Классы-обертки

int – Integer

float – Float

double – Double

long – Long

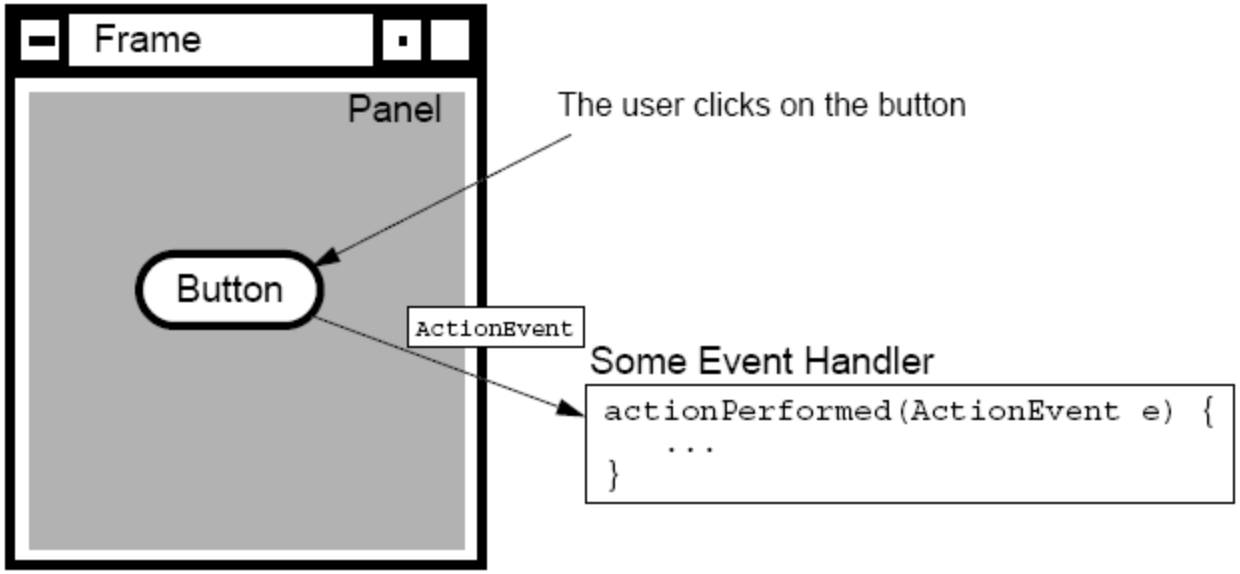
+ примитивные типы внедряются в парадигму ООП, появляются полезные поля и методы.

