9
Interfaccia AWT

9.1 Introduzione

AWT – Abstract Windowing Tool   (javax.swing.*)

$$\begin{tikzcd}
 & & & \text{Component} \arrow[dlll] \arrow[dl] \arrow[drr] \arrow[drrr] & & & \\
\text{Canvas} & & \text{Container} \arrow[dl] \arrow[dr] & & & \text{Button} & \text{TextComponent} \arrow[d] \\
 & \text{Panel} \arrow[d] & & \text{Window} \arrow[dl] \arrow[dr] & & & \text{TextField} \\
 & \text{Applet} & \text{Frame} & & \text{Dialog} & &
\end{tikzcd}$$

Metodi comuni ai componenti:

void setFont(font);	Font getFont();
void setSize(width,height);	Dimension getSize();
void setEnabled(boolean);	boolean isEnabled();
void setVisible(boolean);	boolean isVisible();
void setBounds(x,y,width,height);	Rectangle getBounds();
	Graphics getGraphics();

I componenti vanno inseriti in un pannello o in un applet (che comunque è un pannello) o un frame che li contiene

```
public void init() {
        Button b = new Button("OK");
        this.add(b);
        }
```

9.2 Label

Una label è un testo non editabile

Viene scritta automaticamente (senza bisogno di usare paint())

Costruttori:

```
Label();
Label(String);
Label(String, int);
        Label.LEFT      0
        Label.CENTER    1
        Label.RIGHT     2
```

Metodi:

```
        void setFont(Font); (richiamato sulla label o sul contenitore)
        void setText(String);       String getText();
        void setAlignment(int);     int getAlignment();
```

Esempio

```
import java.awt.*;

public class LabelTest extends java.applet.Applet {

  public void init() {
    setFont(new Font ("Helvetica", Font.BOLD, 14));
    setLayout(new GridLayout(3,1));
    add(new Label("aligned left", Label.LEFT));
    add(new Label("aligned center", Label.CENTER));
    add(new Label("aligned right", Label.RIGHT));
  }
}
```

9.3 Button

Un bottone è un componente che quando viene premuto può innescare una azione

Costruttori:

Button();
Button(String);

Metodi:
	void setLabel(String);	String getLabel();

Esempio

```
import java.awt.*;

public class ButtonTest extends java.applet.Applet {

  public void init() {
  //  setLayout(new FlowLayout());
Button rewind = new Button("Rewind");
          add(rewind);
    add(new Button("Play"));
    add(new Button("Fast Forward"));
    add(new Button("Stop"));
  }
}
```

9.4 Check box

Una check box è un componente che può essere selezionato o deselezionato

Costruttori:

CheckBox();
CheckBox(String);
CheckBox(String,boolean);   (sezionata se true)

Metodi:

    void setLabel(String);   String getLabel();
    void setState(boolean);   boolean getState();

Esempio

```
import java.awt.*;

public class CheckboxTest extends java.applet.Applet {

  public void init() {
    setLayout(new FlowLayout(FlowLayout.LEFT));
    add(new Checkbox("Shoes"));
    add(new Checkbox("Socks"));
    add(new Checkbox("Pants"));
    add(new Checkbox("Underwear", true));
    add(new Checkbox("Shirt"));
  }
}
```

9.5 Radio buttons

I radiobuttons rappresentano un insieme di scelte esclusive

Costruzione:

```
CheckboxGroup grb = new CheckboxGroup(); // definisce l’insieme delle scelte

add(new CheckBox(“Yes”,true,grb); //si inseriscono le singole scelte
add(new CheckBox(“No”,false,grb);
```

Metodi:

```
        quelli per le singole Checkbox
//applicabili alle singole Checkbox
        void setCheckboxGroup(CheckboxGroup);
        CheckboxGroup getCheckboxGroup();

//applicabili ai CheckboxGroup
        void setCurrent(Checkbox);
        Checkbox getCurrent();
```

Esempio

```
import java.awt.*;

public class CheckboxGroupTest extends java.applet.Applet {

  public void init() {
    setLayout(new FlowLayout(FlowLayout.LEFT));
    CheckboxGroup cbg = new CheckboxGroup();

    add(new Checkbox("Red", false, cbg));
    add(new Checkbox("Blue", false, cbg));
    add(new Checkbox("Yellow", false, cbg));
    add(new Checkbox("Green", true, cbg));
    add(new Checkbox("Orange", false, cbg));
    add(new Checkbox("Purple", false, cbg));
  }
}
```

9.6 Menù di scelta

Sono liste a discesa da cui si può selezionare un elemento

Costruzione:

```
Choice c = new Choice();

c.add(“rosso”);
c.add(“verde”);
c.add(“blu”);
```

Metodi:

```
        String getItem(int);                //si inizia a contare da 0
        int getItemCount();
        int getSelectedIndex();
        String getSelectedItem();
        void select(int);
        void select(String);
```

Esempio

```
import java.awt.*;

public class ChoiceTest extends java.applet.Applet {

  public void init() {
    Choice c = new Choice();

    c.add("Apples");
    c.add("Oranges");
    c.add("Strawberries");
    c.add("Blueberries");
    c.add("Bananas");

    add(c);
  }
}
```

9.7 Campi di testo

Un text field fornisce la possibilità di editare una linea di testo

Costruttori:

```
TextField();
TextField(String);
TextField(String,int);
```

Metodi:

```
        void  setText(String);              String getText();
                                            int getColumns();
        void setEditable(boolean);          boolean getEditable();
        void setEchoChar(char);             char getEchoChar();
        boolean echoCharIsSet();
```

```
import java.awt.*;

public class TextFieldTest extends java.applet.Applet {

  public void init() {
    setLayout(new GridLayout(3,2,5,15));

    add(new Label("Enter your name:"));
    add(new TextField("your name here",45));
    add(new Label("Enter your phone number:"));
    add(new TextField(12));
    add(new Label("Enter your password:"));

    TextField t = new TextField(20);
    t.setEchoChar('*');
    add(t);

  }
}
```

9.8 Disposizione dei componenti

Il gestore di layout determina dinamicamente la disposizione dei componenti all'interno di un contenitore

La disposizione può anche essere fissata esplicitamente:

```
setLayout(null);
Button b = new Button("Rewind");
add(b);
b.setBounds(10,200,100,100);
```

FlowLayout dispone i componenti da sinistra a destra, in righe

```
SetLayout(new FlowLayout());
SetLayout(new FlowLayout(FlowLayout.LEFT));
SetLayout(new FlowLayout(FlowLayout.RIGHT,10,20));
```

GridLayout dispone i componenti in righe e colonne

```
SetLayout(new GridLayout(3,4));
SetLayout(new GridLayout(3,4,10,20));
```

BorderLayout:

```
import java.awt.*;
public class BorderLayoutTest extends java.applet.Applet {

public void init() {
setLayout(new BorderLayout());
add("North", new Button("One"));
add("East", new Button("Two"));
add("South", new Button("Three"));
add("West", new Button("Four"));
add("Center", new Button("Five"));
  }
}
```

