Corso sul linguaggio Java

Modulo JAVA5

C1 – Applicazioni grafiche

Prerequisiti

- Programmazione base in Java
- Utilizzo di classi e oggetti
- Utilizzo di elementi grafici ed eventi

Introduzione

Lo scopo di questa Unità è quello di introdurre l'uso del contenitore **Canvas**, un'area di disegno utilizzabile per applicazioni grafiche come istogrammi, grafici di funzioni o disegni.

L'uso dei **Canvas**, congiunto con gli elementi grafici e la gestione degli eventi, consente di realizzare applicazioni grafiche complete.

Il *Canvas*

Il **Canvas** è un esempio di oggetto contenitore destinato alla tracciatura di disegni e all'impostazione di font e colori.

Un possibile schema di utilizzo del **Canvas** è il seguente:

```
import java.awt.*;
public class nomeClasse extends Canvas
{   attributi;
    metodo costruttore e altri;

    public void paint(Graphics g)
    {
        istruzioni e metodi grafici;
        ….
    }
}   /* fine classe*/
```

Il metodo **paint()** è l'unico metodo della classe **Canvas**: viene attivato automaticamente e consente di usare la grafica.

I metodi e le operazioni grafiche vengono attivati tutti sull'oggetto *g*.

Il *Canvas*

La classe di *testing* ha la forma seguente:

```
import java.awt.*;
import java.io.*;
class TestnomeClasse
{
    attributi,
    metodi;
    public static void main(String args[]) throws Exception
    {       Frame f = new Frame ("Area di disegno");
            creazione oggetto g di classe nomeClasse;
            f.setSize (500,400);
            f.add (g);
            f.setVisible (true);
    }
}
```

Aggiunge al *frame* l'area di disegno che, tramite **paint**() esegue la parte grafica

Scritte grafiche

Una delle prime applicazioni che vediamo è la possibilità di scrivere stringhe in modalità grafica.
Alcune classi usate sono:

- **Color** – per la gestione dei colori
- **Font** *nomefont* = **new Font** ("nome", *stile*, *corpo*);

Esempi:

```
Color Colori[]={Color.cyan, Color.magenta, Color.green};
Font arialFont = new Font ("Arial", Font.PLAIN, 20);
Font courierFont = new Font ("Courier", Font.ITALIC, 40);
```

Utilizzo:

```
g.setFont (arialFont);
g.setColor (Colori[1]);
g.drawString (s, 50, 50);
```

Scrive la stringa *s* in posizione (50, 50) con il colore *magenta* e il font *Arial*

Scritte grafiche

```
import java.awt.*;
public class scritteGrafiche extends Canvas
{   private String f1, f2, f3;
    public scritteGrafiche (String F1, String F2, String F3)
       {f1=F1; f2=F2; f3=F3;}
    public void paint(Graphics g)
    { Color Colori[]={Color.cyan, Color.magenta, Color.green};
      Font romanFont = new Font ("TimesRoman", Font.BOLD, 36);
      Font arialFont = new Font ("Arial", Font.PLAIN, 20);
      Font courierFont = new Font ("Courier", Font.ITALIC, 40);
      g.setFont (romanFont);      g.setColor (Colori[0]);
      g.drawString (f1, 50, 50);
      g.setFont (arialFont);      g.setColor (Colori[1]);
      g.drawString (f2, 50, 150);
      g.setFont (courierFont);    g.setColor (Colori[2]);
      g.drawString (f3, 50, 250);
   }
} /* fine classe*/
```

Metodo **paint()**

Array di colori

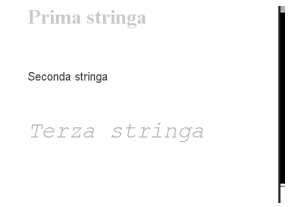
Impostazione font e colore

Tracciatura stringa grafica

Scritte grafiche

ATTIVITA': scrivere la classe di test per la classe *scritteGrafiche*

```
import java.awt.*;
import java.io.*;
class TestscritteGrafiche
{
  public static String Frase;
  public static void main(String args[]) throws Exception
  { Frame f = new Frame ("Area di disegno");
    scritteGrafiche g = new scritteGrafiche
          ("Prima stringa", "Seconda stringa", "Terza stringa");
    f.setSize (500,400);
    f.add (g);
    f.setVisible (true);
   }
}
```



