5
Applet

5.1 Introduzione

Un applet è un programma eseguito all'interno di una pagina Web.

Il riferimento all'applet è contenuto nella pagina Web e utilizza uno speciale tag HTML.

Restrizioni per gli applet:

un applet non può leggere o scrivere nel file system locale

un applet non può comunicare con un server diverso da quello da cui è stato scaricato

un applet non può eseguire programmi nel sistema locale

non può caricare programmi nativi o librerie dal sistema locale

Esistono meccanismi per la definizione dei diritti degli applet

5.2 Struttura di un applet

```
import java.awt.Graphics;
// per il metodo paint

public class MyClass extends java.applet.Applet {

public void init()  {  . . .}
//eseguito quando il metodo è caricato inizialmente

public void start()  {  . . .}
//eseguito dopo init o quando l'applet riparte dopo uno stop

public void stop()  {  . . .}
//eseguito quando il brower lascia la pagina che contiene l'applet

public void destroy()  {  . . .}
// eseguito all'uscita dal browser

public void paint(Graphics g)  {  . . .}
//utilizzato per disegnare
}
```

Esempio

```
import java.awt.Graphics;
import java.awt.Font;
import java.awt.Color;

public class HelloAgainApplet extends java.applet.Applet {

  Font f = new Font("TimesRoman",Font.BOLD+Font.ITALIC,36);

  public void paint(Graphics g) {
    g.setFont(f);
    g.setColor(Color.red);
    g.drawString("Hello again!", 5, 40);
    System.out.println ("CIAO!");

  }
}
```

Codice HTML:

```
<HTML>

<HEAD>
        <TITLE>This page has an applet on it</TITLE>
</HEAD>

<BODY>
        <H2>Chapter Eight: Hello Again</H2>
        <P>My second


        Java applet says:

        <BR />
<APPLET CODE="HelloAgainApplet.class" WIDTH=200 HEIGHT=50>
Hello Again!
</APPLET>
</P>
<P>
        <A HREF="estudium.unipg.it">
        Vai ad Estudium
        </A>
</P>
</BODY>
</HTML>
```

5.3 La pagina HTML

<P>My second Java applet says:

<APPLET CODE="HelloAgainApplet.class" WIDTH=200 HEIGHT=50>
Hello Again!
</APPLET><P>

<APPLET > non è un paragrafo e quindi va incluso in un tag di testo, come <P> o <H1>, <H2>, …

il testo fra <APPLET> e </APPLET> viene mostrato se il tag <APPLET> non è compreso dal browser

attributi:

CODE – indica il file dell'applet
CODEBASE – indica la directory o l'URL dove trovare l'applet
WIDTH
HEIGHT
ALIGN – LEFT, RIGTH, TOP, TEXTOP, MIDDLE, ABSMIDDLE, BASELINE, BOTTOM, ABSBOTTOM
HSPACE – spazio, in pixel, fra l'applet ed il testo che lo circonda
VSPACE – spazio, in pixel, fra l'applet ed il testo che lo circonda

Esempio

<HTML>
<HEAD>
<TITLE>This page has an applet on it, aligned left</TITLE>
</HEAD>
<BODY>
<H2>Chapter Eight: Hello Again (Align)</H2>
<P><APPLET CODE="HelloAgainApplet.class"
WIDTH=200 HEIGHT=50 ALIGN=LEFT>Hello Again!</APPLET>
To the left of this paragraph is an applet. It's asimple, unassuming applet, in which a small string isprinted in red type, set in 36 point Times bold.
<BR CLEAR=ALL>
<P>In the next part of the page, we demonstrate how
under certain conditions, styrofoam peanuts can be
used as a healthy snack.<P>
<A HREF="HelloAgainApplet.java">The Source</A>
</BODY>
</HTML>