CardLayout permette di visualizzare un componente alla volta:

```
import java.awt.*;

public class CardTest extends java.applet.Applet {
CardLayout cl =new CardLayout();
Label l1 = new Label("one");
Label l2 = new Label("two");
Label l3 = new Label("three");

public void init() {

  setLayout(cl);
  Panel one = new Panel();        add("first",one);          one.add(l1);
  Panel two = new Panel();        add("second",two);         two.add(l2);
  Panel three = new Panel();      add("third",three);        three.add(l3);
}


  public void start() {

  cl.show(this,"second");
  cl.previous(this);          cl.next(this);
  cl.first(this);             cl.last(this);
    }
}
```

Il GriBagLayout è il modo più flessibile per gestire la disposizione dei componenti

- definizione dell'oggetto GridBagLayout
- definizione del gestore di layout
- definizione di un oggetto di classe GridBagConstraints
- definizione dei vincoli per ogni componente
- aggiunta dei vincoli del componente al gestore di layout
- aggiunta del componente al pannello

primo passo: disegnare l'interfaccia

secondo passo: definizione griglia

```
import java.awt.*;

public class GridBagTestStep1 extends java.applet.Applet {

  void buildConstraints(GridBagConstraints gbc, int gx, int gy, int gw, int gh,
        int wx, int wy) {
gbc.gridx = gx;
gbc.gridy = gy;
/* posizione dalla cella che contiene il componente */
gbc.gridwidth = gw;
gbc.gridheight = gh;
/* ampiezza del componente in celle */
gbc.weightx = wx;
gbc.weighty = wy;
/* proporzioni delle celle */
  }

  public void init() {
        GridBagLayout gridbag = new GridBagLayout();
        GridBagConstraints constraints = new GridBagConstraints();
        setLayout(gridbag);

        constraints.fill = GridBagConstraints.BOTH;
/* ogni componente riempe la cella che lo contiene */

/*per semplicità inizialmente si definiscono tutti i componenti come bottoni */
        // Name label
        buildConstraints(constraints, 0, 0, 1, 1, 100, 100);
        Button label1 = new Button("Name:");
        gridbag.setConstraints(label1, constraints);
        add(label1);

        // Name text field
        buildConstraints(constraints, 1, 0, 1, 1, 100, 100);
        Button tfname = new Button();
        gridbag.setConstraints(tfname, constraints);
        add(tfname);


        // password label
        buildConstraints(constraints, 0, 1, 1, 1, 100, 100);
        Button label2 = new Button("Password:");
        gridbag.setConstraints(label2, constraints);
        add(label2);

        // password text field
        buildConstraints(constraints, 1, 1, 1, 1, 100, 100);
        Button tfpass = new Button();
        gridbag.setConstraints(tfpass, constraints);
        add(tfpass);

        // OK Button
        buildConstraints(constraints, 0, 2, 2, 1, 100, 100);
        Button okb = new Button("OK");
        gridbag.setConstraints(okb, constraints);
        add(okb);
  }
}
```

terzo passo: fissare le proporzioni

```
        // Name label
        buildConstraints(constraints, 0, 0, 1, 1, 10, 40);
        // Name text field
        buildConstraints(constraints, 1, 0, 1, 1, 90, 0);
        // password label
        buildConstraints(constraints, 0, 1, 1, 1, 0, 40);
        // password text field
        buildConstraints(constraints, 1, 1, 1, 1, 0, 0);
        // OK Button
        buildConstraints(constraints, 0, 2, 2, 1, 0, 20);
/* il valore 0 vuol dire fai come definito altrove*/
```

quarto passo: aggiungere i componenti

quinto passo: sistemare il tutto

```
constraints.fill:       //riempimento della cella
          GridBagConstraints.NONE;              //minime dimensioni del componente
          GridBagConstraints.BOTH;              //riempie completamente
          GridBagConstraints.HORIZONTAL;        //riempie orizzontalmente
          GridBagConstraints.VERTICAL;          //riempie verticalmente

constraints.anchor:          //posizione nella cella
         GridBagConstraints.CENTER;
         GridBagConstraints.NORTH;
         GridBagConstraints.NORTHEAST;
         . . .
         GridBagConstraints.SOUTH;
```

Esempio

```
import java.awt.*;
public class GridBagTestFinal extends java.applet.Applet {

void buildConstraints(GridBagConstraints gbc, int gx, int gy, int gw, int gh,
     int wx, int wy) {
gbc.gridx = gx;            gbc.gridy = gy;            gbc.gridwidth = gw;
gbc.gridheight = gh;       gbc.weightx = wx;          gbc.weighty = wy;
  }

public void init() {
GridBagLayout gridbag = new GridBagLayout();
GridBagConstraints constraints = new GridBagConstraints();
setLayout(gridbag);

// Name label
buildConstraints(constraints, 0, 0, 1, 1, 10, 40);
constraints.fill = GridBagConstraints.NONE;
constraints.anchor = GridBagConstraints.EAST;
Label label1 = new Label("Name:", Label.LEFT);
gridbag.setConstraints(label1, constraints);
add(label1);
// Name text field
buildConstraints(constraints, 1, 0, 1, 1, 90, 0);
constraints.fill = GridBagConstraints.HORIZONTAL;
TextField tfname = new TextField();
gridbag.setConstraints(tfname, constraints);          add(tfname);

// password label
buildConstraints(constraints, 0, 1, 1, 1, 0, 40);
constraints.fill = GridBagConstraints.NONE;
constraints.anchor = GridBagConstraints.EAST;
Label label2 = new Label("Password:", Label.LEFT);
gridbag.setConstraints(label2, constraints);          add(label2);

// password text field
buildConstraints(constraints, 1, 1, 1, 1, 0, 0);
constraints.fill = GridBagConstraints.HORIZONTAL;
TextField tfpass = new TextField();
tfpass.setEchoCharacter('*');
gridbag.setConstraints(tfpass, constraints);          add(tfpass);

// OK Button
buildConstraints(constraints, 0, 2, 2, 1, 0, 20);
constraints.fill = GridBagConstraints.NONE;
constraints.anchor = GridBagConstraints.CENTER;
Button okb = new Button("OK");
gridbag.setConstraints(okb, constraints);             add(okb);
  }
}
```

vincoli per spazio intorno ad un componente:
ipadx
ipady

spazio fra i componenti ed i bordi del pannello che li contiene:

```
public Insets getInsets() {
return new Insets(10,30,10,30);
}
```

9.9 Focus

Tasti Tab e Shift-Tab

Ordine di successione definito dall'ordine degli add()

requestFocus();