Например, Integer.parseInt(s) – возвращает значение int при преводе строки s

Обработка событий в AWT

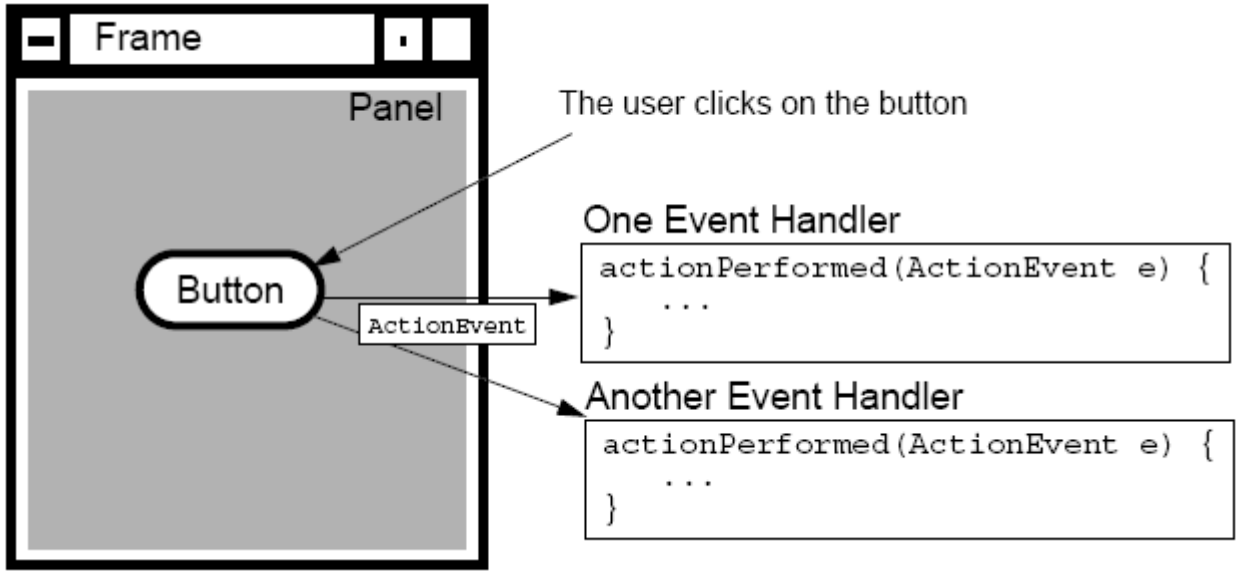
Событие

- Событие – объект, описывающий, что произошло
- Источник – объект, сгенерировавший событие
- Обработчик события – метод, получающий объект «событие» и реагирующий на него


Frame
Panel
Button
The user clicks on the button
ActionEvent
Some Event Handler
actionPerformed(ActionEvent e) {
...
}

Модель делегирования

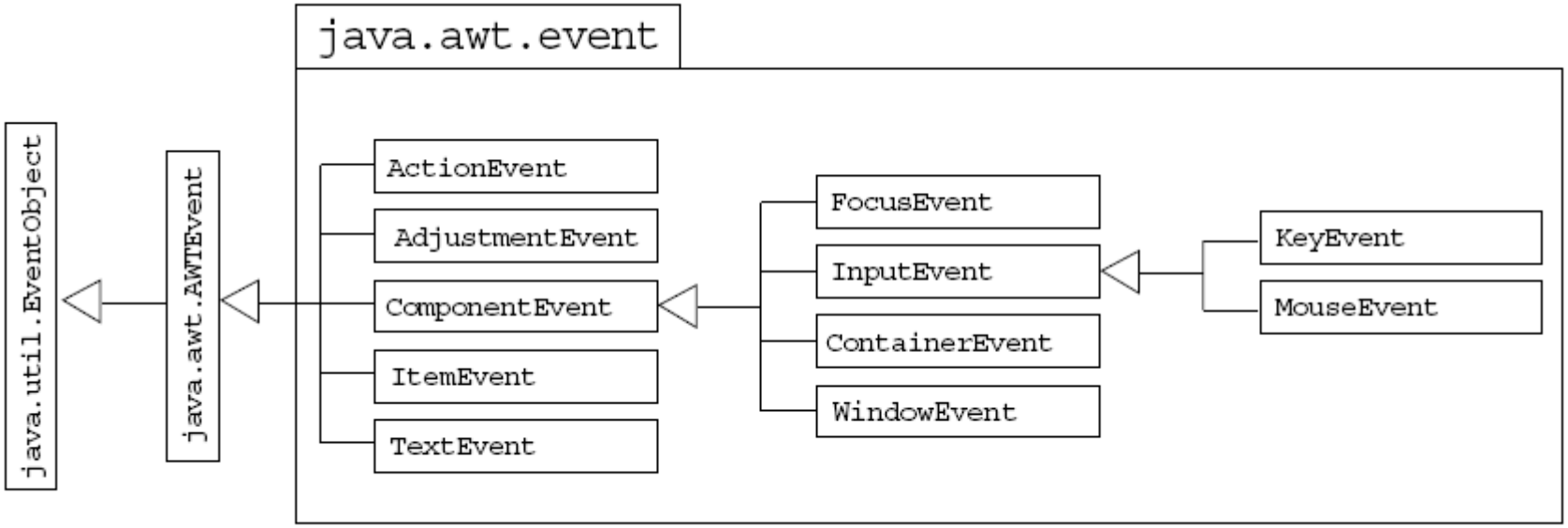
- Событие может посылаться нескольким обработчикам



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

public class TestButton {

    private Frame f;
    private Button b;

    public TestButton() {
        f = new Frame("Test");
        b = new Button("Press Me!");
        b.setActionCommand("ButtonPressed");
    }

    public void launchFrame() {
        b.addActionListener(new ButtonHandler());
        f.add(b, BorderLayout.CENTER);
        f.pack();
        f.setVisible(true);
    }

    public static void main(String args[]) {
        TestButton guiApp = new TestButton();
        guiApp.launchFrame();
    }
}
```

