

## GUI in Java con l'AWT – 1

Stefano Mizzaro

Dipartimento di matematica e informatica  
Università di Udine  
<http://www.dimi.uniud.it/~mizzaro>  
mizzaro@dimi.uniud.it  
Programmazione, lezione 22  
5 maggio 2004

## Scaletta

- GUI
- GUI in Java: l'AWT
- Esempi
- Principi alla base
  - Componenti, eventi, gestori eventi (ascoltatori)
  - Classi interne
  - Layout
- Da fare e da non fare
- Le API dell'AWT

Stefano Mizzaro - AWT1

2

## Cos'è una GUI

- Graphical User Interface
  - Interfaccia Utente Grafica
- Interazione utente-programma basata su
  - Linea di comando (shell Unix)
    - Solo testo, tastiera, cursore
  - GUI (Mozilla, MSWord, XEmacs, ...)
    - Grafica, finestre, pulsanti, puntatore/mouse, menu
- Finora solo linea di comando...
- Come si fanno le GUI in Java?

Stefano Mizzaro - AWT1

3

## GUI in Java

- Tutto è un oggetto
  - Finestre, pulsanti, ... (componenti)
  - Azioni dell'utente! (eventi)
  - Gestori degli eventi
- AWT (Abstract Window Toolkit): classi predefinite (`java.awt.*`, `java.awt.event.*`, ...)
- AWT: GUI indipendenti dalla piattaforma (lo stesso codice funziona su varie piattaforme HW/SW)

Stefano Mizzaro - AWT1

4

## Un esempio

- La nostra prima GUI: una finestra vuota che non fa nulla

```
import java.awt.*;
public class GUI1 extends Frame {
    public GUI1 () {
        super(); // (inutile... super("GUI1");)
        setBounds(100,100,300,200); //Dimensioni
        setTitle("GUI1"); //Titolo finestra
        setVisible(true); //Rende visibile
    }
}

```

- Classe di prova:

```
public class ProvaGUI1 {
    public static void main (String[] args) {
        GUI1 finestraPrincipale = new GUI1();
        System.out.println("Fine");
    }
}

```

>java ProvaGUI1

Stefano Mizzaro - AWT1

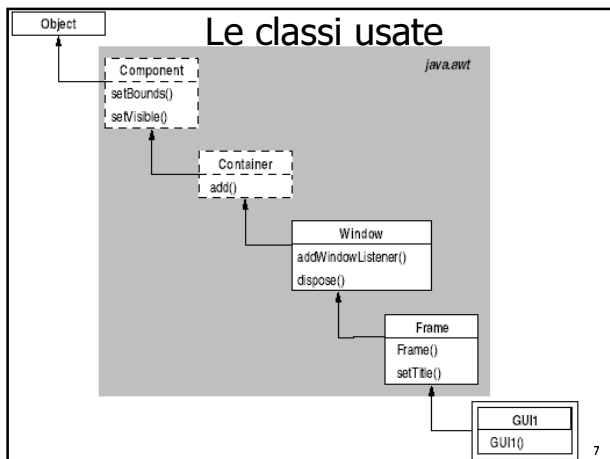
5

## Come funziona?

- In **GUI1** c'è solo il costruttore
- che viene chiamato nel main
- e, usando opportunamente i metodi delle sopraclassi, crea ("costruisce") la finestra
- Notate che "**Fine**" viene visualizzato subito...
- Potrei invocare i metodi anche dal main
- Cosa c'è in **Frame** e nelle superclassi?

Stefano Mizzaro - AWT1

6



## Gestione eventi

- Così com'è non fa nulla!
- Ci sono solo componenti, non gestori di eventi
- Modifichiamo la nostra GUI per gestire un evento (chiusura finestra)
- Click su "X" → evento → che codice eseguire?
- Invocazioni implicite: il programmatore
  - scrive il metodo, non la chiama
  - mette il metodo in un oggetto (ascoltatore)
  - "associa" l'oggetto all'azione

Stefano Mizzaro - AWT1

8

## Ascoltatore (1/3)

```

import java.awt.*;
public class GUI2 extends Frame {
    public GUI2 () {
        Ascoltatore asc = new Ascoltatore();
        setBounds(100,100,300,200);
        setTitle("GUI2");
        setVisible(true);
        addWindowListener(asc); //Aggiunge
        //l'ascoltatore
    }
}
  