Forme grafiche

```
class DrawOval extends Canvas
{   public void paint (Graphics g)
    {  g.drawOval (10, 10, getSize().width - 20,  getSize().height - 20);
    }
}
```

Disegno forma grafica

Dimensioni finestra grafica

```
class DrawAndFillRect extends Canvas
{
   public void paint (Graphics g)
   {
      g.setColor (Color.gray);
      g.fillRect (10, 10, getSize().width - 20,  getSize().height - 20);
   }
}
```

Riempimento forma grafica

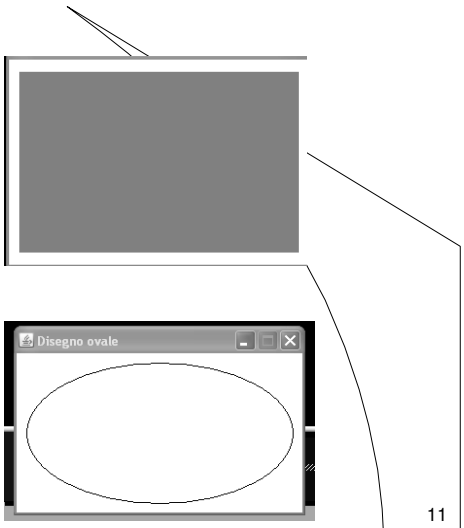
Forme grafiche

ATTIVITA': scrivere la classe *TestCanvas* che visualizzi in due finestre non ridimensionabili *f1* ed *f2,* con titolo "Disegno ovale" e "Disegno rettangolo pieno", rispettivamente un ovale ed un rettangolo pieno utilizzando le classi precedenti.

```
import java.awt.*;
class TestCanvas
{   static public void main(String[] args)
    {Frame f1 = new Frame("Disegno ovale");
     f1.add (new DrawOval());
     f1.setSize (300, 200); f1.setLocation (200,400);
     f1.setResizable (false);          f1.setVisible (true);
     Frame f2 = new Frame ("Disegno rettangolo pieno");
     f2.add (new DrawAndFillRect());
     f2.setSize (300, 200); f2.setLocation (600,200);
     f2.setResizable (false);          f2.setVisible (true);
    }
}
```

Forme grafiche

Questo è l'output della classe.



Disegno di una retta

```
class Retta extends Canvas
{   int x1, y1, x2, y2;
    public Retta (int a1, int b1, int a2, int b2)
    {       x1=a1;
            y1=b1;
            x2=a2;
            y2=b2;
    }
    public void paint(Graphics g)
   {        setBackground (Color.gray);
            g.drawString ("P1(" + x1 + "," + y1 + ")", x1-50, y1-10);
            g.drawString ("P2(" + x2 + "," + y2 + ")", x2-50, y2+30);
            g.drawLine (x1, y1, x2, y2);
   }
}
```

Attributi

Costruttore

Descrizione punti

Tracciatura retta

Disegno di una retta

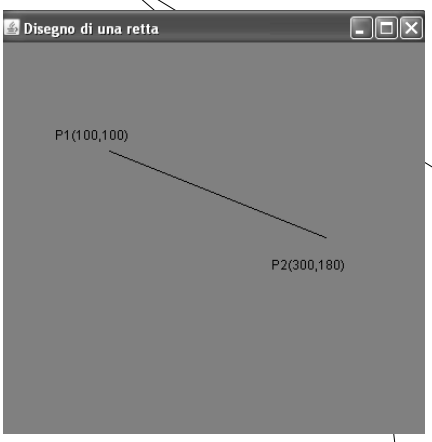
ATTIVITA': scrivere la classe *DisegnaRetta* che visualizzi in una finestra *f* con titolo "Disegno di una retta" una retta ottenuta come oggetto della classe precedente.

```
class DisegnaRetta
{   static public void main(String[] args) throws Exception
    {  InputStreamReader In = new InputStreamReader (System.in);
       BufferedReader Tastiera = new BufferedReader (In);
       System.out.print("x1: ");  int x1 = Integer.parseInt Tastiera.readLine());
       System.out.print("y1: ");  int y1 = Integer.parseInt(Tastiera.readLine());
       System.out.print("x2: ");  int x2 = Integer.parseInt(Tastiera.readLine());
       System.out.print("y2: ");  int y2 = Integer.parseInt(Tastiera.readLine());
       Frame f = new Frame("Disegno di una retta");
       Retta r = new Retta(x1, y1, x2, y2);
       f.add(r);        f.setSize(400, 400);        f.setLocation(200,200);
       f.setVisible (true);
    }
}
```