9.10 Eventi

sensore ActionListener

evento	metodi
azione	public void actionPerformed(ActionEvent e)

sensore ItemListener

evento	metodi
lista selezionata o deselezionata	public void itemStateChanged(ItemEvent e)

sensore FocusListener

evento	metodi
fuoco acquisito	public void focusGained(FocusEvent e)
fuoco perso	public void focusLost(FocusEvent e)

sensore TextListener

evento	metodi
testo cambiato	public void textValueChanged(TextEvent e)

sensore ComponentListener

evento	metodi
componente aggiunto	public void componentAdded(ContainerEvent e)
componente eliminato	public void componentRemoved(ContainerEvent e)

addXXXListener();

Esempi

```
import java.awt.Graphics;
import java.awt.*;
public class EsempioMouse extends java.applet.Applet  {
Label xm = new Label("      ");
Label ym = new Label("      ");
Label l1 = new Label("x=");
Label l2 = new Label("y=");
EventoMouse em;

public void init () {
add(l1);       add(xm);     add(l2);       add(ym);

em = new EventoMouse(this);

addMouseMotionListener(em);
ym.addMouseMotionListener(em);
  }
}
```

```
 import java.awt.*;
 import java.awt.event.*;
 public class EventoMouse extends MouseMotionAdapter     {

 EsempioMouse pp;

 EventoMouse (EsempioMouse p)
         {
         pp = p;
                  }

public void mouseMoved(MouseEvent e){
 pp.xm.setText(String.valueOf(e.getX()));
 pp.ym.setText(String.valueOf(e.getY()));
}

}
```

```
import java.awt.*;

public class CardTest extends java.applet.Applet {
CardLayout cl =new CardLayout();
Label l1 = new Label("one");    Label l2 = new Label("two");
Label l3 = new Label("three");
Button avanti1 = new Button("Avanti");
Button avanti2 = new Button("Avanti");
Button avanti3 = new Button("Avanti");

Button indietro1 = new Button("Indietro");
Button indietro2 = new Button("Indietro");
Button indietro3 = new Button("Indietro");


public void init() {
eventoavanti vaiavanti;    eventoindietro vaiindietro;

vaiavanti = new eventoavanti(this);
avanti1.addActionListener(vaiavanti);avanti2.addActionListener(vaiavanti);
avanti3.addActionListener(vaiavanti);

vaiindietro = new eventoindietro(this);
indietro1.addActionListener(vaiindietro); indietro2.addActionListener(vaiindietro);
  indietro3.addActionListener(vaiindietro);

setLayout(cl);
Panel one = new Panel();        add("first",one);
one.add(l1);                    one.add(avanti1);          one.add(indietro1);
Panel two = new Panel();        add("second",two);
two.add(l2);                    two.add(avanti2);          two.add(indietro2);
Panel three = new Panel();      add("third",three);
three.add(l3);                  three.add(avanti3);        three.add(indietro3);
}

public void start() {cl.show(this,"second");   }

}
```

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

public class eventoavanti implements ActionListener {
CardTest programma;

 eventoavanti(CardTest p) {  programma = p;}

  public void actionPerformed(ActionEvent e) {
  programma.cl.next(programma);
  }
}

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

public class eventoindietro implements ActionListener {
 CardTest programma;

 eventoindietro(CardTest p) {  programma = p;}

public void actionPerformed(ActionEvent e) {
programma.cl.previous(programma);
  }
}
```

==

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

public class ButtonActionsTest2 extends java.applet.Applet
          implements  ActionListener{

Button redButton,blueButton,greenButton,whiteButton,blackButton;

public void init() {
setBackground(Color.white);
setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

redButton = new Button("Red");
redButton.addActionListener(this);
add(redButton);

blueButton = new Button("Blue");
blueButton.addActionListener(this);
add(blueButton);

greenButton = new Button("Green");
greenButton.addActionListener(this); add(greenButton);
whiteButton = new Button("White");
whiteButton.addActionListener(this); add(whiteButton);
blackButton = new Button("Black");
blackButton.addActionListener(this); add(blackButton);
  }

public void actionPerformed(ActionEvent e) {

Button bottonepremuto = (Button) e.getSource();
Color theColor= null;
if (bottonepremuto == redButton)
        theColor= Color.red;
else if (bottonepremuto == greenButton)
        theColor= Color.green;
else if (bottonepremuto == blueButton)
        theColor= Color.blue;
else if (bottonepremuto == whiteButton)
        theColor= Color.white;
else if (bottonepremuto == blackButton)
        theColor= Color.black;
setBackground(theColor);         repaint();
   }



}
```


```
import java.awt.*;

public class ButtonActionsTest extends java.applet.Applet {

Button redButton,blueButton,greenButton,whiteButton,blackButton;

public void init() {
setBackground(Color.white);
setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

HandleButton he;

redButton = new Button("Red");
he = new HandleButton(this, Color.red);
redButton.addActionListener(he);
add(redButton);

blueButton = new Button("Blue");      he = new HandleButton(this, Color.blue);
blueButton.addActionListener(he);     add(blueButton);

greenButton = new Button("Green");  he = new HandleButton(this, Color.green);
greenButton.addActionListener(he);   add(greenButton);
whiteButton = new Button("White");  he = new HandleButton(this, Color.white);
whiteButton.addActionListener(he);   add(whiteButton);
blackButton = new Button("Black");  he = new HandleButton(this, Color.black);
blackButton.addActionListener(he);   add(blackButton);
  }
}


import java.awt.*;
import java.awt.event.*;
public class HandleButton implements ActionListener {

Color theColor;
ButtonActionsTest theApp;

HandleButton(ButtonActionsTest a, Color c) {theApp = a;        theColor = c;   }

public void actionPerformed(ActionEvent e) {
        theApp.setBackground(theColor);
        theApp.repaint();
   }
}
```

9.11 Aree di testo

Un'area di testo contiene una larga area editabile

Costruttori:

```
TextArea();
TextArea(String);
TextArea(String,int,int); //righe e colonne
TextArea(String,int,int,int);
        TextArea.SCROLLBARS_BOTH;
        TextArea.SCROLLBARS_HORIZONTAL_ONLY;
        TextArea.SCROLLBARS_VERTICAL_ONLY;
        TextArea.SCROLLBARS_NONE;
```

Metodi:

```
        int  getColumns();        int  getRows();                void setText(String);
        void setColumns(int);     void  setRows(int);                 String getText();
        void insert(String,int);  void replace(String,int,int);
```

Esempio

```
import java.awt.*;
public class TextAreaTest extends java.applet.Applet {
public void init() {
String str = "Once upon a midnight dreary, while I pondered, weak and weary,\n" +
"Over many a quaint and curious volume of forgotten lore,\n" +
"While I nodded, nearly napping, suddenly there came a tapping,\n" +
"As of some one gently rapping, rapping at my chamber door.\n" +
"\"'Tis some visitor,\" I muttered, \"tapping at my chamber door-\n" +
"Only this, and nothing more.

add(new TextArea(str));
  }
}
```

9.12 Scrollig list

Permette selezioni (anche multiple) di elementi

Costruttori:

```
List();
List(int);
List(int,boolean); //true permette selezioni multiple
```

