# Java Swing Events

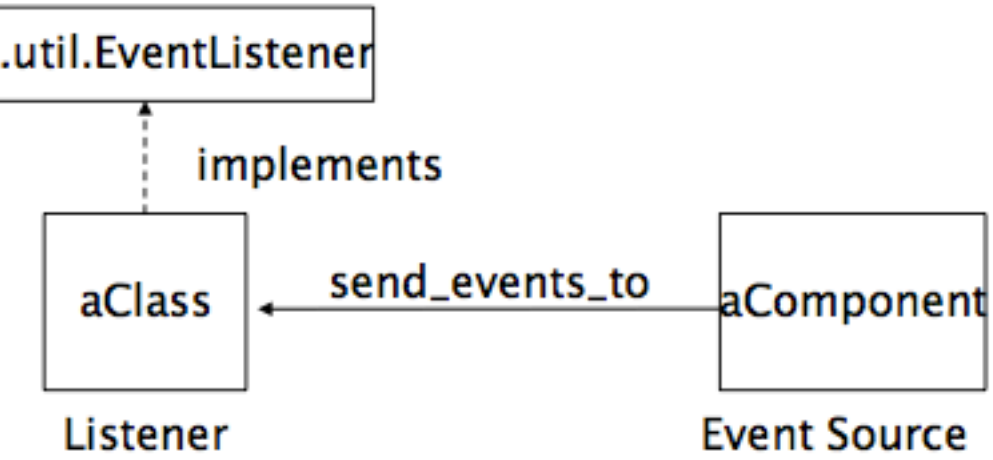
Università di Modena e Reggio Emilia
*Prof. Nicola Bicocchi (nicola.bicocchi@unimore.it)*

ING

# Event Delegation Model

- From Java 1.1
  - Events are classified by type (MouseEvent, KeyEvent, ActionEvent, …)
  - Events are generated in components (source)
  - Objects can be registered to components as listeners (target)
- Whenever an event occurs, the event thread send a message to all the registered listener objects (the event is passed as a parameter)
- A listener object must implement appropriate interface (to make possible the call-back)

