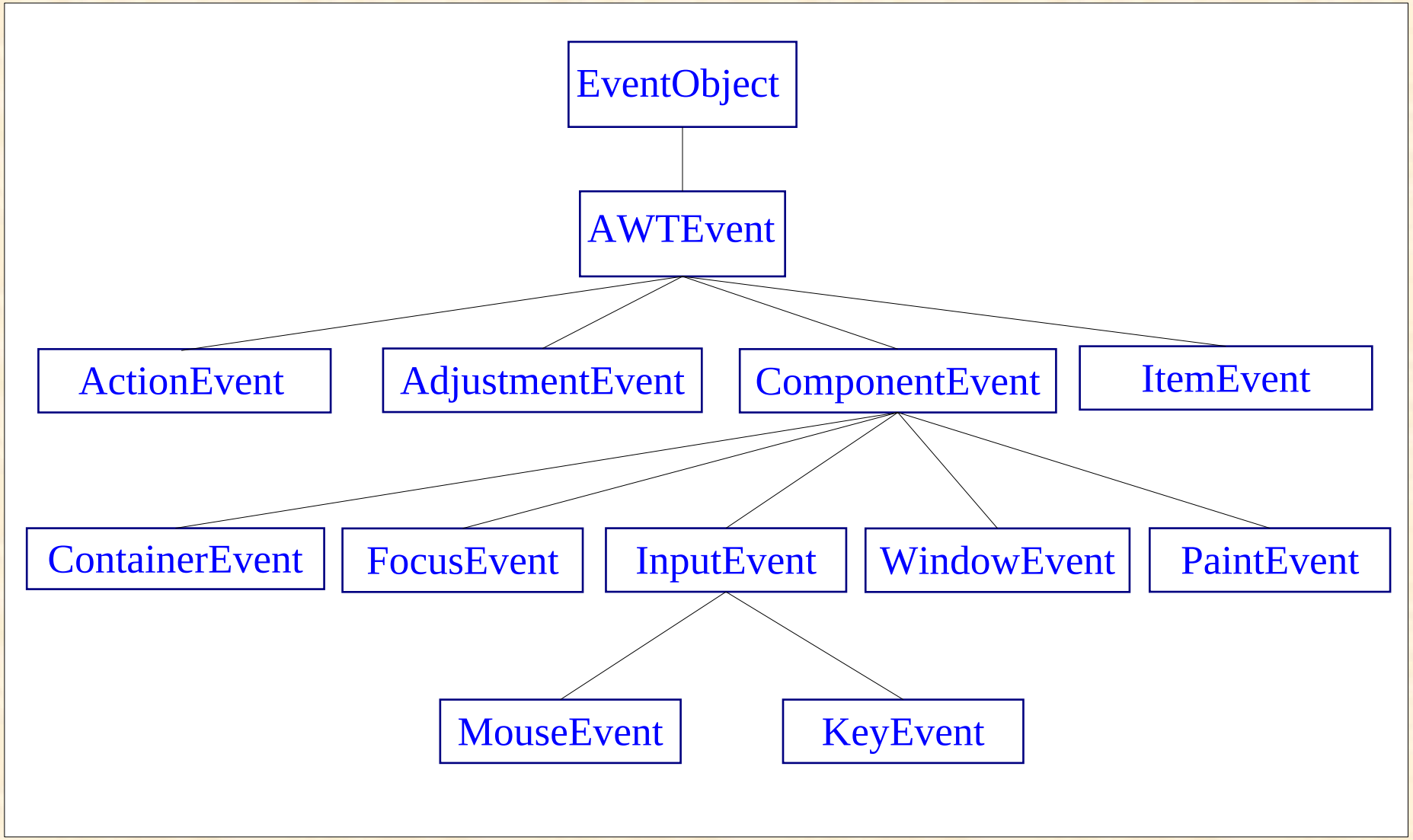
Ereignisbehandlung in Java

Das Ereignismodell des JDK 1.1

- Aktivitäten in der Benutzerschnittstelle lösen sogenannte Ereignisse aus
- Gruppe von Ereignissen wird durch Objekte einer bestimmten Klasse repräsentiert
- Ereignisverarbeitung wird in Form eines Delegationsmodells durchgeführt
  - Ereignisempfänger müssen sich bei der Ereignisquelle (z.B. Button) registrieren lassen
  - beim Auslösen von Ereignissen werden für die Empfängerobjekte bestimmte Methoden aufgerufen