Metodi:

```
        void add(String);            int  getItemCount();
        int getSelectedIndex();      int[] getSelectedIndexes();
        String getSelectedItem();    String[] getSelectedItems();
        void select(int);            String getItem(int index)
```

Eventi:

doppio click ←→ actionPerformed()

selezionare o deselezionare ←→ itemStateChange()
getStateChange()

Esempio

```
import java.awt.*;

public class ListsTest extends java.applet.Applet {

  public void init() {
    List lst = new List(5, true);

    lst.add("Hamlet");       lst.add("Claudius");
    lst.add("Gertrude");     lst.add("Polonius");
    lst.add("Horatio");      lst.add("Laertes");
    lst.add("Ophelia");
    add(lst);
  }
}
```

9.13 Barre di scorrimento

Permettono di selezionare un valore in un intervallo

Costruttori:

```
Scrollbar();
Scrollbar(int orientamento);
        Scrollbar.VERTICAL
        Scrollbar.HORIZONTAL
Scrollbar(int orientamento, int valoreiniziale, int ampiezza, int minimo, int massimo);
```

Metodi:

```
        int getMaximum();              void setMaximum(int);
        int getMinimum();              void setMinimum(int);
        int getOrientation();          void setOrientation(int orientamento);
        int getValue();                void setValue(int);
        int getUnitIncrement();        void setUnitIncrement(int);
        int getBlockIncrement();       void setBlockIncrement(int);
```

Eventi:

```
sensore AdjustmentListener
        metodi:
                public void adjustmentValueChanged(AdjustmentEvent e)

AdjustmentEvent
        metodi:
                public int getAdjustmentType()
                        UNIT_INCREMENT
                        UNIT_DECREMENT
                        BLOCK_INCREMENT
                        BLOCK_DECREMENT
                        TRACK           //spostamento assoluto

addAdjustmentListener();
```

Esempio

```
import java.awt.*; import java.awt.event.*;
public class SliderTest extends java.applet.Applet implements AdjustmentListener
{
Label l;
public void init() {
setLayout(new GridLayout(1,2));
l = new Label("1", Label.CENTER);    add(l);
Scrollbar sb = new Scrollbar(Scrollbar.HORIZONTAL,0,0,1,100);
sb.addAdjustmentListener(this);                add(sb);
  }

public Insets getInsets() {return new Insets(15,15,15,15);}

public void adjustmentValueChanged(AdjustmentEvent e) {
int v = ((Scrollbar)e.getSource()).getValue();
l.setText(String.valueOf(v));
repaint();
  }
}
```

9.14 Pannelli con scorrimento

Sono pannelli con sbarre di scorrimento

Costruttori:
```
ScrollPane();
ScrollPane(int);
        ScrollPane.SCROLLBARS_ALWAYS
        ScrollPane.SCROLLBARS_AS_NEEDED
        ScrollPane.SCROLLBARS_NEVER
```

Uno ScrollPane può contenere un solo componente, di solito un pannello (per poter inserire altri componenti)

Esempio

```
ScrollPane sp = new ScrollPane();
Panel p = new Panel();
```

```
sp.add(p);
add(sp);
```

9.15 Canvas

E’ un’area su cui disegnare

Accetta eventi del mouse e della tastiera

Esempio

```
Canvas can = new Canvas();
add(can);
```

9.16 Cursore

Si può definire la forma che il cursore assume sopra un componente

Costruttori:

```
Cursor(int)
        Cursor.CROSSHAIR_CURSOR         Cursor.DEFAULT_CURSOR
        Cursor.E_RESIZE_CURSOR          Cursor.HAND_CURSOR
        Cursor.MOVE_CURSOR              Cursor.N_RESIZE_CURSOR
        Cursor.NE_RESIZE_CURSOR         Cursor.NW_RESIZE_CURSOR
        Cursor.S_RESIZE_CURSOR          Cursor.SE_RESIZE_CURSOR
        Cursor.SW_RESIZE_CURSOR         Cursor.TEXT_CURSOR
        Cursor.W_RESIZE_CURSOR          Cursor.WAIT_CURSOR
```

Metodi:

```
        void  setCursor(Cursor);
        Cursor  getCursor();            Cursor  getPredefinedCursor();
```

Esempio

```
Cursor c = new Cursor(Cursor.CROSSHAIR_CURSOR);
setCursor(c);
```

Esempio

```
import java.awt.*;

public class ColorTest extends java.applet.Applet {
  ColorControls RGBcontrols, HSBcontrols;
  Canvas swatch;

  public void init() {
    setLayout(new GridLayout(1,3,5,15));

    // The color swatch
    swatch = new Canvas();            swatch.setBackground(Color.black);

          // the subpanels for the controls
    RGBcontrols = new ColorControls(this, "Red", "Green", "Blue");
    HSBcontrols = new ColorControls(this, "Hue", "Saturation", "Brightness");

          //add it all to the layout
    add(swatch);    add(RGBcontrols);    add(HSBcontrols);
  }

  public Insets getInsets() {    return new Insets(10,10,10,10);    }
```

```
 void update(ColorControls controlPanel) {
  Color c;
      // get string values from text fields, convert to ints
  int value1 = Integer.parseInt(controlPanel.tfield1.getText());
  int value2 = Integer.parseInt(controlPanel.tfield2.getText());
  int value3 = Integer.parseInt(controlPanel.tfield3.getText());

  if (controlPanel == RGBcontrols) {  // RGB has changed, convert HSB
   c = new Color(value1, value2,value3);

       // convert RGB values to HSB values
   float[] HSB = Color.RGBtoHSB(value1, value2, value3, (new float[3]));
   HSB[0] *= 360;     HSB[1] *= 100;     HSB[2] *= 100;

       // reset HSB fields
   HSBcontrols.tfield1.setText(String.valueOf((int)HSB[0]));
   HSBcontrols.tfield2.setText(“”+((int)HSB[1]));
   HSBcontrols.tfield3.setText(String.valueOf((int)HSB[2]));
  }
  else {  // HSB has changed, update RGB
   c = Color.getHSBColor((float)value1 / 360,
        (float)value2 / 100, (float)value3 / 100);

       // reset RGB fields
   RGBcontrols.tfield1.setText(String.valueOf(c.getRed()));
   RGBcontrols.tfield2.setText(“”+(c.getGreen()));
   RGBcontrols.tfield3.setText(String.valueOf(c.getBlue()));
  }

      //update swatch
      swatch.setBackground(c);
      swatch.repaint();
 }
}
```

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

class ColorControls extends Panel implements FocusListener, ActionListener {
TextField tfield1, tfield2, tfield3;
ColorTest applet;