```
class ButtonHandler implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        System.out.println("Action occurred");
        System.out.println("Button's command is: "
                + e.getActionCommand());
    }
}
```

Категории событий



Категория	Интерфейс	Методы
Action	ActionListener	actionPerformed(ActionEvent)
Item	ItemListener	itemStateChanged(ItemEvent)
Mouse	MouseListener	mousePressed(MouseEvent) mouseReleased(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mouseClicked(MouseEvent)
Mouse motion	MouseMotionListener	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
Key	KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
Focus	FocusListener	focusGained(FocusEvent) focusLost(FocusEvent)

Категория	Интерфейс	Методы
Component	ComponentListener	componentMoved(ComponentEvent) componentHidden(ComponentEvent) componentResized(ComponentEvent) componentShown(ComponentEvent)
Window	WindowListener	windowClosing(WindowEvent) windowOpened(WindowEvent) windowIconified(WindowEvent) windowDeiconified(WindowEvent) windowClosed(WindowEvent) windowActivated(WindowEvent) windowDeactivated(WindowEvent)
Container	ContainerListener	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
Text	TextListener	textValueChanged(TextEvent)

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

public class TwoListener implements MouseMotionListener, MouseListener {
    private Frame f;
    private TextField tf;

    public TwoListener() {
        f = new Frame("Two listeners example");
        tf = new TextField(30);
    }

    public void launchFrame() {
        Label label = new Label("Click and drag the mouse");
        f.add(label, BorderLayout.NORTH);
        f.add(tf, BorderLayout.SOUTH);
        f.addMouseMotionListener(this);
        f.addMouseListener(this);
        f.setSize(300, 200);
        f.setVisible(true);
    }
```

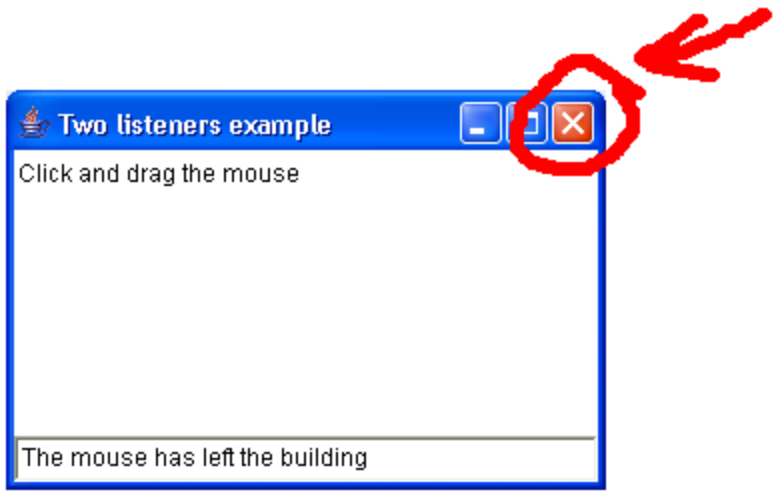
```
public void mouseDragged(MouseEvent e) {
    String s = "Mouse dragging: X = " + e.getX() + " Y = " + e.getY();
    tf.setText(s);
}

public void mouseEntered(MouseEvent e) {
    String s = "The mouse entered";
    tf.setText(s);
}

public void mouseExited(MouseEvent e) {
    String s = "The mouse has left the building";
    tf.setText(s);
}
```

```
    public void mouseMoved(MouseEvent e) {
    }

    public void mousePressed(MouseEvent e) {
    }

    public void mouseClicked(MouseEvent e) {
    }

    public void mouseReleased(MouseEvent e) {
    }

    public static void main(String args[]) {
        TwoListener two = new TwoListener();
        two.launchFrame();
    }

}
```

Задание

Добавить к написанной программе возможность закрытия окна

Использовать метод *dispose()* для окна или System.exit(0);



Адаптеры

- Адаптер – это класс-слушатель, реализующий все методы интерфейса в виде пустых заглушек
- Создайте наследника этого класса и переопределите только те методы, которые Вам нужны

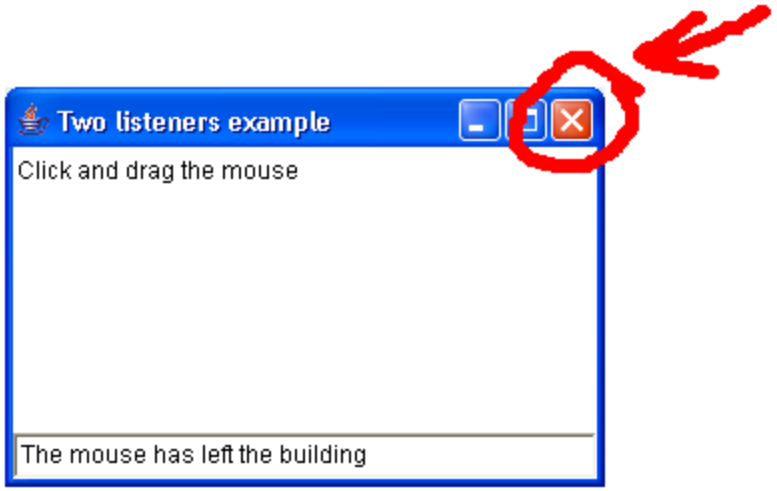
```
import java.awt.*;
import java.awt.event.*;

public class MouseClickHandler extends MouseAdapter {

    public void mouseClicked(MouseEvent e) {

        //...

    }

}
```