Klassenhierarchie der Ereignis-Klassen



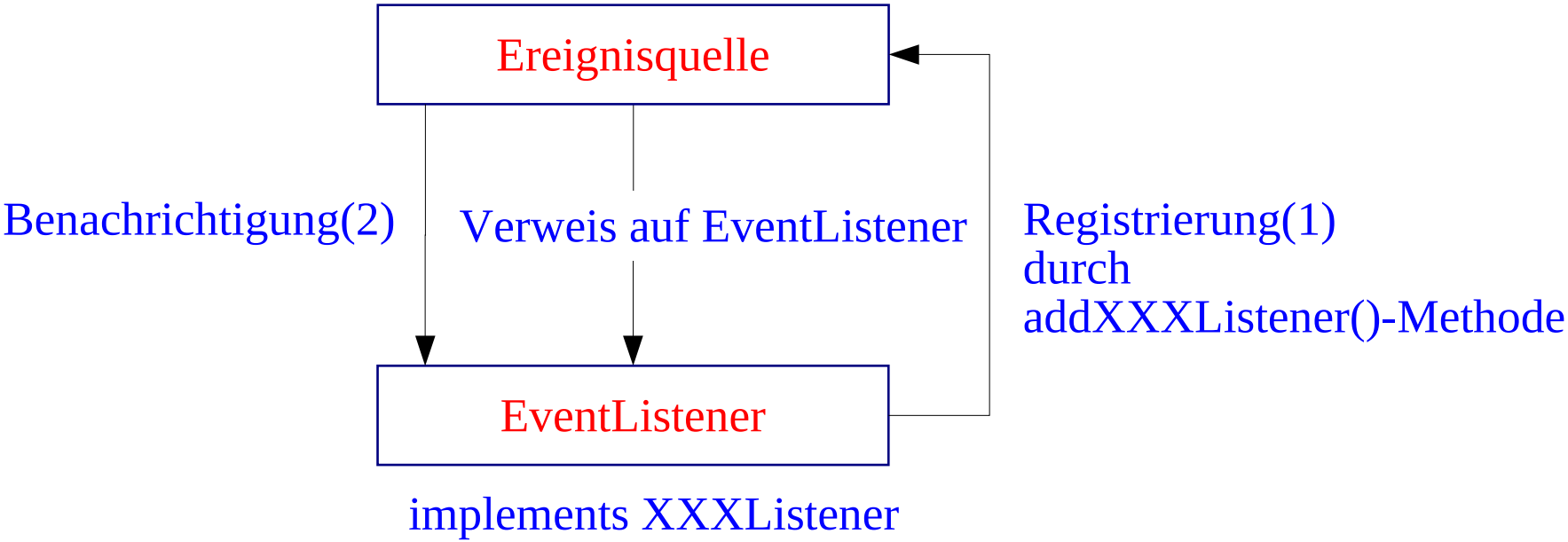
Methoden der Klasse AWTEvent

- getSource() - liefert einen Verweis auf die Ereignisquelle
- getID() - liefert die ID des Ereignisses

ActionEvent	ACTION_PERFORMED	Button,List
KeyEvent	KEY_PRESSED, KEY_RELEASED, KEY_TYPED	Component
MouseEvent	MOUSE_ENTERED, MOUSE_EXITED, MOUSE_PRESSED, MOUSE_RELEASE, MOUSE_MOVED, MOUSE_DRAGGED	Component
ItemEvent	SELECTED, DESELECTED, ITEM_STATE_CHANGED	List, Choice
AdjustmentEvent	TRACK, UNIT_INCREMENT, UNIT_DECREMENT, BLOCK_INCREMENT	Scrollbar

Delegationsmodell

- Ereignis wird durch drei Elemente repräsentiert
  - die Komponente, in der sich das Ereignis zugetragen hat
  - die Art des Ereignisses
  - die Komponenten, die über das Ereignis informiert werden wollen
- Zusammenspiel dieser Elemente



Registrierung

- Implementierung einer ActionListener-Schnittstelle

```
class MyListener implements ActionListener {
        void actionPerformed(ActionEvent e) { . . . }
}
```

- die Klasse wird instanziert

```
MyListener myListener = new MyListener();
```

- die Klasse wird bei der Ereignisquelle registriert

```
Button b = new Button("Cancel");
b.addActionListener(myListener);
```

Beispiel: Drücken eines Buttons

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

public class ActionTest implements ActionListener {
    private Button b;
    public static void main(String [] args) { new ActionTest().startTest(); }
    void startTest(){
            Frame frame = new Frame();  frame.setSize(200,200); frame.setVisible(true);
            frame.setLayout(new FlowLayout());
            b = new Button("Java");
            b.addActionListener(this);
            frame.add(b);
    }
    public void actionPerformed(ActionEvent e) {
            System.out.println("Action in Button Java");
    }
}
```

Beispiel: Abfangen von Mouse-Clicks

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


public class MouseListenerDemo implements MouseListener {

    public static void main(String [] args) {
       new MouseListenerDemo().startTest();
    }

    void startTest(){
       Frame frame = new Frame();
       frame.setSize(300,200);
       frame.setVisible(true);
       frame.setBackground(Color.blue);
       frame.addMouseListener(this);
    }

  ...
```