ColorControls(ColorTest parent, String l1, String l2, String l3) {
        // get hook to outer applet parent
applet = parent;
        //do layouts
setLayout(new GridLayout(3,2,10,10));
tfield1 = new TextField("0");    tfield2 = new TextField("0");
tfield3 = new TextField("0");

add(new Label(l1, Label.RIGHT));
tfield1.addFocusListener(this);
tfield1.addActionListener(this);
add(tfield1);

add(new Label(l2, Label.RIGHT));     tfield2.addFocusListener(this);
tfield2.addActionListener(this);         add(tfield2);
add(new Label(l3, Label.RIGHT));     tfield3.addFocusListener(this);
tfield3.addActionListener(this);         add(tfield3);
  }

 public Insets getInsets() {return new Insets(10,10,0,0);}

 public void focusGained(FocusEvent e) {}

 public void focusLost(FocusEvent e) {applet.update(this);}

 public void actionPerformed(ActionEvent e) {
   if (e.getSource() instanceof TextField)  applet.update(this);
 }
}
```

9.17 Finestre

$$\begin{tikzcd}
 & \text{Component} \arrow[d] & \\
 & \text{Container} \arrow[d] & \\
 & \text{Window} \arrow[dl] \arrow[dr] & \\
\text{Frame} & & \text{Dialog} \arrow[d] \\
 & & \text{FileDialog}
\end{tikzcd}$$

9.17.1 Frames

Un frame è una finestra dipendente dalla piattaforma con titolo, pulsanti per chiusura, ecc.

Costruttori:
Frame();
Frame(String titolo);

Metodi:

public String getTitle()	public void setTitle(String)
public Image getIconImage()	public void setIconImage(Image image)
public MenuBar getMenuBar()	public void setMenuBar(MenuBar mb)
public boolean isResizable()	public void setResizable(boolean resizable)
public boolean isVisible()	public void setVisible(boolean v)
public boolean isShowing()	

setSize(new Dimension(150,150))

public void pack()
public void dispose()
Esempio

```
import java.awt.*;

public class PopupWindow extends java.applet.Applet {
Frame window;
Button open, close;

public void init() {
PopupActions handlebutton = new PopupActions(this);

open = new Button("Open Window");
open.addActionListener(handlebutton);        add(open);

close = new Button("Close Window");
close.addActionListener(handlebutton);       add(close);

window = new BaseFrame1("A Popup Window");
// window.resize(150,150);
window.setSize(new Dimension(150,150);
window.setVisible(true);
}
}


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

class BaseFrame1 extends Frame {
Label l;          String message = "This is a Window";

BaseFrame1(String title) {
super(title);
setLayout(new BorderLayout());

l = new Label(message, Label.CENTER);
l.setFont(new Font("Helvetica", Font.PLAIN, 12));
add("Center", l);
}

public Insets getInsets() {   return new Insets(20,0,25,0);  }
}
```

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

public class PopupActions implements ActionListener {

PopupWindow theApp;

PopupActions(PopupWindow win) { theApp = win; }

public void actionPerformed(ActionEvent e) {
  if (e.getSource() instanceof Button) {
    if (e.getSource() == theApp.open) {
      if (!theApp.window.isShowing())  theApp.window.setVisible(true);
    }
    else if (e.getSource() == theApp.close) {
      if (theApp.window.isShowing())  theApp.window.setVisible(false);
    }
  }
}
}
```

9.17.2 Dialoghi

Un dialogo è una finestra temporanea

Costruttori:

```
Dialog(Frame);
Dialog(Frame,String titolo);
Dialog(Frame,String titolo, boolean modale);
```

Metodi:

```
 public void setSize(int,int)       public void setVisible(boolean)
 public boolean isShowing()         public boolean isVisible()
 public boolean isModal()           public void setModal(boolean b)
 public String getTitle()           public void setTitle(String title)
 public boolean isResizable()       public void setResizable(boolean)
```

Per collegare dialoghi ad applet:

```
Object a = getParent();
while (! (a instanceof Frame)) a=((Component) a).getParent();
Dialog dl = new Dialog((Frame) a, “Dialogo”, false);
```

Esempio

```
import java.awt.*;
public class PopupWindowDialog extends java.applet.Applet {
Frame window;              Button open, close;

public void init() {
PopupActions2 handlebutton = new PopupActions2(this);

open = new Button("Open Window");
open.addActionListener(handlebutton);              add(open);
close = new Button("Close Window");
close.addActionListener(handlebutton);             add(close);
window = new BaseFrame2("A Popup Window");
window.setSize(new Dimension(150,150);             window.setVisible(true);
    }
}


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

public class PopupActions2 implements ActionListener {

PopupWindowDialog theApp;

PopupActions2(PopupWindowDialog win) {        theApp = win; }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() instanceof Button) {
      if (e.getSource() == theApp.open) {
        if (!theApp.window.isShowing())  theApp.window.setVisible(true); }
      else if (e.getSource() == theApp.close) {
               if (theApp.window.isShowing()) theApp.window.setVisible(false);}
     } } }
```

```
import java.awt.*;

class BaseFrame2 extends Frame {
String message = "This is a Window";          TextDialog dl;       Label l;

BaseFrame2(String title) {
super(title);
setLayout(new BorderLayout());

l = new Label(message, Label.CENTER);
l.setFont(new Font("Helvetica", Font.PLAIN, 12));                add("Center", l);

// make a dialog for this window
dl = new TextDialog(this, "Enter Text", true);
dl.setSize(new Dimension(150,100));

Button b = new Button("Set Text");
BaseFrameActions handlebutton = new BaseFrameActions(this);
b.addActionListener(handlebutton);                              add("South", b);
  }

  public Insets getInsets() {  return new Insets(20,0,20,0);}
}
```

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

public class BaseFrameActions implements ActionListener {

  BaseFrame2 theApp;

  BaseFrameActions(BaseFrame2 win) { theApp = win;}

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() instanceof Button) theApp.dl.setVisible(true);}
}
```

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

class TextDialog extends Dialog implements ActionListener {
TextField tf;
BaseFrame2 theFrame;

TextDialog(Frame parent, String title, boolean modal) {
super(parent, title, modal);

theFrame = (BaseFrame2)parent;
setLayout(new BorderLayout(10,10));
setBackground(Color.white);
tf = new TextField(theFrame.message,20);
add("Center", tf);

Button b = new Button("OK");
b.addActionListener(this);
add("South", b);
}

  public Insets getInsets() {return new Insets(30,10,10,10);}

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() instanceof Button) {
      String label = ((Button)e.getSource()).getLabel();
//      if (label == "OK") {
      if (label.equals("OK")) {
        setVisible(false);
        theFrame.l.setText(tf.getText());
      }
    }
  }
}
```

9.17.3 File Dialog

Un dialogo per selezionare il nome di un file

Costruttori:

```
FileDialog(Frame,String titolo);
FileDialog(Frame,String titolo, int tipo);
        FileDialog.LOAD
        FileDialog.SAVE
```

Metodi:

String getDirectory();	String getFile()
void setDirectory(String);	void setFile(String)