5.4 Archivi

Un file archivio (.jar) consente di compattare in un unico file tutte le classi, i file di immagine, ecc. di cui l'applet ha bisogno

dovendo scaricare un unico file e quindi dovendo effettuare un'unica connessione, si velocizza il caricamento del file

jar cf Animate.jar *.class *.gif

<APPLET CODE="Animate.class" ARCHIVES="Animate.jar"
WIDTH=200 HEIGHT=50 ALIGN=LEFT>Hello Again!</APPLET>

5.5 Parametri degli applet

Tag <PARAM>:

```
 <P>
<APPLET CODE="HelloApplet.class" WIDTH=200 HEIGHT=50>
 <PARAM NAME=nome VALUE="Pippo">
 Hello to whoever you are!
</APPLET>
<P>
```

Metodo getParameter:

```
 name = getParameter("nome");
```

```
 import java.awt.Graphics;        import java.awt.Font;      import java.awt.Color;

 public class MoreHelloApplet extends java.applet.Applet {

    Font f = new Font("TimesRoman", Font.BOLD, 36);
    String name;

    public void init() {
        name = getParameter("nome");
        if (name == null)   name = "Laura";
      name = "Hello " + name + "!";       }

    public void paint(Graphics g) {
        g.setFont(f);
        g.setColor(Color.red);
        g.drawString(name, 5, 40);  }
                                                                                  }
```

6
Grafica, font e colori

6.1 Disegno

rette:

```
import java.awt.Graphics;

public class MyLine extends java.applet.Applet {
  public void paint(Graphics g) {
    g.drawLine(25,25,75,75);  }
}
```

rettangoli:

```
    g.drawRect(20,20,60,60);
    g.fillRect(120,20,60,60);
    g.drawRoundRect(20,20,60,60,10,10);
    g.fillRoundRect(120,20,60,60,20,20);
    g.draw3DRect(20,20,60,60,true);
    g.fill3DRect(120,20,60,60,false);
```

poligoni:

```
  public void paint(Graphics g) {
    int exes[] = { 39,94,97,142,53,58,26 };
    int whys[] = { 33,74,36,70,108,80,106 };
    int pts = exes.length;
    g.drawPolygon(exes,whys,pts);
/*
Polygon poly = new Polygon(exes,whys,pts);
    g.fillPolygon(poly);
 */
// g.drawPolyline(exes,whys,pts);
 }
```

ovali:

```
    g.drawOval(20,20,60,60);
    g.fillOval(120,20,100,60);
```

archi:

```
   g.drawArc(20,20,60,60,90,180);
   g.fillArc(120,20,60,60,90,180);
   g.drawArc(10,20,150,50,25,-130);
   g.fillArc(10,80,150,50,25,-130);

g.copyArea(0,0,100,100,100,0);

g.clearRect(0,0,getSize().width,getSize().heigth);
```

Esempio

```
import java.awt.*;

public class Lamp extends java.applet.Applet {

  public void paint(Graphics g) {
     // the lamp platform
     g.fillRect(0,250,290,40);


     // the base of the lamp
     g.drawLine(125,250,125,160);
     g.drawLine(175,250,175,160);

     // the lamp shade, top and bottom edges
      g.drawArc(85,157,130,50,-65,312);
      g.drawArc(85,87,130,50,62,58);

      // lamp shade, sides
      g.drawLine(85,177,119,89);
      g.drawLine(215,177,181,89);

      // dots on the shade
      g.fillArc(78,120,40,40,63,-174);
      g.fillOval(120,96,40,40);
      g.fillArc(173,100,40,40,110,180);
  }
}
```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

6.2 Testo e font

```
import java.awt.Font;
import java.awt.Graphics;

public class ManyFonts extends java.applet.Applet {

  public void paint(Graphics g) {
     Font f = new Font("TimesRoman", Font.PLAIN, 18);
     Font fb = new Font("TimesRoman", Font.BOLD, 18);
     Font fi = new Font("TimesRoman", Font.ITALIC, 18);
     Font fbi = new Font("TimesRoman", Font.BOLD + Font.ITALIC, 18);

     g.setFont(f);
     g.drawString("This is a plain font", 10, 25);
     g.setFont(fb);
     g.drawString("This is a bold font", 10, 50);
     g.setFont(fi);
     g.drawString("This is an italic font", 10, 75);
     g.setFont(fbi);
     g.drawString("This is a bold italic font", 10, 100);
  }
}
```