```

```

this.addWindowListener(asc);
  
```

Stefano Mizzaro - AWT1

9

## Ascoltatore (2/3)

```

import java.awt.event.*;
public class Ascoltatore extends WindowAdapter
{
    public void windowClosed (WindowEvent e) {
        System.out.println(e);
        System.exit(0);
    }

    public void windowClosing (WindowEvent e) {
        e.getWindow().dispose();
        System.out.println(e);
    }
}
  
```

Stefano Mizzaro - AWT1

10

## Ascoltatore (3/3)

- (quasi nessuna modifica)

```

public class ProvaGUI2 {
    public static void main (String[] args) {
        GUI2 finestraPrincipale = new GUI2();
        System.out.println("Fine");
    }
}
  
```

```

>java ProvaGUI2
  
```

```

Frame
  
```

Stefano Mizzaro - AWT1

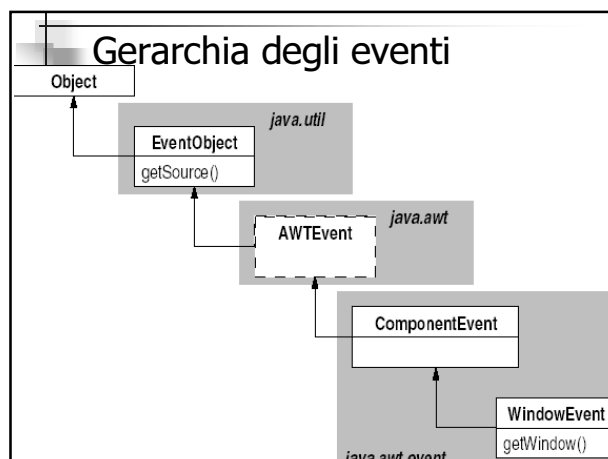
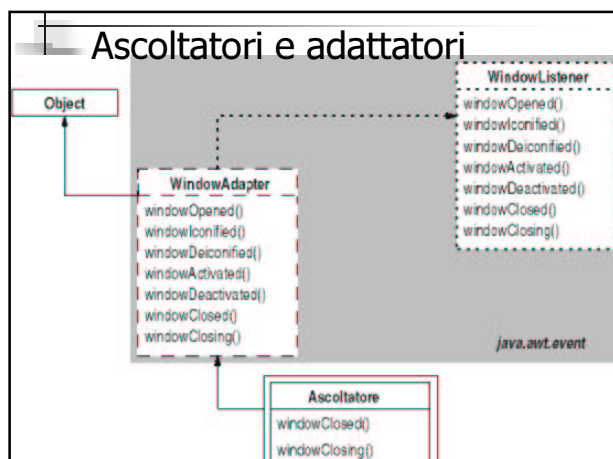
11

## Meccanismo di gestione eventi

- Azione utente →
- Creazione dell'istanza dell'evento opportuno (sottoclasse di **AWTEvent**) →
- Invocazione (automatica) dei metodi dell'ascoltatore (un'istanza di **EventListener**)
  - L'ascoltatore deve essersi registrato in precedenza per quegli eventi (**addWindowListener**)
  - L'istanza-evento viene passata come argomento al metodo invocato automaticamente

Stefano Mizzaro - AWT1

12



### Tutto è un oggetto!

- 3 istanze:
  - La GUI (**Frame**)
  - L'evento generato dal click del mouse sul "pulsante" di chiusura finestra (**WindowEvent**)
  - L'ascoltatore con il codice eseguito per gestire l'evento (**WindowListener**)
- Facciamo un diagramma unico!
- (4a istanza: il gestore della posizione degli oggetti che potrebbero esserci nella finestra)

Stefano Mizzaro - AWT1 15

### Senza l'adattatore...

```

import java.awt.event.*;
public class Ascoltatore
    implements WindowListener {
    public void windowOpened(WindowEvent e) {}
    public void windowIconified(WindowEvent e) {}
    public void windowDeiconified(WindowEvent e) {}
    public void windowActivated(WindowEvent e) {}
    public void windowDeactivated(WindowEvent e) {}
    public void windowClosed(WindowEvent e) {
        System.out.println(e);
        System.exit(0);
    }
    public void windowClosing(WindowEvent e) {
        e.getWindow().dispose();
        System.out.println(e);
    }
}
  