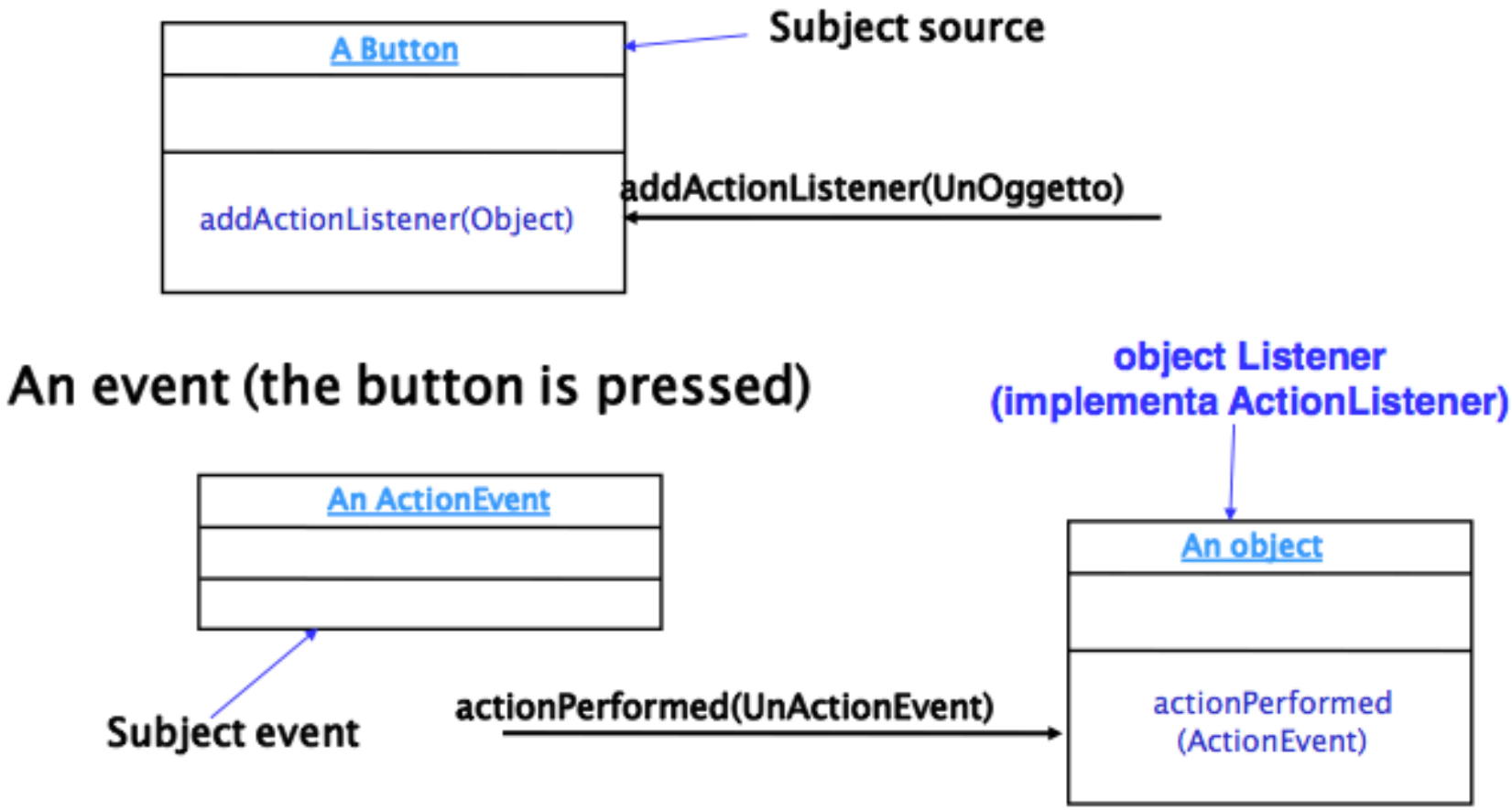
# Event Delegation Model




Multiple listeners can register to be notified of events of a particular type from a particular source. Also, the same listener can listen to notifications from different objects.