6.3 Colori

```
import java.awt.Color;

Color c = new Color(255,255,0);        //giallo  //RGB  //Red Green Blue

g.setColor(c);                         g.getColor();

this.setBackground(c);                 this.setForeground(c);
```

Colori standard:

Color.white	Color.black	Color.ligthGray	Color.gray
Color.darkGray	Color.red	Color.green	Color.blue
Color.yellow	Color.magenta	Color.cyan	Color.pink
Color.orange			

Esempio

```
import java.awt.Graphics;
import java.awt.Color;

public class ColorBoxes extends java.applet.Applet {

    public void paint(Graphics g) {
        int rval, gval, bval;

        for (int j = 30; j < (getSize().height -25); j += 30)
          for (int i = 5; i < (getSize().width -25); i += 30) {
             rval = (int)Math.floor(Math.random() * 256);
             gval = (int)Math.floor(Math.random() * 256);
             bval = (int)Math.floor(Math.random() * 256);

             g.setColor(new Color(rval,gval,bval));
             g.fillRect(i, j, 25, 25);
             g.setColor(Color.black);
             g.drawRect(i-1, j-1, 25, 25);
          }
    }
}
```

Colori di sistema:

SystemColor.activeCaption SystemColor.activeCaptionBorder

SystemColor.inactiveCaption SystemColor.inactiveCaptionBorder

SystemColor.activeCaptionText
SystemColor.inactiveCaptionText

SystemColor.desktop SystemColor.window SystemColor.windowText

SystemColor.TextHighlight SystemColor.TextHighlightText

```
import java.awt.Graphics;
import java.awt.SystemColor;

public class SysColor extends java.applet.Applet {

        public void init() {
      setBackground(SystemColor.window);
        }

        public void paint(Graphics g) {
      g.setColor(SystemColor.windowText);
      g.drawString("Window text is this color", 20, 50);
      g.setColor(SystemColor.activeCaption);
      g.fillRect(5, 58, 180, 19);
      g.setColor(SystemColor.activeCaptionText);
      g.drawString("Active caption colors", 20, 70);
      g.setColor(SystemColor.inactiveCaption);
      g.fillRect(5, 78, 180, 19);
      g.setColor(SystemColor.inactiveCaptionText);
      g.drawString("Inactive caption colors", 20, 90);
        }
}
```

7
Animazioni, immagini e suoni

7.1 Thread

Usiamo i thread quando vogliamo portate avanti più operazioni in contemporanea.

Esempio:
un server che risponde contemporaneamente a più client collegati

Un caso particolare di operazioni in contemporanea è:
una operazione computazionalmente costosa
le altre attività di sistema

L'uso del thread permette di decidere di sospendere momentaneamente le operazioni in corso (usando il metodo sleep(int))

Un thread può essere considerato un insieme di operazioni (all'interno di uno stesso processo).

Un programma Java può eseguire più Thread contemporaneamente

Java offre una classe predefinita Thread
Java offre una interfaccia predefinita Runnable

In entrambi i casi le operazioni da fare in contemporanea vanno definite all'interno di un metodo che si chiama run()

Cambia un po' il modo in cui metodo run() viene eseguito.