Задание

Добавить к написанной ранее программе возможность закрытия окна

Использовать метод *dispose()*

Использовать *WindowAdapter*



Обработка событий

- В основном классе
- Во внешнем классе
- Во вложенном классе
- В анонимном классе

Обработка событий в том же классе

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

public class FrameClass implements ActionListener {
    int counter;

    Frame f = new Frame();
    Label label = new Label("" + counter);
    Button incButton = new Button("+");
    Button decButton = new Button("-");

    public FrameClass() {
        incButton.addActionListener(this);
        decButton.addActionListener(this);
        f.add(incButton, BorderLayout.EAST);
        f.add(decButton, BorderLayout.WEST);
        f.add(label, BorderLayout.CENTER);
        f.pack();
        f.setVisible(true);
    }
```

Обработка событий в том же классе (продолжение)

```
public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("+")) {
            label.setText("" + ++counter);
        } else if (e.getActionCommand().equals("-")) {
            label.setText("" + --counter);
        }

    }

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

- Один обработчик для одного типа событий
- Чем больше компонентов, тем более громоздкий обработчик

Обработка событий во внешних классах

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

public class FrameClass {
    int counter;
    Frame f = new Frame();
    Label label = new Label("" + counter);
    Button incButton = new Button("+");
    Button decButton = new Button("-");

    public FrameClass() {
        incButton.addActionListener(new IncActionListener(this));
        decButton.addActionListener(new DecActionListener(this));
        f.add(incButton, BorderLayout.EAST);
        f.add(decButton, BorderLayout.WEST);
        f.add(label, BorderLayout.CENTER);
        f.pack();
        f.setVisible(true);
    }

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

Обработка событий во внешних классах (продолжение)

```
class IncActionListener implements ActionListener {
    private FrameClass fm;

    public IncActionListener(FrameClass fm) {
        this.fm = fm;
    }

    public void actionPerformed(ActionEvent e) {
        fm.label.setText("" + ++fm.counter);
    }
}

class DecActionListener implements ActionListener {
    private FrameClass fm;

    public DecActionListener(FrameClass fm) {
        this.fm = fm;
    }

    public void actionPerformed(ActionEvent e) {
        fm.label.setText("" + --fm.counter);
    }
}
```

- Требуется много вспомогательного кода для организации доступа к полям и методам объекта основного класса

Обработка событий во вложенных классах

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

public class FrameClass {
    int counter;
    Frame f = new Frame();
    Label label = new Label("" + counter);
    Button incButton = new Button("+");
    Button decButton = new Button("-");

    public FrameClass() {
        incButton.addActionListener(new IncActionListener());
        decButton.addActionListener(new DecActionListener());
        f.add(incButton, BorderLayout.EAST);
        f.add(decButton, BorderLayout.WEST);
        f.add(label, BorderLayout.CENTER);
        f.pack();
        f.setVisible(true);
    }

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

Обработка событий во вложенных классах (продолжение)

```
    class IncActionListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            label.setText("" + ++counter); // Доступ через неявную ссылку
        }
    } // Конец вложенного класса

    class DecActionListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            label.setText("" + --counter); // Доступ через неявную ссылку
        }
    } // Конец вложенного класса

} // Конец основного класса
```

- Вложенный класс имеет неявную ссылку на внешний объект
- Вспомогательного кода для хранения ссылки не требуется

Обработка событий в анонимных классах

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

public class FrameClass {
    int counter;
    Frame f = new Frame();
    Label label = new Label("" + counter);
    Button incButton = new Button("+");
    Button decButton = new Button("-");