```

Stefano Mizzaro - AWT1 16

### Ascoltatori e classi interne

- Meglio estendere un Adapter che implementare un Listener
- Però c'è eredità singola...
- ... e poi è scomodo creare tante classi...
- ... e poi come accedo agli attributi...
- Soluzione: classi interne per gli ascoltatori
  - Membro ("dentro la classe") (usate questa!)
  - Locali ("dentro un metodo")
  - Anonime ("dentro un metodo e senza nome")

Stefano Mizzaro - AWT1 17

### Ascoltatore e classe membro

```

import java.awt.*;
import java.awt.event.*;
public class GUI2 extends Frame {
    class Ascoltatore extends WindowAdapter {
        // idem come prima
    }
    public GUI2 () {
        // idem come prima
    }
}
  
```

Stefano Mizzaro - AWT1 18

## Ascoltatore e classe locale

```
import java.awt.*;
import java.awt.event.*;
public class GUI2 extends Frame {
    public GUI2 () {
        class Ascoltatore extends WindowAdapter {
            // idem come prima
        }
        // idem come prima
    }
}
```

Stefano Mizzaro - AWT1

19

## Ascoltatore e classe anonima

```
import java.awt.*;
import java.awt.event.*;
public class GUI2 extends Frame {
    public GUI2 () {
        WindowAdapter asc = (new WindowAdapter() {
            // idem come prima
        });
        // idem come prima
    }
}

addWindowListener(new WindowAdapter() {
    // idem come prima
});
```

Stefano Mizzaro - AWT1

20

## Aggiungiamo qualcosa...

- Ok, ora l'interfaccia fa qualcosa, ma è vuota...
- Aggiungiamo:
  - un etichetta (Label)
  - un pulsante (Button)
- **add**, definito in **Container**

Stefano Mizzaro - AWT1

21

## Etichetta e pulsante (1/5)

```
import java.awt.*;
class GUI3 extends Frame {
    public GUI3 () {
        Ascoltatore asc = new Ascoltatore();
        setBounds(100,100,300,200);
        setTitle("GUI3");
        this.add(new Label("Premi il pulsante!"),
            BorderLayout.NORTH);
        Button b = new Button("Premimi!");
        Panel p = new Panel();
        p.add(b);
        this.add(p, BorderLayout.SOUTH);
        this.addWindowListener(asc);
        setVisible(true);
    }
}
```

Stefano Mizzaro - AWT1

22

## Etichetta e pulsante (2/5)

```
public class ProvaGUI3 {
    public static void main (String[] args) {
        GUI3 finestraPrincipale = new GUI3();
    }
}
```

```
>java ProvaGUI3
```

- Eh, il pulsante non fa nulla...
- ... perché manca l'ascoltatore!
- Click su pulsante → **ActionEvent**

Stefano Mizzaro - AWT1

23

## Etichetta e pulsante (3/5)

```
import java.awt.event.*;
class AscoltatorePulsante
    implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        System.out.println(
            "Hai premuto il pulsante!");
        System.out.println(e);
    }
}
```

Stefano Mizzaro - AWT1

24

## Etichetta e pulsante (4/5)

```
import java.awt.*;
class GUI4 extends Frame {
    public GUI4 () {
        Ascoltatore asc = new Ascoltatore();
        AscoltatorePulsante ap = new
            AscoltatorePulsante();
        setBounds(100,100,300,200);
        setTitle("GUI4");
        this.add(new Label("Premi il pulsante!",
            BorderLayout.NORTH);
        Button b = new Button("Premimi!");
        Panel p = new Panel();
        p.add(b);
        this.add(p, BorderLayout.SOUTH);
        this.addWindowListener(asc);
        b.addActionListener(ap);
        setVisible(true);
    }
}
```

## Etichetta e pulsante (5/5)

```
public class ProvaGUI4 {
    public static void main (String[] args) {
        GUI4 finestraPrincipale = new GUI4();
    }
}
```

```
>java ProvaGUI4
```

Stefano Mizzaro - AWT1

26

## Layout (cenni)

- Come disporre gli oggetti dentro un **Container** ("contenitore")
- **LayoutManager**: gestore di layout
- Ogni sottoclasse di **Container** ha associata un'istanza di **LayoutManager**
- Gestore di layout di default (**BorderLayout** per **Frame**)
- Ne riparleremo...