class Animazione extends java.applet.Applet, Thread { //in Java questo non si può fare

class Animazione extends java.applet.Applet implements Runnable { //però questo si può fare

```
public class DigitalClock extends java.applet.Applet  implements Runnable {
//Runnable è un interfaccia: contiene le definizioni  metodi (run in questo caso)
//        comuni a più classi
   Thread runner;

   public void start() {
      if (runner == null) {
         runner = new Thread(this);
         runner.start();       //il thread viene lanciato
      }
   }

   public void stop() {
      if (runner != null) {
         runner.interrupt();          //il thread viene fermato
         runner = null;
      }
   }


   public void run() {
// istruzioni eseguite dal thread:
//ciclo
         //calcola il prossimo fotogramma
         //disegna il prossimo fotogramma
         //aspetta un pochino richiamando sleep()
}
```

```
import java.awt.Graphics;
import java.awt.Font;
import java.util.Calendar;
import java.util.GregorianCalendar;

public class DigitalClock extends java.applet.Applet
   implements Runnable {

   Font theFont = new Font("TimesRoman",Font.BOLD,24);
   GregorianCalendar theDate;
   Thread runner;

   public void start() {
      if (runner == null) {
         runner = new Thread(this);   //this è l'applet
         runner.start();        }
   }

   public void stop() {
      if (runner != null) {
         runner.interrupt();
         runner = null;        }
   }

   public void run() {
      while (true) {
         repaint();  //cancella e richiama paint
         try { Thread.sleep(1000); }
         catch (InterruptedException e) { e.printStackTrace();}
      }
   }

   public void paint(Graphics g) {
      theDate = new GregorianCalendar();
      g.setFont(theFont);
      g.drawString("" + theDate.getTime(), 10, 50);
   }
}
```

7.2 Sfarfallio

Per ridisegnare lo schermo si usa repaint() che a sua volta richiama update():

```
public void update (Graphics g) {
        g.SetColor(getBackground());
        g.fillRect(0,0,getWidth(),getHeight());
        g.setColor(getForeground());
        paint(g);
        }
```

le parti dello schermo che non cambiano passano rapidamente dall'essere cancellate all'essere ridisegnate; questo provoca lo sfarfallio

rimedi:
- ridefinire update() in modo che non cancelli lo schermo o cancelli solo la parte di schermo che cambia
- usare il doppio buffering: lo schermo viene ridisegnato su di un buffer a parte e poi il buffer viene copiato sull'intera superficie dello schermo

Esempio

```
import java.awt.Graphics;        import java.awt.Color;    import java.awt.Font;

public class ColorSwirl extends java.applet.Applet implements Runnable {

   Font f = new Font("TimesRoman", Font.BOLD, 48);
   Color colors[] = new Color[50];
   Thread runner;

   public void start() {
      if (runner == null) { runner = new Thread(this); runner.start();}
   }

   public void stop() {
      if (runner != null) { runner.interrupt(); runner = null; }
   }



  public void run() {

      // initialize the color array
      float c = 0;
      for (int i = 0; i < colors.length; i++) {
         colors[i] =  Color.getHSBColor(c, (float)1.0,(float)1.0);  //Hue Saturation
Brightness
         c += .02;
      }

      // cycle through the colors
      int i = 0;
      while (true) {
         setForeground(colors[i]);
         repaint();
         i++;
         try { Thread.sleep(200); }
         catch (InterruptedException e) { }
         if (i == colors.length ) i = 0;
      }
   }
```

```
public void paint(Graphics g) {
        g.setFont(f);
        g.drawString("Look to the Cookie!", 15, 50);
    }

    public void update(Graphics g) { paint(g); }
}
```

7.3 Immagini

Caricamento dell'immagine:

- getImage(URL);
- getImage(URL,path);

getDocumentBase() restituisce un oggetto URL rappresentante la directory del file HTML che contiene l'applet

getCodeBase() restituisce una stringa contenente la directory dell'applet

Esempio

```
import java.awt.Image;

Image img;

    img = getImage(new URL( "http://www.server.com/files/image.gif"));

    img = getImage(getDocumentBase(), "imag.gif");

    img = getImage(getCodeBase(), "images/ladybug.gif");
```

dimensioni dell'immagine:

```
img.getWidth(this);

img.getHeight(this);
```

visualizzazione immagine:

```
    g.drawImage(img, 10, 10, this);


    int iwidth = img.getWidth(this);
    int iheight = img.getHeight(this);


    // 25 %
    g.drawImage(bugimg, 30, 10, iwidth / 4, iheight / 4, this);
```

