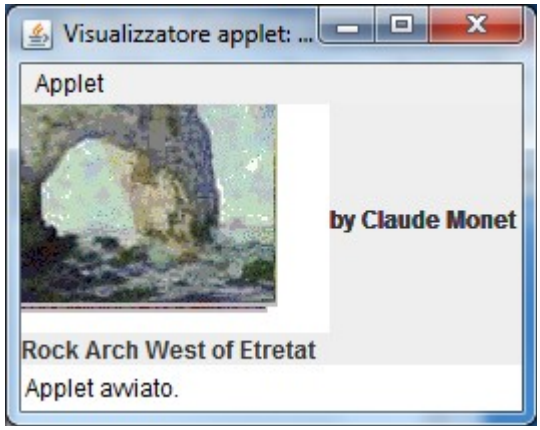
```
/**
 * @(#)SlideShow.java
 *
 * SlideShow JApplet si confronti come SWING forzi unico thread [*]
 *
 * modificato da http://java.sun.com/developer/technicalArticles/Threads/applet/
 *
 * lanciare con:  appletviewer -J-Djava.security.policy=java.policy slides.html
 *                    se URL  "http://developer.java.sun.com:8080/" +
 *                                "developer/technicalArticles/Threads/applet/"
 *
 * @author 4AI_SIS
 * @version 1.00 2011/6/3
 */

import java.awt.*;
import javax.swing.*;
import java.net.*;

public class SlideShow extends JApplet {

  private Image[] images;
  private String[] text;
  private JLabel captions;

// La keyword volatile è legata, generalmente, al multithreading del linguaggio Java.
// Essa indica alla Java Virtual Machine (JVM) che una variabile può essere modificata
// in modo "asincrono" da diversi thread. La JVM in questo modo ricarica e salva in memoria
// il valore della variabile ad ogni accesso alla variabile.
  private volatile int curFrame;
  private volatile boolean noStopRequested;

  private Thread timerThread;
  private boolean paused;
  private final Object pauseLock = new Object();

  private void printThreadName(String prefix) {
    String name = Thread.currentThread().getName();
    System.out.println(prefix + name);
  }

  public void init() {
    images = new Image[3];
    text = new String[3];
    captions = new JLabel();
    setLayout(new BorderLayout());
    add(captions,BorderLayout.PAGE_END);   // dove apparirà il testo di ogni slide
    URL fig = null;
    try {
          fig = new URL(getCodeBase(),"IMG/"); // immagini in sottocartella IMG con percorso del bytecode
    } catch (java.net.MalformedURLException ex) {
                System.out.println("Bad URL");
                return;
            }
    images[0] = getImage(fig, "bordighera.jpg");
    images[1] = getImage(fig, "etretat.jpg");
    images[2] = getImage(fig, "leyden.jpg");
```

```
  text[0] = "Garden in Bordighera";
  text[1] = "Rock Arch West of Etretat";
  text[2] = "Bulbfield and Windmill near Leyden";
  printThreadName("init is ");
  startThread();
 }

 private void startThread() {
   paused = true;
   noStopRequested = true;
                                                        // Use this inner class to hide the public run method
   Runnable r = new Runnable() { public void run() {
                                                          runWork();
                                                        }
                                      };
   timerThread = new Thread(r, "Timer");
   timerThread.start();
   printThreadName("startThread is ");
}

private void stopThread() {
  noStopRequested = false;
  timerThread.interrupt();
  printThreadName("stopThread is ");
}

private void runWork() {                // note that this is private
  printThreadName("run is ");
  curFrame = 0;
  try {
          while ( noStopRequested ) {
                                          waitWhilePaused();
                                          curFrame = ( curFrame + 1 ) % images.length;
                                          repaint();
                                          Thread.sleep(3000);
                                        }
  } catch ( InterruptedException x ) {     // reassert interrupt
                                           Thread.currentThread().interrupt();
                                           System.out.println("interrupt and return from run");
                                        }
}
private void setPaused(boolean newPauseState) {
  synchronized ( pauseLock ) {
                                      if ( paused != newPauseState ) {
                                         paused = newPauseState;
                                         pauseLock.notifyAll();
                                       }
                                     }
}

private void waitWhilePaused() throws InterruptedException {
  synchronized ( pauseLock ) {
                                        while ( paused ) {
                                             pauseLock.wait();
                                         }
                                     }
}
```

```
  public void paint(Graphics g) {
    update(g);                          // solo dalla seconda presentazione appaiono le legende
                                        // etichette captions a fondo pagina  [*]
    printThreadName("paint is ");
  }

  public void update(Graphics g) {
    g.drawImage(images[curFrame], 0, 0, this);
    captions.setText(text[curFrame]);

    // si noti - ridimensionando la finestra - la necessità della gestione dello sfondo
    // la scritta infatti risulta sovrascritta
    JLabel name = new JLabel("by Claude Monet ", JLabel.CENTER); // indifferente l'allineamento
