);
  else                  g.drawImage(img2, x - 50, y - 50, this);
  g.drawString("Points:"+points,400,400);    }
```

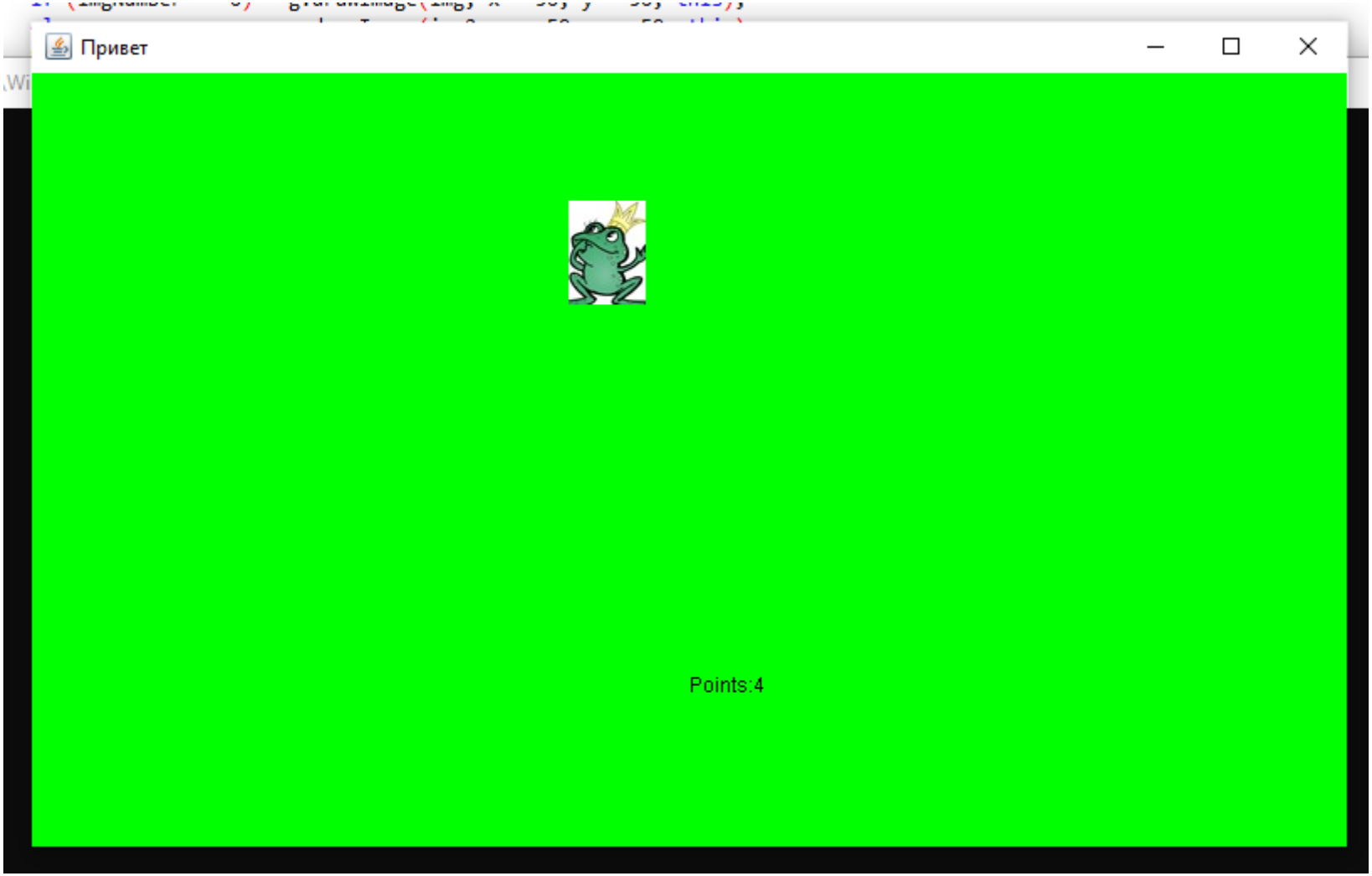
```
public void run( ) {   // действия потока t
      while (true)  {
            try {
                  x = 50 + (int) (Math.random() * 400);
                   y = 50 + (int) (Math.random() * 400);
                   repaint();
                   t.sleep(period);   }
               catch (InterruptedException e) {  }
              }   }

public void   mousePressed ( MouseEvent   me)   {
   if ( (x - 50) < me.getX()  &&  me.getX() < (x + 50)
          &&  (y - 50) < me.getY() &&  me.getY() < (y + 50) )    {
                points++;
                period/=1.1;
                imgNumber = 1;
                repaint();
              }       }
```

```
public void   mouseReleased ( MouseEvent   me)   {
        imgNumber = 0;
        repaint();
    }

public void   mouseClicked  ( MouseEvent   me) {   }
public void   mouseEntered ( MouseEvent   me) {   }
public void   mouseExited ( MouseEvent   me) {    }


public static void main(String args []) {
                                new ClickMe("Привет");     }
}
```


Привет
Points:4

```
public void   mouseClicked ( MouseEve
public void   mouseEntered
public void   mouseExited (

public static void main(Stri
                        ne
}
```

Привет

Points:9

27

28

28

Щелкните, чтобы добавить заметки