Esempio

```
import java.awt.Graphics;
import java.awt.Image;

public class LadyBug extends java.applet.Applet {

   Image bugimg;

   public void init() {
      bugimg = getImage(getCodeBase(), "images/ladybug.gif");
   }

   public void paint(Graphics g) {
      int iwidth = bugimg.getWidth(this);
      int iheight = bugimg.getHeight(this);
      int xpos = 10;

      // 25 %
      g.drawImage(bugimg, xpos, 10,   iwidth / 4, iheight / 4, this);

      // 50 %
      xpos += (iwidth / 4) + 10;
      g.drawImage(bugimg, xpos , 10,  iwidth / 2, iheight / 2, this);

      // 100%
      xpos += (iwidth / 2) + 10;
      g.drawImage(bugimg, xpos, 10, this);

      // 150% x, 25% y
      g.drawImage(bugimg,10,iheight+ 30, (int)(iwidth * 1.5), iheight / 4,
this);
   }
}
```

7.4 Neko: un esempio di animazione

```
import java.awt.Graphics;  import java.awt.Image;  import java.awt.Color;

public class Neko extends java.applet.Applet implements Runnable {

   Image nekopics[] = new Image[9];
   Image currentimg;
   Thread runner;
   int xpos;
   int ypos = 50;

   public void init() {
      String nekosrc[] = { "right1.gif", "right2.gif", "stop.gif", "yawn.gif",
"scratch1.gif", "scratch2.gif","sleep1.gif", "sleep2.gif","awake.gif" };

      for (int i=0; i < nekopics.length; i++)
         nekopics[i] = getImage(getCodeBase(), "images/" + nekosrc[i]);
   }

   public void start() {
if (runner == null) { runner = new Thread(this);      runner.start();}
   }

   public void stop() {
      if (runner != null) { runner.interrupt();  runner = null; }
   }
```

```
public void run() {
        setBackground(Color.white);

        // run from one side of the screen to the middle
        nekorun(0, getSize().width / 2);

        // stop and pause
        currentimg = nekopics[2];
        repaint();        pause(1000);

        // yawn
        currentimg = nekopics[3];
        repaint();     pause(1000);


  // scratch four times
        nekoscratch(4);

        // sleep for 5 "turns"
        nekosleep(5);

        // wake up and run off
        currentimg = nekopics[8];
        repaint();        pause(500);
        nekorun(xpos, getSize().width + 10);
    }

    void nekorun(int start, int end) {
        for (int i = start; i < end; i += 10) {
            xpos = i;
            // swap images
            if (currentimg == nekopics[0])
                currentimg = nekopics[1];
            else currentimg = nekopics[0];
                repaint();
            pause(150);
        }
    }
```

```
    void nekoscratch(int numtimes) {
        for (int i = numtimes; i > 0; i--) {
            currentimg = nekopics[4];
            repaint();          pause(150);
            currentimg = nekopics[5];
            repaint();          pause(150);
        }
    }

    void nekosleep(int numtimes) {
        for (int i = numtimes; i > 0; i--) {
            currentimg = nekopics[6];
            repaint();          pause(250);
            currentimg = nekopics[7];
            repaint();          pause(250);
        }
    }

    void pause(int time) {
        try { Thread.sleep(time); }
        catch (InterruptedException e) { }
    }

    public void paint(Graphics g) {
        if (currentimg != null)
            g.drawImage(currentimg, xpos, ypos, this);
    }
}
```

7.5 Suoni

Suono file audio (.au):

play(new URL("http://www.server.com/files/s.au"));

play(getDocumentBase(), "s.au"); .waw .midi

play(getCodeBase(), "audio/s.au");

caricamento file audio:

AudioClip clip = getAudioClip(getCodeBase(), "audio/s.au");

suono:

clip.loop();

clip.stop();

clip.play();