Disegno di una retta

Esecuzione dell'applicazione con coordinate:

X1 =100;
Y1 =100;
X2 =300;
Y2 = 180



I metodi della classe Graphics

METODO	EFFETTO
void drawLine (**int** *x1*, **int** *y1*, **int** *x2*, **int** *y2*)	Disegna una linea di estremi (*x1*,*y1*) e (*x2*,*y2*)
void drawRect (**int** *x*, **int** *y*, **int** *larg*, **int** *alt*)	Disegna un rettangolo *larg x alt* in posizione (*x*,*y*)
void drawOval (**int** *x*, **int** *y*, **int** *larg*, **int** *alt*)	Disegna un ovale inscritto in un rettangolo *larg x alt* in posizione (*x*,*y*)
void drawArc (**int** *x*, **int** *y*, **int** *larg*, **int** *alt*, **int** *alfa1*, **int** *alfa2*)	Disegna un arco inscritto in un rettangolo *larg x alt* in posizione (*x*,*y*), a partire dall'angolo *alfa1* per un ampiezza di *alfa2*.
void fillRect (**int** *x*, **int** *y*, **int** *larg*, **int** *alt*)	Disegna un rettangolo pieno *larg x alt* in posizione (*x*,*y*)
void fillOval (**int** *x*, **int** *y*, **int** *larg*, **int** *alt*)	Disegna un ovale pieno inscritto in un rettangolo *larg x alt* in posizione (*x*,*y*)
void fillArc (**int** *x*, **int** *y*, **int** *larg*, **int** *alt*, **int** *alfa1*, **int** *alfa2*)	Disegna un arco pieno inscritto in un rettangolo *larg x alt* in posizione (*x*,*y*), a partire dall'angolo *alfa1* per un ampiezza di *alfa2*.
void drawString (**String** *s*, **int** *x*, **int** *y*)	Scrive la stringa *s* in posizione (*x*,*y*)
void setColor (**Color** *c*)	Imposta il colore con il valore *c*
Color getColor ()	Restituisce il colore corrente
void setFont (**Font** *f*)	Imposta il font di caratteri *f* (descritto di seguito)
Font getFont ()	Restituisce il font corrente
void paint (**Graphics** g)	Ridisegna il componente

Coordinate effettive e di schermo

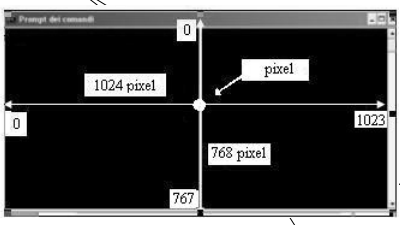
Quando si disegna su finestra grafica *utilizzando un riferimento cartesiano*, occorre distinguere tra **coordinate effettive** e **coordinate di schermo**.

Le **coordinate effettive**:

- sono espresse da numeri reali (positivi e negativi);
- vanno, generalmente, da - ∞ a + ∞;

Le **coordinate di schermo**:

- sono espresse da numeri naturali;
- variano in base alla risoluzione;



PROBLEMA 1: come rappresentare un disegno reale in modo da poterlo inserire all'interno del **piano di visualizzazione**?

PROBLEMA 2: come trasformare le coordinate effettive in quelle di schermo?

Coordinate effettive e di schermo

Per risolvere il **PROBLEMA 1** occorre rappresentare gli intervalli reali (*xMin, xMax*) e (*yMin, yMax*) in coordinate di schermo e poi definire una **unità di misura su X ed Y**.

ATTIVITA': scrivere le istruzioni per calcolare le unità di misura *unitX* sull'asse X e *unitY* sull'asse Y, indicando con (*xMin, xMax*) e (*yMin, yMax*) gli intervalli reali, sapendo che i valori **getSize().height** e **getSize().width** danno le dimensioni correnti in *pixel*.

```
h = getSize().height;
w = getSize().width;
unitX = (double) w / (xMax-xMin);
unitY = (double) h / (yMax-yMin);
```


Coordinate effettive e di schermo

Per risolvere il **PROBLEMA 2** occorre scrivere un metodo *trasforma* con interfaccia:

```
public Point trasforma (double x, double y) { … }
```

che trasforma una coppia di coordinate effettive, in coordinate di schermo, generando un oggetto di classe **Point**.