9.17.4 Eventi delle finestre

sensore ComponentListener

evento	metodi
finestra mossa	public void ComponentMoved(ComponentEvent e)

sensore WindowListener (WindowAdapter)

evento	metodi
finestra aperta	public void windowOpened(WindowEvent e)
finestra attivata	public void windowActivated(WindowEvent e)
finestra disattivata	public void windowDeactivated(WindowEvent e)
finestra iconificata	public void windowIconified(WindowEvent e)
finestra deiconificata	public void windowDeiconified(WindowEvent e)
finestra in chiusura	public void windowClosing(WindowEvent e)
finestra chiusa	public void windowClosed(WindowEvent e)

addXXXListener();

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

public class FinestraPrincipale extends Frame{
   FileDialog F;
   Label l =new Label("                                                    ");

   /**
    * @param args the command line arguments
    */
  public FinestraPrincipale(String titolo) {
      super(titolo);
      setLayout(new FlowLayout());

      SensoreBottone handlebutton = new SensoreBottone(this);
      Button open = new Button("Open File");
      open.addActionListener(handlebutton);
      add(open);

      add(l);
      F=new FileDialog(this, "Scegli un File");
      F.setVisible(false);
}

public static void main(String[] args) {
   FinestraPrincipale finestra = new FinestraPrincipale("Mia
applicazione");
   finestra.setSize(700,500);
   finestra.setVisible(true);
   SensoreFinestra s = new SensoreFinestra(finestra);
   finestra.addWindowListener(s);
}

}

class SensoreFinestra extends WindowAdapter {

   Frame win;

   public SensoreFinestra(Frame f)
      {  win=f;  }

public void windowClosing(WindowEvent e)
   {
   win.setVisible(false);
   win.dispose();
   System.exit(0);
   }
}

class SensoreBottone implements ActionListener {

FinestraPrincipale theApp;

public SensoreBottone(FinestraPrincipale win) { theApp = win;
}

public void actionPerformed(ActionEvent e) {
       String nomefile;

       if (!theApp.F.isShowing())
          theApp.F.setVisible(true);
       nomefile =theApp.F.getFile();
       theApp.l.setText("nome del file: "+nomefile);
     }
   }
```

9.18 Menu

$$\begin{tikzcd}
 & \text{MenuComponent} \arrow[dl] \arrow[dr] & & \\
\text{MenuBar} & & \text{MenuItem} \arrow[dl] \arrow[dr] & \\
 & \text{Menu} & & \text{CheckboxMenuItem}
\end{tikzcd}$$

- **creazione della barra menu per un finestra:**
  MenuBar mbar = new MenuBar();
- **inserimento della barra nella finestra:**
  window.setMenuBar(mbar);
- **inserimento singoli menu:**
  Menu MyMenu = new Menu("File");
  mbar.add(MyMenu);
  Menu EditMenu = new Menu("Edit");
  mbar.add(EditMenu);
- **inserimento menu di help:**
  Menu helpmenu = new Menu("?");
  mbar.add(helpmenu);
  mbar.setHelpMenu(helpmenu);
- **aggiunta voci (MenuItem, CheckboxMenuItem, Menu, separatori) ai singoli menu:**
  Menu Tools = new Menu("Tools");
  MenuItem Info = new MenuItem("Info");
  MenuItem Colors = new MenuItem("Colors");
  Tools.add(Info);
  Tools.add(Colors);

  Menu sm = new Menu("Sizes");
  MenuItem Small = new MenuItem("Small");

```
MenuItem Medium = new MenuItem("Medium");
MenuItem Large = new MenuItem("Large");
sm.add(Small);
sm.add(Medium);
sm.add(Large);

Tools.add(sm);

CheckboxMenuItem coord = new CheckboxMenuItem("Mostra Coordinate");
Tools.add(coord);

MenuItem sep = new MenuItem("-");
Tools.add(sep);

mbar.add(Tools);
```

metodi MenuItem (e Menu):

public void setEnabled(boolean b)	public boolean isEnabled()

- **scorciatoie:**

  costruttori: MenuShortcut(int);
  MenuShortcut(int,boolean); //devo premere anche Shift?

  ```
  MenuItem copy = new MenuItem("Copia",MenuShortcut('c'););
  ```

metodi MenuShortcut:
public int getKey();

metodi MenuItem :
public MenuShortcut getShortcut();
public void setShortcut(MenuShortcut s)
public void deleteShortcut()

Eventi legati ai MenuItem:

ActionListener	ActionPerformed()

Eventi legati ai CheckboxMenuItem:

ItemListener	ItemStateChanged() (per i CheckboxMenuItem)

- **popmenu**

```
PopupMenu p = new PopupMenu(“Azioni”);
MenuItem C = new MenuItem(“Copia”);
MenuItem I = new MenuItem(“Incolla”);
p.add(C);
p.add(I);
add(p);

public void mouseReleased(MouseEvent e)      {
//in windows funziona questo
 if (e.isPopupTrigger())
    p.show(e.getComponent(),e.getX(),e.getY());
}
```

Esempio

```
import java.awt.*;

public class PopupWindowMenu extends java.applet.Applet {
Frame window;       Button open, close;

public void init() {
PopupActions3 handlebutton = new PopupActions3(this);

open = new Button("Open Window");
open.addActionListener(handlebutton);      add(open);

close = new Button("Close Window");
close.addActionListener(handlebutton);     add(close);

window = new BaseFrame3("A Popup Window");
window.setSize(150,150);       window.setVisible(true);
    }
}
```

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

public class PopupActions3 implements ActionListener {

PopupWindowMenu theApp;

PopupActions3(PopupWindowMenu win) { theApp = win;}

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() instanceof Button) {
      if (e.getSource() == theApp.open) {
        if (!theApp.window.isShowing())  theApp.window.setVisible(true);}
      else if (e.getSource() == theApp.close) {
        if (theApp.window.isShowing()) theApp.window.setVisible(false);  }
    }
  }
}
import java.awt.*;

class BaseFrame3 extends Frame {
  String message = "This is a Window";
  TextDialog2 dl;           Label l;

  BaseFrame3(String title) {
   super(title);
   setLayout(new BorderLayout());

   l = new Label(message, Label.CENTER);
   l.setFont(new Font("Helvetica", Font.PLAIN, 12));
   add("Center", l);

   // make a dialog for this window
   dl = new TextDialog2(this, "Enter Text", true);
   dl.resize(150,120);

BaseFrameActions2 handleact = new BaseFrameActions2(this);

   Button b = new Button("Set Text");
   b.addActionListener(handleact);                add("South", b);
MenuBar mb = new MenuBar();
   Menu m = new Menu("Colors");
   MenuItem redmi = new MenuItem("Red");
   redmi.addActionListener(handleact);
   m.add(redmi);
   MenuItem bluemi = new MenuItem("Blue");
   bluemi.addActionListener(handleact);
   m.add(bluemi);
   MenuItem greenmi = new MenuItem("Green");
   greenmi.addActionListener(handleact);
   m.add(greenmi);
   m.add(new MenuItem("-"));

   CheckboxMenuItem boldmi = new CheckboxMenuItem("Bold Text");
    SensoreCheckboxMenu scm = new SensoreCheckboxMenu(this);
    boldmi.addItemListener(scm);
    m.add(boldmi);
   mb.add(m);
   setMenuBar(mb);
 }

  public Insets getInsets() {   return new Insets(20,0,25,0);}
}



class SensoreCheckboxMenu implements ItemListener {

   BaseFrame3 theFrame;

   SensoreCheckboxMenu (BaseFrame3 a){ theFrame =a;}

   public void itemStateChanged(ItemEvent e) {
    if (theFrame.l.getFont().isPlain()) {
       theFrame.l.setFont(new Font("Helvetica", Font.BOLD, 16));}
    else
       theFrame.l.setFont(new Font("Helvetica", Font.PLAIN, 12));
   }

}
```