Esempio

```
import java.awt.Graphics;          import java.applet.AudioClip;

public class AudioLoop extends java.applet.Applet implements Runnable {

   AudioClip bgsound;
   AudioClip beep;
   Thread runner;

   public void start() {
       if (runner == null) { runner = new Thread(this); runner.start(); }
   }

   public void stop() {
      if (runner != null) {
         if (bgsound != null) bgsound.stop();
         runner.interrupt();  runner = null; }
   }

   public void init() {
      bgsound = getAudioClip(getCodeBase(),"audio/loop.au");
      beep = getAudioClip(getCodeBase(), "audio/beep.au");
   }

   public void run() {
      if (bgsound != null) bgsound.loop();
      while (runner != null) {
         try { Thread.sleep(5000); }
         catch (InterruptedException e) { }
         if (beep != null) beep.play();  //ogni 5 secondi si sovrappone al
                                          //suono della base
      }
   }

   public void paint(Graphics g) {
      g.drawString("Playing Sounds....", 10, 10);
   }
}
```

7.6 Doppio buffering

E' una tecnica per evitare lo sfarfallio: lo schermo viene ridisegnato su di un buffer a parte e poi il buffer viene copiato sull'intera superficie dello schermo

E' un sistema di ottima resa grafica anche se un po' inefficiente

```
Image offscreenImg;
Graphics offscreenG;

  offscreenImg = createImage(this.getSize().width, this.getSize().height);
  offscreenG = offscreenImg.getGraphics();


public void update(Graphics g) {  paint(g); }

public void paint(Graphics g) {
  // Draw background
  offscreenG.setColor(Color.black);
  offscreenG.fillRect(0,0,100,100);
  offscreenG.setColor(Color.white);
  offscreenG.fillRect(100,0,100,100);

  // Draw checker
  offscreenG.setColor(Color.red);
  offscreenG.fillOval(xpos,5,90,90);

  g.drawImage(offscreenImg,0,0,this);
}

public void destroy() {    offscreenG.dispose(); }
```

Esempio

```
import java.awt.Graphics;   import java.awt.Color;   import java.awt.Image;

public class Checkers2 extends java.applet.Applet implements Runnable {

  Thread runner;
  int xpos;
  Image offscreenImg;
  Graphics offscreenG;

  public void init() {
    offscreenImg = createImage(this.getSize().width, this.getSize().height);
    offscreenG = offscreenImg.getGraphics();
  }

  public void start() {
    if (runner == null); { runner = new Thread(this); runner.start(); }
  }


  public void stop() {
    if (runner != null) { runner.interrupt();  runner = null;}
  }

  public void run() {
    while (true) {
      for (xpos = 5; xpos <= 105; xpos+=4) {
        repaint();
        try { Thread.sleep(100); }
        catch (InterruptedException e) { }
      }
    }
  }

  public void update(Graphics g) {  paint(g); }
```

```
public void paint(Graphics g) {
   // Draw background
   offscreenG.setColor(Color.black);
   offscreenG.fillRect(0,0,100,100);
   offscreenG.setColor(Color.white);
   offscreenG.fillRect(100,0,100,100);

   // Draw checker
   offscreenG.setColor(Color.red);
   offscreenG.fillOval(xpos,5,90,90);

   g.drawImage(offscreenImg,0,0,this);
  }

  public void destroy() {   offscreenG.dispose(); }
}
```

8
Eventi del mouse e della tastiera

8.1 Il modello degli eventi

Evento: risposta ad una azione che accade durante l'esecuzione di una applicazione

Eventi del mouse, della tastiera, dell'interfaccia utente, di finestre

- individuare gli eventi a cui si è interessati ed il sensore di eventi (listener) associato
- creare il codice per il sensore
- registrare il sensore nell'applet