    add(name, BorderLayout.LINE_END);  // regione ad EAST

    printThreadName("update is ");
  }

  public void start() {
    setPaused(false);
    printThreadName("start is ");
  }

  public void stop() {
    setPaused(true);
    printThreadName("stop is ");
  }

  public void destroy() {
    stopThread();

    for (int i = 0; i < images.length; i++) {
      images[i].flush();
      images[i] = null;
      text[i] = null;
    }

    images = null;
    text = null;
    printThreadName("destroy is ");
  }
}
```



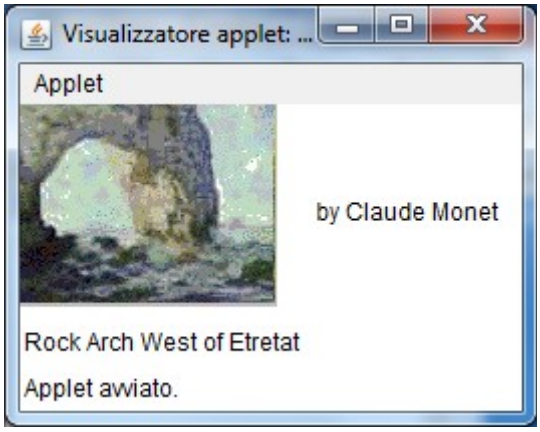
Pagina html:

```
<html>
    <head><title>Mostra Slide</title></head>
    <body>

        <object
            code    = "SlideShow.class"
            width   = "250"
            height  = "130"
            >
        </object>
    </body>
</html>
```

```
/**
 * @(#)SlideShow0.java
 *
 * SlideShow0 Applet
 *
 * http://java.sun.com/developer/technicalArticles/Threads/applet/
 *
 * lanciare con:  appletviewer -J-Djava.security.policy=java.policy slides.html
 *                      se URL  "http://developer.java.sun.com:8080/" +
 *                                  "developer/technicalArticles/Threads/applet/"
 *
 * @author 4AI_SIS
 * @version 1.00 2011/6/3
 */

import java.awt.*;
import java.applet.*;
import java.net.*;

public class SlideShow0 extends Applet {
  private Image[] images;
  private String[] text;
  private Label captions;

  private volatile int curFrame;
  private volatile boolean noStopRequested;

  private Thread timerThread;
  private boolean paused;
  private final Object pauseLock = new Object();

  private void printThreadName(String prefix) {
    String name = Thread.currentThread().getName();
    System.out.println(prefix + name);
  }

  public void init() {
    images = new Image[3];
    text = new String[3];
    captions = new Label();
    setLayout(new BorderLayout());
    add(BorderLayout.SOUTH, captions);

    Label name = new Label("by Claude Monet");
    //name.setAlignment(Label.CENTER); allineamento indifferente
    add(BorderLayout.EAST, name);

    URL fig = null;
    try {
      fig = new URL(getCodeBase(),"IMG/");    // immagini in sottocartella IMG con percorso del bytecode
    } catch (java.net.MalformedURLException ex) {
      System.out.println("Bad URL");
      return;
    }

    images[0] = getImage(fig, "bordighera.jpg");
    images[1] = getImage(fig, "etretat.jpg");
```

```
  images[2] = getImage(fig, "leyden.jpg");
  text[0] = "Garden in Bordighera";
  text[1] = "Rock Arch West of Etretat";
  text[2] = "Bulbfield and Windmill near Leyden";
  printThreadName("init is ");
  startThread();
 }

 private void startThread() {
   paused = true;
   noStopRequested = true;

   // Use this inner class to hide the public run method
   Runnable r = new Runnable() {
     public void run() {
       runWork();
     }
   };
   timerThread = new Thread(r, "Timer");
   timerThread.start();
   printThreadName("startThread is ");
}

private void stopThread() {
  noStopRequested = false;
  timerThread.interrupt();
  printThreadName("stopThread is ");
}

private void runWork() { // note that this is private
  printThreadName("run is ");
  curFrame = 0;

  try {
     while ( noStopRequested ) {
     waitWhilePaused();

     curFrame = ( curFrame + 1 ) % images.length;
     repaint();

     Thread.sleep(3000);
    }
  } catch ( InterruptedException x ) {
    // reassert interrupt
   Thread.currentThread().interrupt();
   System.out.println("interrupt and return from run");
  }
}

private void setPaused(boolean newPauseState) {
  synchronized ( pauseLock ) {
    if ( paused != newPauseState ) {
      paused = newPauseState;
      pauseLock.notifyAll();
    }
  }
}
```

```
  private void waitWhilePaused() throws InterruptedException {
    synchronized ( pauseLock ) {
      while ( paused ) {
        pauseLock.wait();
      }
    }
  }

  public void paint(Graphics g) {
    update(g);
    printThreadName("paint is ");
  }

  public void update(Graphics g) {
    g.drawImage(images[curFrame], 0, 0, this);
    captions.setText(text[curFrame]);
    printThreadName("update is ");
  }

  public void start() {
    setPaused(false);
    printThreadName("start is ");
  }

  public void stop() {
    setPaused(true);
    printThreadName("stop is ");
  }

  public void destroy() {
    stopThread();

    for (int i = 0; i < images.length; i++) {
      images[i].flush();
      images[i] = null;
      text[i] = null;
    }

    images = null;
    text = null;
    printThreadName("destroy is ");
  }
}
```



Pagina html:

```
<html>
    <head><title>Mostra Slide</title></head>
    <body>

        <object
            code    = "SlideShow0.class"
            width   = "250"
            height  = "130"
            >
        </object>
    </body>
</html>
```