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

public class BaseFrameActions2 implements ActionListener {

BaseFrame3 theFrame;

BaseFrameActions2(BaseFrame3 win) {   theFrame = win;         }

public void actionPerformed(ActionEvent e) {
     if (e.getSource() instanceof Button)  theFrame.dl.setVisible(true);
     else if (e.getSource() instanceof MenuItem) {
        String label = ((MenuItem)e.getSource()).getLabel();
        if (label.equals("Red"))  theFrame.l.setBackground(Color.red);
        else if (label.equals("Blue"))  theFrame.l.setBackground(Color.blue);
        else if (label.equals("Green"))  theFrame.l.setBackground(Color.green);
        theFrame.repaint();
     }  }       }
```

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

class TextDialog2 extends Dialog implements ActionListener {
TextField tf;   BaseFrame3 theFrame;

TextDialog2(Frame parent, String title, boolean modal) {
super(parent, title, modal);                    theFrame = (BaseFrame3)parent;
setLayout(new BorderLayout(10,10));           setBackground(Color.white);
tf = new TextField(theFrame.message,20);    add("Center", tf);
Button b = new Button("OK");  b.addActionListener(this);         add("South", b);
  }

public Insets getInsets() {    return new Insets(30,10,30,10);       }

public void actionPerformed(ActionEvent e) {
   if (e.getSource() instanceof Button) {
     String label = ((Button)e.getSource()).getLabel();
     if (label.equals("OK")) { setVisible(false);        theFrame.l.setText(tf.getText());
     }  }        }        }
```

9.19 Applicazioni grafiche standalone

Una applicazione grafica standalone deriva dalla classe Frame

```
import java.awt.*;

class MiaApplicazione extends Frame{
        Button ok = new Button(“OK”);
        Button reset = new Button(“Reset”);

MiaApplicazione(String titolo) {
        super(titolo);
        setLayout(new FlowLayout());
        add(ok);
        add(reset);
}

public static void main(String[] args) {
MiaApplicazione finestra = new MiaApplicazione(“Mia applicazione”);
finestra.setSize(700,500);
finestra.setVisible(true);
}

}
```

chiusura applicazione:

```
public void windowClosing(WindowEvent e)
   {
   win.setVisible(false);
   win.dispose();
   System.exit(0);
   }
```

Esempio

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

public class FinestraPrincipale extends Frame{
   FileDialog F;
   Label l =new Label("                                        ");

  public FinestraPrincipale(String titolo) {
      super(titolo);
        setLayout(new FlowLayout());
      SensoreBottone handlebutton = new SensoreBottone(this);
      Button open = new Button("Open File");
      open.addActionListener(handlebutton);
      add(open);
      add(l);
      F=new FileDialog(this, "Scegli un File");
      F.setVisible(false);

}

public static void main(String[] args) {
   FinestraPrincipale finestra;
        finestra = new FinestraPrincipale("Mia applicazione");
   finestra.setSize(700,500);
   finestra.setVisible(true);
   SensoreFinestra s = new SensoreFinestra(finestra);
   finestra.addWindowListener(s);

}

}

class SensoreFinestra extends WindowAdapter {

   Frame win;

   public SensoreFinestra(Frame f)
      {  win=f;  }

public void windowClosing(WindowEvent e){
   win.setVisible(false);
   win.dispose();
   System.exit(0);
   }

}
class SensoreBottone implements ActionListener {

FinestraPrincipale theApp;

public SensoreBottone(FinestraPrincipale win) { theApp = win; }

public void actionPerformed(ActionEvent e) {
        String nomefile;

        if (!theApp.F.isShowing())
           theApp.F.setVisible(true);

        nomefile =theApp.F.getFile();

        theApp.l.setText("nome del file: "+nomefile);
      }
   }
```

9.20 Stampa

modello 1.1
- definizione del processo di stampa
- stampa sull'oggetto grafico usando printall() o print() (che richiama paint())

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

public class PrintTest extends Frame implements ActionListener {

  public static void main(String[] arg) {
     PrintTest test = new PrintTest();
     test.setSize(500,150);          test.show();
   }

   PrintTest() {
     setLayout(new FlowLayout());
    String str = "Once upon a midnight dreary, while I pondered, weak and weary,\n" +
     "Over many a quaint and curious volume of forgotten lore,\n" +
     "While I nodded, nearly napping, suddenly there came a tapping,\n" +
     "As of some one gently rapping, rapping at my chamber door.\n" +
     "\"'Tis some visitor,\" I muttered, \"tapping at my chamber door-\n" +
     "Only this, and nothing more.\"\n\n";

    add(new TextArea(str));

     Button b = new Button("Print Text");
     b.addActionListener(this);     add(b);
   }

  public void actionPerformed(ActionEvent e) {
     if (e.getSource() instanceof Button) {
        PrintJob job = getToolkit().getPrintJob(this, "Poe: Nevermore",
          (Properties)null);
        if (job != null) {
           Graphics pg = job.getGraphics();
           if (pg != null) {printAll(pg);  pg.dispose();}
           job.end();
        }}}}
```