Eventi mouse:

sensore MouseListener (MouseAdapter)

evento	metodi
mouse giù	public void mousePressed(MouseEvent e)
mouse su	public void mouseReleased(MouseEvent e)
mouse clicca	public void mouseClicked(MouseEvent e)
mouse entra	public void mouseEntered(MouseEvent e)
mouse esce	public void mouseExited(MouseEvent e)

sensore MouseMotionListener (MouseMotionAdapter)

evento	metodi
mouse mosso	public void mouseMoved(MouseEvent e)
mouse drag	public void mouseDragged(MouseEvent e)

Eventi tastiera:

sensore KeyListener (KeyAdapter)

evento	metodi
tasto giù	public void keyPressed(KeyEvent e)
tasto su	public void keyReleased(KeyEvent e)
tasto premuto	public void keyTyped(KeyEvent e)

Implementazione sensore:

si può realizzare una classe sensore separata o definire l'applet stesso come sensore

si devono implementare tutti i metodi dell'interfaccia sensore scelto, a meno di usare un adattatore:

Interfacce	Classi
MouseListener ←→	MouseAdapter
MouseMotionListener ←→	MouseMotionAdapter
KeyListener ←→	KeyAdapter

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

public class eventomouse implements MouseMotionListener     {

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())); }

public void mouseDragged(MouseEvent e){}
}
```

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

public class mioApplet extends java.applet.Applet
        implements MouseMotion, KeyListener     {
. . .}
```

Registrazione sensore

```
eventomouse em;

        public void init () {
em = new eventomouse(this);
addMouseMotionListener(em);
. . .}
```

Se l'applet stesso è il sensore:
addMouseMotionListener(this);

Esempio (implemento l'interfaccia MouseListener)

```
/* draw blue spots at each mouse click */
import java.awt.Graphics; import java.awt.Color; import java.awt.event.*;

public class Spots extends java.applet.Applet implements MouseListener{

   final int MAXSPOTS = 10;
   int xspots[] = new int[MAXSPOTS];
   int yspots[] = new int[MAXSPOTS];
   int currspots = 0;

   public void init() {
      setBackground(Color.white);
      this.addMouseListener(this);
   }

        // needed to satisfy listener interfaces
        public void mouseClicked(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
        public void mouseReleased(MouseEvent e) {    }

   public void mousePressed(MouseEvent e) {
         if (currspots < MAXSPOTS) { addspot(e.getX(),e.getY());}
      else { System.out.println("Too many spots.");}
   }

   void addspot(int x,int y) {
       xspots[currspots] = x;
       yspots[currspots] = y;
       currspots++;
       repaint();
   }

   public void paint(Graphics g) {
      g.setColor(Color.blue);
      for (int i = 0; i < currspots; i++) {
           g.fillOval(xspots[i] - 10, yspots[i] - 10, 20, 20);
```

```
      }
   }
}
```

Esempio (Estendo la classe MouseAdapter)

```
/* draw blue spots at each mouse click */
import java.awt.*;   import java.awt.event.*;

class Sensore extends MouseAdapter{

        Spots riferimento;

   public Sensore(Spots applet){
        riferimento= applet;

}

   public void mousePressed(MouseEvent e) {
         if (riferimento.currspots < riferimento.MAXSPOTS)
                { riferimento.addspot(e.getX(),e.getY());}
      else { System.out.println("Too many spots.");}
   }
}
```

```
public class Spots extends java.applet.Applet {

   final int MAXSPOTS = 10;
   int xspots[] = new int[MAXSPOTS];
   int yspots[] = new int[MAXSPOTS];
   int currspots = 0;

   public void init() {
      setBackground(Color.white);
      Sensore s= new Sensore(this);
      this.addMouseListener(s);
   }


   void addspot(int x,int y) {
       xspots[currspots] = x;
       yspots[currspots] = y;
       currspots++;
       repaint();
   }

   public void paint(Graphics g) {
      g.setColor(Color.blue);
      for (int i = 0; i < currspots; i++) {
           g.fillOval(xspots[i] - 10, yspots[i] - 10, 20, 20);
     }
   }
}
```