Stefano Mizzaro - AWT1

27

## Da fare e da non fare

- Finestra con due bottoni
- Classe "**MyFrame**" che
  - Estende **Frame**
  - Implementa uno o più listener
- Si può fare in due modi...
  - Male
  - Bene

Stefano Mizzaro - AWT1

28

## MalFatto.java (1/3)

```
import java.awt.*;
import java.awt.event.*;
public class MalFatto extends Frame
    implements ActionListener, WindowListener{
    Button b1 = new Button("Pulsante 1");
    b2 = new Button("Pulsante 2");
    public MalFatto() {
        ...
        add(b1); add(b2);
        b1.addActionListener(this);
        b2.addActionListener(this);
        addWindowListener(this);
    }
}
```

Stefano Mizzaro - AWT1

29

## MalFatto.java (2/3)

```
public void actionPerformed(ActionEvent e) {
    Object source = e.getSource();
    if (source == b1)
        System.out.println("Premuto Pulsante 1");
    else if (source == b2)
        System.out.println("Premuto Pulsante 2");
    else
        System.out.println("Qualcos'altro...");
}
```

Stefano Mizzaro - AWT1

30

## MalFatto.java (3/3)

```
public void windowClosing(WindowEvent e) {
    System.out.println("Window Closing");
    System.exit(0);
}
public void windowClosed(WindowEvent e) {}
public void windowDeiconified(WindowEvent e) {}
public void windowIconified(WindowEvent e) {}
public void windowActivated(WindowEvent e) {}
public void windowDeactivated(WindowEvent e) {}
public void windowOpened(WindowEvent e) {}

public static void main(String[] args) {
    Frame finestra = new MalFatto();
}
}
```

Stefano Mizzaro - AWT1

31

## Perché è fatto male

- Classe che fa 2 cose
  - Costruisce la GUI
  - Gestisce gli eventi
- Tanti metodi vuoti inutili
- Catene di if/else
- N.B. Purtroppo molti libri usano questa tecnica...☹

Stefano Mizzaro - AWT1

32

## BenFatto.java (1/2)

```
import java.awt.*;
import java.awt.event.*;
public class BenFatto extends Frame {
    public BenFatto() {
        Button b1 = new Button("Pulsante 1"),
            b2 = new Button("Pulsante 2");
        add(b1); add(b2);
        ...
        b1.addActionListener(new
            AscoltatorePulsante1());
        b2.addActionListener(new
            AscoltatorePulsante2());
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.out.println("Window Closing");
                System.exit(0);
            }
        });
    }
}
```

Stefano Mizzaro - AWT1

35

## BenFatto.java (2/2)

```
public class AscoltatorePulsante1
    implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Premuto Pulsante 1");
    }
}
public class AscoltatorePulsante2
    implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Premuto Pulsante 2");
    }
}
public static void main(String[] args) {
    Frame finestra = new BenFatto();
}
}
```

Stefano Mizzaro - AWT1

34

## Perché è fatto bene

- Ascoltatori con classi interne
- Evito codice inutile
- Ogni classe fa un'unica cosa
- Evito catene di if/else, uso l'eredità
- Una classe, una responsabilità