modello 1.2
- definizione del processo di stampa
- definizione di un oggetto stampabile
- stampa sull'oggetto stampabile (definendo il metodo print())

```
import java.awt.*; import java.awt.event.*; import java.awt.font.*; import java.util.*;
import java.awt.print.*;


public class MioPrintTest
{
  public static void main(String[] args)
    { Frame frame = new PrintTestFrame();
      frame.show();
    }
}

class PrintTestFrame extends Frame implements ActionListener
{
  Button pr = new Button("Print");
  Button ps = new Button("Page setup");
  TextArea ta = new TextArea("Prova stampa",5,30);
  PrintPanel canvas;
  Panel panel;
  private PageFormat pageFormat;

  public PrintTestFrame()
   {  setTitle("PrintTest");
      setSize(300, 300);
      addWindowListener(new WindowAdapter()
         {public void windowClosing(WindowEvent e){System.exit(0);}});
      canvas = new PrintPanel(this);
      panel = new Panel();
      setLayout(new FlowLayout());
      pr.addActionListener(this);
      ps.addActionListener(this);
      panel.setBackground(Color.red);
      add(ta);   panel.add(pr);   panel.add(ps);     add(panel);
    }




  public void actionPerformed(ActionEvent event)
    {  Object source = event.getSource();
       if (source == pr)
      {  PrinterJob printJob = PrinterJob.getPrinterJob();
         if (pageFormat == null)
            pageFormat = printJob.defaultPage();
          printJob.setPrintable(canvas, pageFormat);
         if (printJob.printDialog())
         {  try {printJob.print();}
            catch (PrinterException exception)
              {System.out.println(exception);}
         }
       }
       else if (source == ps)
        { PrinterJob printJob = PrinterJob.getPrinterJob();
          if (pageFormat == null) pageFormat = printJob.defaultPage();
          pageFormat = printJob.pageDialog(pageFormat);
        }
    }
```

```
}

class PrintPanel extends Panel implements Printable
{
 PrintTestFrame app;

 PrintPanel(PrintTestFrame ptf)
   {super();  app = ptf;}

  public int print(Graphics g, PageFormat pf, int page) throws PrinterException
  {  if (page >= 1) return Printable.NO_SUCH_PAGE;  //non stampa altre pagine
     g.setColor(Color.black);
     g.translate((int) pf.getImageableX(), (int) pf.getImageableY());
     g.drawRect(0, 0, (int) pf.getImageableWidth(), (int) pf.getImageableHeight());
     Font f = new Font("Times", Font.PLAIN, 14);
     g.setFont(f);
     g.drawString(app.ta.getText(),30, 50);
     return Printable.PAGE_EXISTS;  //continua la stampa delle altre pagine
  }
}
```

9.21 Taglia e incolla

Un oggetto trasferibile può essere spostato da un componente ad un altro

Il flavor descrive il formato dei dati da trasferire

StringSelection permette d trasferire stringhe di testo

Si può usare la clipboard di sistema o clipboard interne

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

public class CopyPaste extends Frame
        implements ActionListener, ClipboardOwner {
  Button copy, paste;       TextField tfcopy, tfpaste;Clipboard clip;

  public static void main(String[] arg) {
     CopyPaste test = new CopyPaste();
     test.setSize(200,150);
```

```
    test.show();
  }

  CopyPaste() {
    super("Copy and Paste");
    clip = getToolkit().getSystemClipboard();
    setLayout(new FlowLayout());

    copy = new Button("Copy From");        copy.addActionListener(this);  add(copy);
    tfcopy = new TextField("", 25);    add(tfcopy);

    paste = new Button("Paste To");   paste.addActionListener(this);
    paste.setEnabled(false);              add(paste);
    tfpaste = new TextField("",25);      add(tfpaste);
  }


  // needed to satisfy cliboardowner inerface: eseguito quando un altro oggetto
// prende il controllo della clipboard
  public void lostOwnership(Clipboard clip, Transferable contents) {}





  public void actionPerformed(ActionEvent e) {
    if ((e.getSource() == copy) && (tfcopy.getText() != null)) {
       String txt = tfcopy.getText();
       StringSelection trans = new StringSelection(txt);
       clip.setContents(trans,this);
       paste.setEnabled(true);
    }
    else if (e.getSource() == paste) {
      Transferable topaste = clip.getContents(this);
      if (topaste != null) {
        try {
               String txt = (String)topaste.getTransferData(
               DataFlavor.stringFlavor);
               tfpaste.setText(txt);
                paste.setEnabled(false);
        }
        catch (Exception except) {System.out.println("Can't get data.");}
      } }   }
```

}

9.22 Internazionalizzazione

Permette di adattare il programma a lingue diverse

locale: particolare lingua o cultura

diverse parti del programma possono avere differenti locale

Costruttori:

```
 public Locale(String language, String country)

 public Locale(String language, String country, String variant)
```

```
en    GB,  en    US,  it    IT,   fr    FR,  fr    BE ,  DE   DE
. . .
```

metodi:

```
 public Locale getLocale();             public void setLocale(Locale l)
```

NumberFormat, DateFormat, MessageFormat

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

public class InterTest extends Frame {

  public static void main(String[] arg) {
     InterTest test = new InterTest();
     test.setSize(500,150);                test.show();
   }

   InterTest()
   {  setTitle("Test Internazionalizzazione");      setSize(300, 300);
      addWindowListener(new WindowAdapter()
         {  public void windowClosing(WindowEvent e) { System.exit(0);}} );
    setLayout(new FlowLayout());
```

```
//     setLocale(new Locale("en","US"));

    Locale loc = getLocale();
    ResourceBundle labels =  ResourceBundle.getBundle("Tasti",loc);
    Button ok = new Button(labels.getString("okLabel"));
    Button cancel = new Button(labels.getString("cancelLabel"));
    add(ok);      add(cancel);
//NUMERI
    NumberFormat nf1 = NumberFormat.getCurrencyInstance(loc);
    NumberFormat nf2 = NumberFormat.getCurrencyInstance(Locale.ENGLISH);
    System.out.println("Importo: "+nf1.format(123.45));
    System.out.println("Importo: "+nf2.format(123.45));
//DATA
    Date data = new Date(System.currentTimeMillis());
    DateFormat df1 = DateFormat.getDateInstance(DateFormat.DEFAULT,loc);
    DateFormat df2 =
 DateFormat.getDateTimeInstance(DateFormat.DEFAULT,DateFormat.SHORT,loc);
    DateFormat df3 =
DateFormat.getDateTimeInstance
              (DateFormat.DEFAULT,DateFormat.DEFAULT,Locale.ENGLISH);
System.out.println("Data: "+df1.format(data));
System.out.println("Data: "+df2.format(data));
 System.out.println("Data: "+df3.format(data));   //tiene conto del fuso orario
```

```
 //MESSAGGI
  Object vals [] = new Object[4];
  vals[0] = new Integer(4); //deve essere un OGGETTO di tipo intero
  vals[1] = new Integer(56);       vals[2] = "disco 3";          vals[3] = data;
    MessageFormat mf1 = new MessageFormat
              ("oggi {3} {0} documenti sul disco {2} esaminati in {1} secondi");
  MessageFormat mf2 = new MessageFormat("Disk: {2} Files {0} Time elapsed: {1}");
  System.out.println(mf1.format(vals));
        System.out.println(mf2.format(vals));
  }
}
```

file delle risorse (da compilare separatamente):

file Tasti.java: (senza indicazione del Locale fornisce la risorsa di default)

```
import java.util.*;
public class Tasti extends ListResourceBundle {  //scelta di default

public Object [] [] getContents() { return contents;}

static final Object [] [] contents ={{  "okLabel", "Si"}, {  "cancelLabel", "NO"}};
}
```

file Tasti_it_IT.java:

```
import java.util.*;
public class Tasti_it_IT extends ListResourceBundle {
public Object [] [] getContents() {return contents;}
static final Object [] [] contents ={{"okLabel", "Vai"}, {"cancelLabel", "Cancella"}};
}
```

file Tasti_en_US.java:

```
public class Tasti_en_US extends ListResourceBundle {
public Object [] [] getContents() { return contents;}
static final Object [] [] contents ={{"okLabel", "ok"},{"cancelLabel", "Cancel"}};
}
```