Esempio

```
/* draw lines at each click and drag */
import java.awt.Graphics;         import java.awt.Color;
import java.awt.Point;     import java.awt.event.*;

public class LinesNew extends java.applet.Applet
  implements MouseListener,MouseMotionListener {

   final int MAXLINES = 10;
   Point starts[] = new Point[MAXLINES]; // starting points
   Point ends[] = new Point[MAXLINES];    // endingpoints
   Point anchor;    // start of current line
   Point currentpoint; // current end of line
   int currline = 0; // number of lines

   public void init() {
      setBackground(Color.white);
                // register event listeners
        this.addMouseListener(this);
        addMouseMotionListener(this);}

        // needed to satisfy listener interfaces
   public void mouseClicked(MouseEvent e) {}
   public void mouseEntered(MouseEvent e) {}
   public void mouseExited(MouseEvent e) {}

   public void mousePressed(MouseEvent e) {
     if (currline < MAXLINES)
        anchor = new Point(e.getX(),e.getY());
     else
        System.out.println("Too many lines.");
   }

   public void mouseReleased(MouseEvent e) {
      if (currline < MAXLINES)  addline(e.getX(),e.getY());    }

   public void mouseMoved(MouseEvent e) {}

  public void mouseDragged(MouseEvent e) {
     if (currline < MAXLINES) {
        currentpoint = new Point(e.getX(),e.getY());
        repaint();        }
```

```
    }

    void addline(int x,int y) {
        starts[currline] = anchor;
        ends[currline] = new Point(x,y);
        currline++;
          anchor = null;
          currentpoint = null;
        repaint();
    }

    public void paint(Graphics g) {

        // Draw existing lines
        for (int i = 0; i < currline; i++) {
            g.drawLine(starts[i].x, starts[i].y,
                ends[i].x, ends[i].y);
        }

        // draw current line
        g.setColor(Color.blue);
        if (currentpoint != null)
            g.drawLine(anchor.x,anchor.y,
            currentpoint.x,currentpoint.y);
    }
}
```

Esempio

```
/* press a key then use arrows to move it around */

import java.awt.Graphics;        import java.awt.Color;
import java.awt.event.*;         import java.awt.Font;

public class Keys extends java.applet.Applet implements KeyListener{

  char currkey;
  int currx;
  int curry;

  public void init() {
    currx = (this.getSize().width / 2) -8;  //default
    curry = (getSize().height / 2) -16;
    setBackground(Color.white);
    setFont(new Font("Helvetica",Font.BOLD,36));
    addKeyListener(this);
  }


  public void keyReleased(KeyEvent e) {}
   public void keyTyped(KeyEvent e) {}

  public void keyPressed(KeyEvent e) {
        int key = e.getKeyCode();
        switch (key) {
                case KeyEvent.VK_DOWN:
                        curry += 5;   break;
                case KeyEvent.VK_UP:
                        curry -= 5;    break;
                case KeyEvent.VK_LEFT:
                        currx -= 5;    break;
                case KeyEvent.VK_RIGHT:
                        currx += 5;   break;
                default:
                        //      currkey = (char)key;
                        currkey =e.getKeyChar();
                }
           repaint();
  }
```

```
public void paint(Graphics g) {
   if (currkey != 0) {
     g.drawString(String.valueOf(currkey), currx,curry);
//     g.drawString("”+currkey, currx,curry);

   }
  }
  }
```

metodi eventi mouse:
getX() getY() getClickCount() IsPopupTrigger()

metodi eventi tastiera:
getKeyCode() getKeyChar()

metodi eventi mouse e tastiera:
isShiftDown(); isControlDown(); isAltDown(); isMetaDown();

codici tasti speciali:

VK_CLEAR	VK_DOWN	VK_END	VK_ESCAPE
VK_HELP	VK_HOME	VK_LEFT	VK_INSERT
VK_PAUSE	VK_RIGHT	VK_UP	VK_F1…VK_F12
VK_NUMPAD0…VK_NUMPAD9		VK_PRINTSCREEN	
VK_PAGE_DOWN		VK_PAGE_UP	