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

Обработка событий в анонимных классах (продолжение)

```
    // Конструктор
    public FrameClass() {
        incButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                label.setText("" + ++counter);
            }
        });
        decButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                label.setText("" + --counter);
            }
        });
        f.add(incButton, BorderLayout.EAST);
        f.add(decButton, BorderLayout.WEST);
        f.add(label, BorderLayout.CENTER);
        f.pack();
        f.setVisible(true);
    }

} // Конец основного класса
```

- Анонимный класс имеет неявную ссылку на внешний объект
- Вспомогательного кода для хранения ссылки не требуется
- Имеет смысл использовать для коротких и простых обработчиков событий

	Act	Adj	Cmp	Cnt	Foc	Itm	Key	Mous	MM	Text	Win
Button	✓		✓		✓		✓	✓	✓		
Canvas			✓		✓		✓	✓	✓		
Checkbox			✓		✓	✓	✓	✓	✓		
CheckboxMenu Item						✓					
Choice			✓		✓	✓	✓	✓	✓		
Component			✓		✓		✓	✓	✓		
Container			✓	✓	✓		✓	✓	✓		
Dialog			✓	✓	✓		✓	✓	✓		✓
Frame			✓	✓	✓		✓	✓	✓		✓
Label			✓		✓		✓	✓	✓		
List	✓		✓		✓	✓	✓	✓	✓		
MenuItem	✓										
Panel			✓	✓	✓		✓	✓	✓		
ScrollBar			✓		✓		✓	✓	✓		
ScrollPane		✓	✓	✓	✓		✓	✓	✓		
TextArea			✓		✓		✓	✓	✓	✓	
TextField	✓		✓		✓		✓	✓	✓	✓	
Window			✓	✓	✓		✓	✓	✓		✓

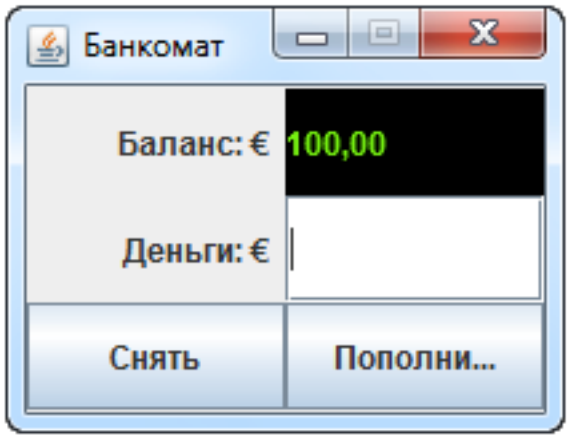
Задание

1. Написать программу, которая вызывает графический пользовательский интерфейс с такими функциями:

-слева находятся панель и кнопка, после нажатия на которую панель меняет свой цвет на случайный;

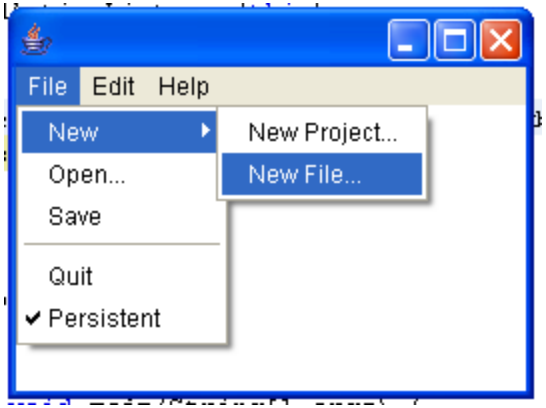
-справа находятся текстовое поле с 0 и две кнопки, при нажатии на которые значение в поле или увеличивается или уменьшается.

2. Реализовать работу банкомата — снятие и пополнение денег.



Работа с меню

1. Создать MenuBar и добавить его во Frame с предыдущего задания с помощью setMenuBar
2. Создать один или несколько Menu и добавить их в MenuBar
3. Создать один или несколько MenuItem и добавить их в Menu



Задание

Создать окно входа в систему. Окно должно содержать:

- текстовые поля для ввода логина и пароля с подписями (введенный пароль не должен быть виден);
- кнопки "Войти", "Регистрация" и "Отмена".

Окно должно соответствовать стандартам расположении элементов управления.

При выборе "Отмена" работа приложения завершается.

При выборе "Войти" выполняется проверка правильности логина и пароля и выдается сообщение о результате проверки.

При выборе "Регистрация" окно переходит в режим регистрации нового пользователя (изменяется заголовок окна, появляется новое поле для подтверждения пароля, скрывается кнопка "Войти").

Для хранения логинов и паролей в памяти использовать в многомерных массивов.