ATTIVITA': scrivere il metodo pubblico *trasforma()* che genera un oggetto di classe **Point** ricevendo due coordinate *x* ed *y* rappresentate come numeri reali.

```
public Point trasforma (double x, double y)
{
   Point t = new Point();
   t.x = (int) Math.round((x - xMin)*unitX);
   t.y = (int) Math.round((yMax - y)*unitY);
   return t;
}
```

Grafico di una funzione

Abbiamo tutti gli elementi per disegnare un piano cartesiano in coordinate di schermo, *rapportate in scala a quelle reali*.

ATTIVITA': scrivere la classe *DisegnaGrafico()* che crea un *frame* ed aggiunge un *canvas* come piano di visualizzazione in cui genera un oggetto di classe *Grafico*.

```
class DisegnaGrafico
{   static public void main(String[] args)throws Exception
    {  Frame f = new Frame("Disegno di una funzione");
       Grafico g = new Grafico();
       f.add(g);
       f.setSize(400, 400);
       f.setLocation(200,200);
       f.setVisible(true);
    }
}
```


Grafico di una funzione

ATTIVITA': creare la classe *Grafico()* che eredita da **Canvas,** avente come attributi *xMax, xMin, yMax, yMin, unitX, unitY* e formata da un costruttore senza parametri che acquisisce *xMin, xMax, yMin, yMax*.

```
class Grafico extends Canvas   //eredita da Canvas il metodo getSize()
{  double xMax, xMin, yMax, yMin,  unitX, unitY;
   public Grafico() throws Exception   // imposta le dimensioni del piano
   {  InputStreamReader In = new InputStreamReader(System.in);
      BufferedReader Tastiera = new BufferedReader (In);
      System.out.println("Intervallo ascisse:");
      System.out.print("xMin: ");   xMin = Double.parseDouble (Tastiera.readLine());
      System.out.print("xMax: ");  xMax = Double.parseDouble (Tastiera.readLine());
      System.out.print("yMin: ");   yMin = Double.parseDouble (Tastiera.readLine());
      System.out.print("yMax: ");  yMax = Double.parseDouble (Tastiera.readLine());
}
```

Grafico di una funzione

ATTIVITA': aggiungere alla classe *Grafico()* un metodo *asseX* che disegna l'asse X come retta tra i punti *trasformati* di (*xMin, 0*) e (*xMax, 0*).

```
public void asseX(Graphics g)
{
    Point p1, p2;
    p1=trasforma(xMin, 0);
    p2=trasforma(xMax, 0);
    g.setColor(Color.red);
    g.drawLine(p1.x, p1.y, p2.x, p2.y);
}
```


Grafico di una funzione

ATTIVITA': aggiungere alla classe *Grafico()* un metodo *asseY* che disegna l'asse Y come retta tra i punti *trasformati* di (*0, yMin*) e (*0, yMax*);

```
public void asseY(Graphics g)
{
    Point p1, p2;
    p1=trasforma(0, yMin);
    p2=trasforma(0, yMax);
    g.setColor(Color.red);
    g.drawLine(p1.x, p1.y, p2.x, p2.y);
}
```

Grafico di una funzione

ATTIVITA': aggiungere alla classe *Grafico()* un metodo *origine()* che disegna l'origine degli assi con una etichetta (0, 0), posizionata 30 pixel a sinistra e 20 pixel al di sotto dell'origine di schermo.

```
public void origine(Graphics g)
{
    Point p;
    p=trasforma(0, 0);
    g.drawString("(0,0)", p.x-30, p.y+20);
}
```


Grafico di una funzione

ATTIVITA': aggiungere alla classe *Grafico()* il metodo
public static double f (**double** x)
{ **return** *f(x);* *// con f(x) funzione algebrica o trascendente*;
}
che, chiamato da **paint()** disegna il grafico di *f()*

```
public static double f (double x)
{
   return Math.sin (x);      //2 * Math.pow (x, 2)-5*x+3;
}
```