Stefano Mizzaro - AWT1

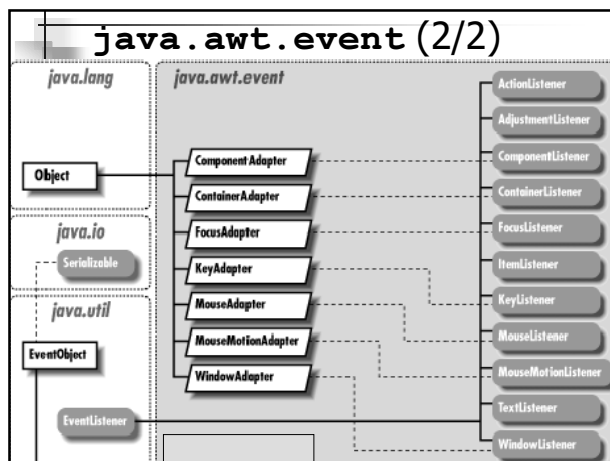
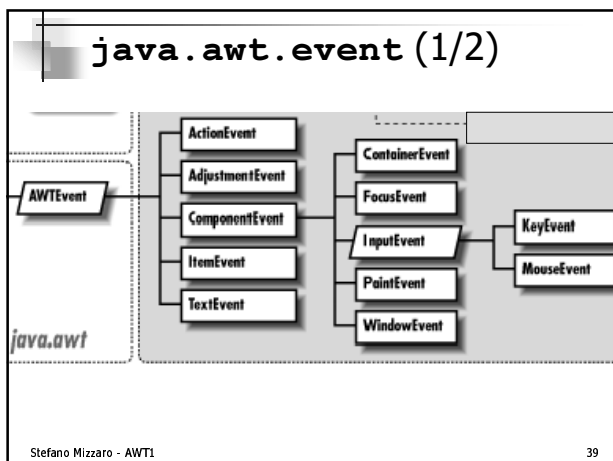
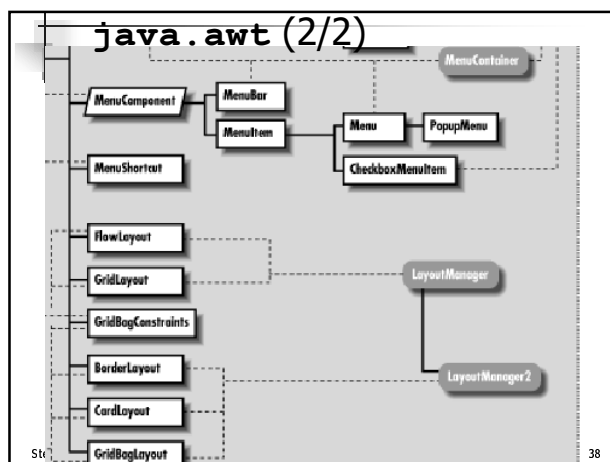
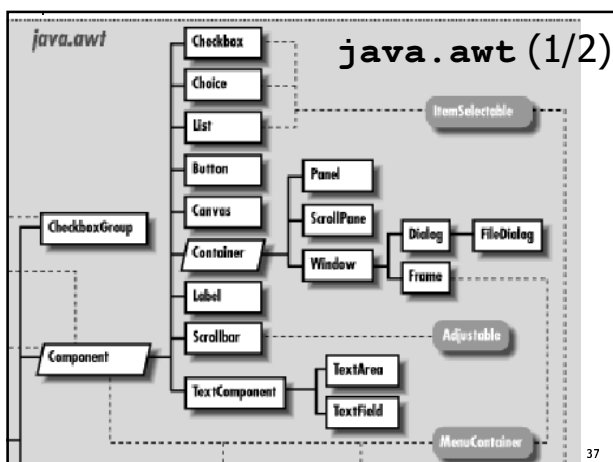
35

## Analisi API: 2 package

- **java.awt** (93 classi + 16 interfacce)
  - Contiene le classi per i componenti
  - (e altro: `Font`, `Color`, ...)
- **java.awt.event** (25 + 18)
  - Contiene le classi per gli eventi
    - Rappresentazione eventi
    - Gestione eventi

Stefano Mizzaro - AWT1

36



## Riassunto

- GUI
- GUI in Java: l'AWT
- Esempi
- Principi alla base
  - Componenti, eventi, gestori eventi (ascoltatori)
    - Classi interne
    - Layout e layout manager (cenni)
- Da fare e da non fare
- Le API dell'AWT

## Riferimenti

- Libri & lucidi
- Javadoc API
- Fotocopie
- Tutorial SUN (vecchio):
  - <http://java.sun.com/docs/books/tutorial/information/download.html>
  - <ftp://ftp.javasoft.com/docs/tut-OLDui.zip>
  - <http://java.sun.com/docs/books/tutorial/download/tut-OLDui.zip>