Beispiel: Abfangen von Mouse-Clicks

```
...

public void mouseClicked(MouseEvent e) {
   System.out.println("click at x:"+e.getX()+" y:"+e.getY());
}

public void mousePressed(MouseEvent e) {}

public void mouseReleased(MouseEvent e) {}

public void mouseEntered(MouseEvent e) {}

public void mouseExited(MouseEvent e) {}

}
```

Die Klasse ActionEvent

- Ein Action-Event wird ausgelöst, wenn in einer Komponente eine Aktion geschieht
  - Button wird gedrückt
  - Eingabe in einem Textfeld wird durch Return-Taste abgeschlossen
  - Auswahl eines Elements aus einer Liste, Choice oder Menü
- Bearbeitung über Implementierung des Interface ActionListener
  - public void actionPerformed(ActionEvent)
- Methoden der Klasse ActionEvent
  - String getActionCommand() - liefert das Kommando zu dieser Aktion
  - int getModifier() - liefert Modifier-Tasten (Shift, Control, Meta), die während der Aktion gedrückt wurden

Beispiel: Ereignisse in unterschiedlichen Button

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

public class ActionEventDemo implements ActionListener {
   public static void main(String [] args) { new ActionEventDemo().startTest(); }
   void startTest() {
      Frame frame = new Frame(); frame.setSize(300,200); frame.setVisible(true);
      frame.setLayout(new FlowLayout());
      Button open = new Button("open");      // Button erzeugen
      Button close = new Button("close");
      frame.add(open);                       // Button positionieren
      frame.add(close);
      open.addActionListener(this);          // Registrierung
      close.addActionListener(this);
      open.setActionCommand("open");         // Aktionskommandos setzen
      close.setActionCommand("close");
   }
```

Beispiel: Ereignisse in unterschiedlichen Button

```
        ...

    public void actionPerformed(ActionEvent e) {
        if ("open".equals(e.getActionCommand()))
                        System.out.println("action in open");
        if ("close".equals(e.getActionCommand()))
                    System.out.println("action in close");
    }
}
```

Beispiel: Ereignisse in unterschiedlichen Textfeldern

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

public class TextFieldTest implements ActionListener {
   TextField topField, bottomField; Label messageLabel;

   public static void main(String [] args) { new TextFieldTest().startTest(); }

   void startTest() {
      Frame frame = new Frame();  frame.setSize(300,200); frame.setVisible(true);

      topField = new TextField();              // Komponenten erstellen
      bottomField = new TextField();
      messageLabel = new Label();
      messageLabel.setAlignment(Label.CENTER);

      frame.add("North", topField);            // Anordnen der Komponenten
      frame.add("South", bottomField);
      frame.add("Center", messageLabel);

      topField.addActionListener(this);        // Registrierung
      bottomField.addActionListener(this);
   }

...
```

Beispiel: Ereignisse in unterschiedlichen Textfeldern

```
   ...

   public void actionPerformed(ActionEvent e)
   {
      if (e.getSource() == topField){            // Eingabe in TopField

         messageLabel.setText("Action in top text field: " + topField.getText());
      }

      if (e.getSource() == bottomField){       // Eingabe in BottomField

         messageLabel.setText("Action in bottom text field: " + bottomField.getText());
      }
   }
}
```

Die Klasse ItemEvent

- Item-Event wird ausgelöst, sobald der Benutzer eine Selektion in einem Listenfeld vornimmt
- Bearbeitung über Implementierung des Interface ItemListener
  - public void itemStateChanged(ItemEvent)
- Methoden der Klasse ItemEvent
  - Object getItem() - liefert den Eintrag in dem das Ereignis stattfand
  - int getStateChange() - liefert den Zustand des Eintrags, der durch das Ereignis geändert wurde (SELECTED, DESELECTED)
  - getItemSelectable - liefert einen Verweis auf die Komponente in der das Ereignis stattgefunden hat
- Doppelklick oder Return-Auswahl liefert ein ItemEvent und ActionEvent

Beispiel: Auswahl in einem Listenfeld

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

public class ListEventDemo implements ActionListener, ItemListener {
   private TextArea messages; private List myList;

   public static void main(String [] args) { new ListEventDemo().startTest(); }

   void startTest(){
      Frame frame = new Frame();              // Frame erzeugen
      frame.setSize(300,200); frame.setVisible(true);

      myList = new List(3,false);             // Listenfeld erstellen
      myList.add("Java"); myList.add("Coffee");
      myList.add("Espresso"); myList.add("Capuccino");

      messages = new TextArea();              // Textbereich erstellen

      myList.addActionListener(this);         // Registrierung
      myList.addItemListener(this);

      frame.add("North",myList);              // Positionierung
      frame.add("Center",messages);
   }
   ...
```

Beispiel: Auswahl in einem Listenfeld

```
    ...

    public void actionPerformed(ActionEvent e) {
      append("Action: " + e.getActionCommand());
    }

    public void itemStateChanged(ItemEvent e){
      append("Item selected: " + myList.getItem((Integer) e.getItem()));
    }

    public void append(String s) { messages.append(s + "\n"); }

}
```