Grafico di una funzione

ATTIVITA': aggiungere alla classe *Grafico()* il metodo **paint()** che, tramite i metodi precedenti, disegna il grafico di *f(x)* come spezzata

```
public void paint (Graphics g)
{  int h, w;
   Point p1, p2;                                  // punti da unire
   double x, deltax=0.01;                         // ascissa e incremento
   Calcolo unità di misura;
   Disegno contorno;                              // area di disegno bordo nero
   Disegno assi e origine;
   p1=trasforma(xMin, f(xMin));                   // primo punto
   g.setColor (Color.black);
   for (x=xMin+deltax; x<=xMax;x+=deltax)         // ciclo che traccia la funzione
   {    p2=trasforma(x, f(x));                    // secondo punto
        g.drawLine(p1.x, p1.y, p2.x, p2.y);       // taccia segmento
        p1.x=p2.x;   p1.y=p2.y;                   // p2 diventa primo punto
   }
}
```


Grafico di una funzione

Calcolo unità di misura

```
h=getSize ().height;
w=getSize().width;
unitX = (double) w/(xMax-xMin);
unitY = (double) h/(yMax-yMin);
```

Disegno contorno

```
g.setColor (Color.black);  g.drawRect (1, 1, w-3, h-3);
```

Disegno assi e origine;

```
asseX(g);
asseY(g);
origine(g);
```

Visualizzazione immagini

Le caratteristiche grafiche di Java consentono in modo semplice la visualizzazione di immagini (file .jpg, .gif, ecc) grazie alla classe **Image.**

La procedura è la seguente:

- creare una classe *Immagine* che eredita da **Canvas;**
- inserire come attributi, una o più immagini mediante il metodo seguente:
  Image img = **Toolkit.getDefaultToolkit**().**getImage** *(imagefile)*;
- inserire nel metodo **paint()** l'istanza di uno dei metodi di disegno
  g.**drawImage** (*imagefile, ascissa, ordinata*, **this**);
  g.**drawImage** (*imagefile, ascissa, ordinata*, *larghezza, altezza,* **this**);

  che traccia l'immagine nel *canvas.*

Visualizzazione immagini

ATTIVITA': creare la classe *VisualizzaImmagine* con un frame *f* avente sfondo grigio chiaro, posizionato in (200, 200) e di ampiezza (750, 400) in cui inserire un oggetto di classe *Immagine.*

```
import java.awt.*;                                  // per la classe
  Graphics
public class VisualizzaImmagine
{   public static void main(String args[] )
  {   Frame f = new Frame("Visualizzatore di immagini");
      f.setBackground(Color.lightGray);             // colore sfondo
      Immagine g =new Immagine();                   // crea oggetto Immagine
      f.add(g);                                     // operazioni sul frame
      f.setLocation(200, 200);  f.setSize(750, 400);  f.setVisible(true);
  }
}
```

Visualizzazione immagini

ATTIVITA': creare la classe *Immagine* per visualizzare i file "auto.gif" (50, 50), "ninfee.jpg" (200, 50, 200, 200) e "Croazia.jpg" (430, 50, 300, 300). Ogni immagine ha una didascalia di colore blu, in "Arial" grassetto, corpo 16.

```
class Immagine extends Canvas
{   Image img1=Toolkit.getDefaultToolkit().getImage ("auto.gif");
    Image img2=Toolkit.getDefaultToolkit().getImage ("ninfee.jpg");
    Image img3=Toolkit.getDefaultToolkit().getImage ("Croazia.jpg");
    public void paint(Graphics g)
    {   g.setFont (new Font ("Arial", Font.BOLD, 16)); g.setColor (Color.blue); //  font e colore
        g.drawString ("Auto d'epoca", 50, 40);
        g.drawString ("Ninfee", 200, 40);                    // didascalie immagini
        g.drawString ("Dalla Croazia", 430, 40);
        g.drawImage (img1,50,50, this);                      // disegno immagini
        g.drawImage (img2, 200, 50, 200, 200, this);
        g.drawImage (img3, 430, 50, 300, 300, this);
    }
}
```


Graphics2D

SLIDE NASCOSTA
CN pag. 22
Contesto grafico
Metodi di Graphics2D
Forme CN 25
Colori CN 26
Trama CN 27
Font CN 28
Immagini (già dette prima, provare ora con Graphics2D) 28

Argomenti

- Il ***Canvas***
- Scritte grafiche
- Forme grafiche
- Disegno di una retta
- I metodi della classe **Graphics**
- Coordinate effettive e di schermo
- Grafico di una funzione
- Visualizzazione immagini