ING

# Example

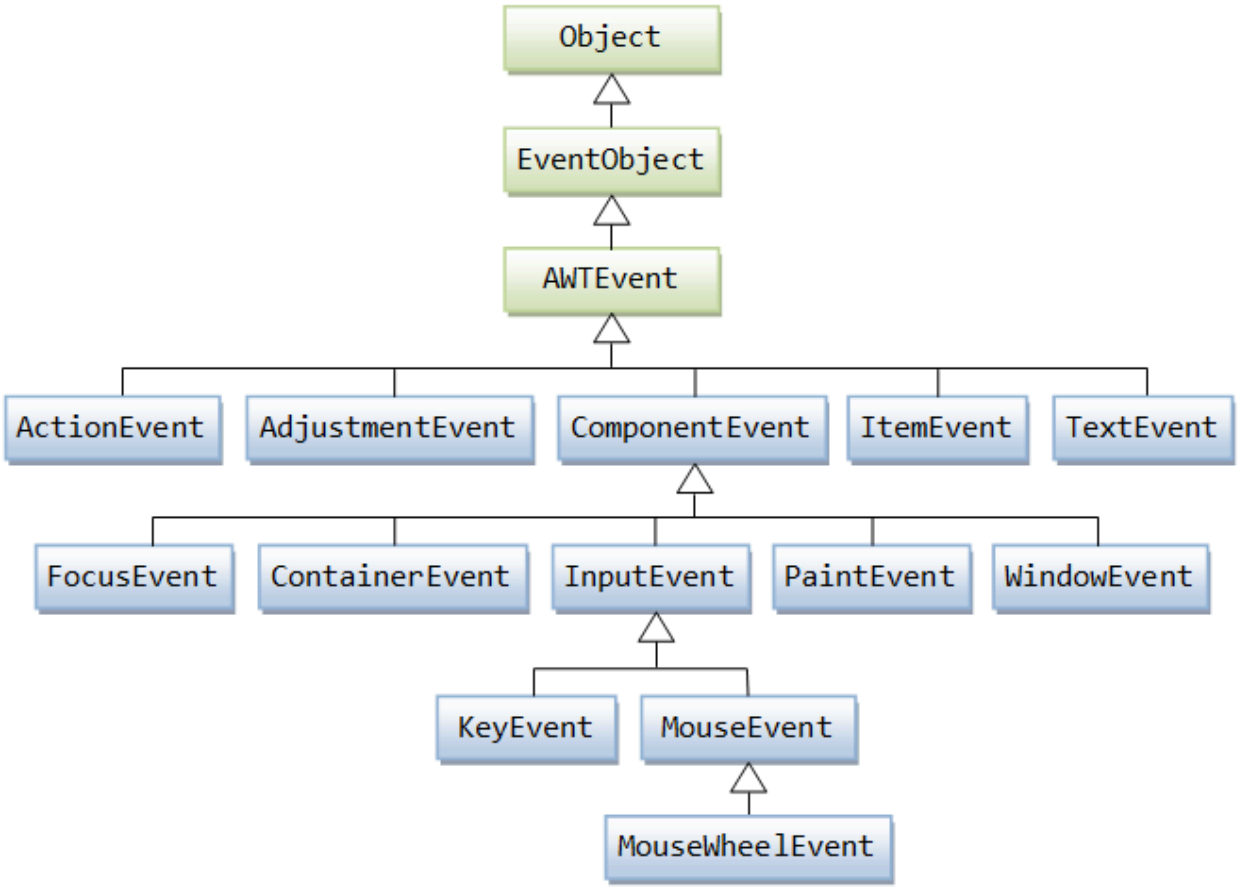

A Button
Subject source
addActionListener(Object)
addActionListener(UnOggetto)
An event (the button is pressed)
object Listener
(implementa ActionListener)
An ActionEvent
An object
Subject event
actionPerformed(UnActionEvent)
actionPerformed
(ActionEvent)

# Event Delegation Model

- Events are organized by type and need specific listeners

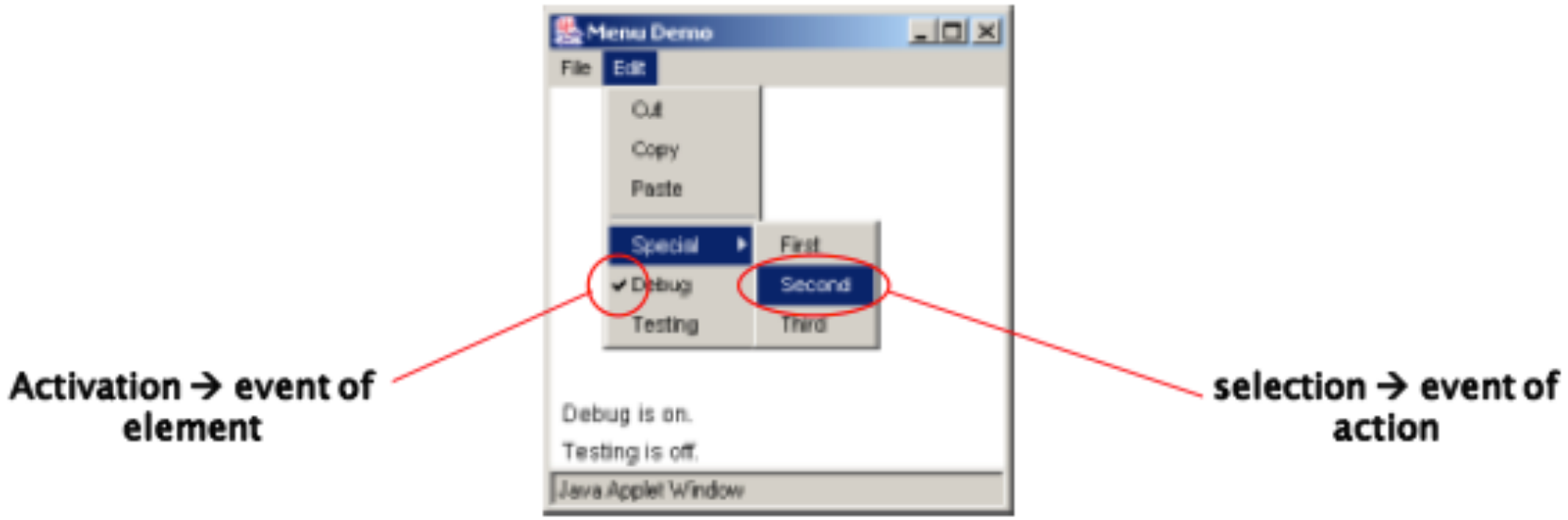
| User Action | Event Triggered | Event Listener interface |
|---|---|---|
| Click a Button, JButton | ActionEvent | ActionListener |
| Open, iconify, close Frame, JFrame | WindowEvent | WindowListener |
| Click a Component, JComponent | MouseEvent | MouseListener |
| Change texts in a TextField, JTextField | TextEvent | TextListener |
| Type a key | KeyEvent | KeyListener |
| Click/Select an item in a Choice, JCheckbox, JRadioButton, JComboBox | ItemEvent, ActionEvent | ItemListener, ActionListener |

# EventObject




ING

# “selection” and “activation”




ING

# A complete example

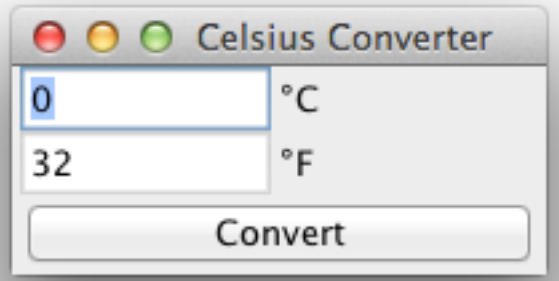
```
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;

public class CelsiusConverter extends JFrame implements ActionListener {
      private JButton convertButton;
      private JTextField fahrenheitLabel;
      private JTextField tempTextField;

      public CelsiusConverter() {
            super("Celsius Converter");
            tempTextField = new JTextField("0");
            fahrenheitLabel = new JTextField("32");
            fahrenheitLabel.setEditable(false);
            convertButton = new JButton("Convert");
            convertButton.addActionListener(this);

            JPanel p = new JPanel();
            p.setLayout(new GridLayout(2, 2));
            p.add(tempTextField); p.add(new JLabel("°C"));
            p.add(fahrenheitLabel); p.add(new JLabel("°F"));
```




# A complete example

```
        setLayout(new BorderLayout());
        add(p, BorderLayout.CENTER);
        add(convertButton, BorderLayout.SOUTH);

        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        setSize(200, 100);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        int tempFahr = (int)
            ((Double.parseDouble(tempTextField.getText())) * 1.8 + 32);
        fahrenheitLabel.setText(Integer.toString(tempFahr));
    }

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

# How to manage events in Java

- The principle underlying the events is quite similar to the exceptions :
  - the class declares which event is able to deal with (one or more) by implementing one or more interfaces
  - joins a component that are source of events (JButton, JTextField, etc..)
    - JButton.addActionListener(this)
- Pay attention!You're implementing interfaces, so you must overwrite all methods of those interfaces!

# How to manage events in Java

- Components can:
  1. Handle events on their own
  2. Delegate events to their parent class
  3. Delegate events to external classes

# Handle events on their own

```
class ButtonWithEvents extends JButton implements InterfaceWithEvents {
    addListener(this);

    void methodOfTheInterfaceWithEvents() {...}
    void anotehrMethodOfTheInterfaceWithEvents() {...}
} //end class
```

# Delegate events to their parent class

```
class FrameWithEvents extends JFrame implements InterfaceWithEvents {
    JComponent componentSourceofEvents = new JComponent();
    componentSourceOfEvents.addListener(this);

    void methodOfTheInterfaceWithEvents() {...}
    void anotehrMethodOfTheInterfaceWithEvents() {...}
} //end class
```

# Delegate events to external classes

```
class MyListener  implements InterfaceWithEvents {
    void methodOfTheInterfaceWithEvents() {...}
    void anotehrMethodOfTheInterfaceWithEvents() {...}
} //end class


class Frame extends JFrame {
    MyListener listener = new MyListener();
    JComponent componentSourceofEvents = new JComponent();
    componentSourceOfEvents.addListener(listener);
} //end class
```

# Dealing with multiple sources

- getSource() and object references
  - if (evt.getSource() == ButtonSelfDestruction ) {}
- getActionCommand() and custom strings
  - If (evt.getActionCommand() == “destroy”) {}
- Event classes
  - If (evt instanceof(KeyEvent)) {}

# Extracting information from events

- Every function that appears at the interfaces presents a common argument (KeyEvent, MouseEvent, etc.)
- Each argument is an object and it provides methods to get information about the event
- ActionListener
  - String getActionCommand(): returns a string identifying the component which generated the command
  - String paramString(): returns a string describing the event type (common to all event objects)

# Event Interfaces

- ActionListener
  - void actionPerformed (ActionEvent evt)
- FocusListener
  - void focusGained (FocusEvent evt)
  - void focusLost (FocusEvent evt)
- ItemListener
  - void itemStateChanged (ItemEvent evt)

# Event Interfaces

- MouseListener
  - void mouseClicked (MouseEvent evt)
  - void mouseEntered (MouseEvent evt)
  - void mouseExited (MouseEvent evt)
  - void mousePressed (MouseEvent evt)
  - void mouseReleased (MouseEvent evt)
- MouseMotionListener
  - void mouseDragged (MouseEvent evt)
  - void mouseMoved (MouseEvent evt)

# Event Interfaces

- KeyListener
  - void keyPressed (KeyEvent evt)
  - void keyReleased(KeyEvent evt)
  - void keyTyped(KeyEvent evt)
- WindowListener
  - void windowActivated(WindowEvent evt)
  - void windowClosed (WindowEvent evt)
  - void windowClosing (WindowEvent evt)
  - void windowDeactivated (WindowEvent evt)
  - void windowDeiconified (WindowEvent evt)
  - void windowIconified (WindowEvent evt)
  - void windowOpened (WindowEvent evt)