APPLET

Java si è diffuso storicamente, e trae forza, dal concetto di ***applet***[1] come piccola (almeno all'inizio dei tempi) applicazione da eseguirsi dentro un browser Internet

- *grafica portabile ed eseguibile ovunque*
- *modello di sicurezza "sandbox"*

Una *applet* ("applicazioncina") è una applicazione *non autonoma*, ma pensata per *far parte di una pagina in Internet*

- porta dinamicità alle pagine HTML "statiche"
- viene *eseguita dal browser*, che quindi deve *incorporare un interprete Java* e questo può causare un tempo di visualizzazione più alto

In quanto applicazione *non autonoma*, un'applet:

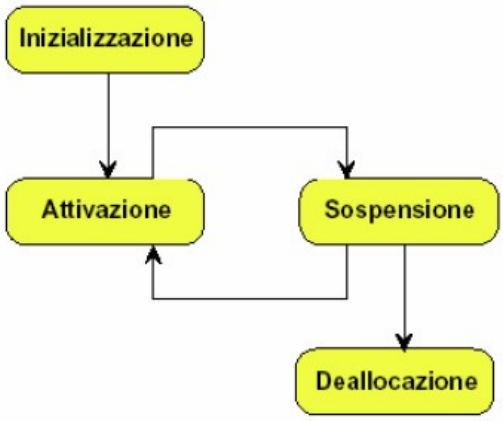
· **non ha un main**, perché la sua vita è dipendente dalla pagina in cui è visualizzata, **eredita** dalla superclasse ed è **public**.

· non deve creare un frame principale, perché usa la finestra del browser che la ospita (in effetti, Applet deriva direttamente da Panel e quindi è essa stessa un pannello)

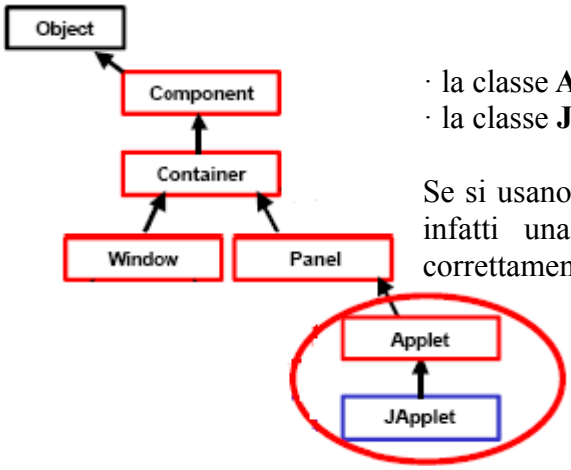
· è organizzata intorno a 4 metodi standard:

- **init()**, che gioca il ruolo del costruttore
- **start()** e **stop()**, chiamati dal browser ogni volta che occorre avviare /fermare l'applet
- **destroy()**, invocato quando il browser viene chiuso.

Diagramma degli stati di un'applet



Esistono *due versioni* di Applet:



· la classe **Applet** dell'AWT standard *(da Java 1.0 in poi)*
· la classe **JApplet** di Swing *(da Java 1.1.6 in poi)*

Se si usano componenti Swing, occorre necessariamente usare JApplet infatti una Applet con componenti Swing non viene disegnata correttamente

[1] Da http://www33.brinkster.com/4binf/dispense/java/JavaApplet.pdf

```
// AppletSaluto.java
import java.awt.Graphics;  // importa solo la necessaria classe Graphics
                           // definita all'interno del package
                           // che implementa l'Interfaccia Utente detta Abstract Windowing Toolkit

public class AppletSaluto extends java.applet.Applet    {     // AppletSaluto è sottoclasse di Applet

        public void paint(Graphics g)  {
              g.drawString("Ciao alla citta'!", 5, 25);    // coordinate dell'angolo in basso
           }                                               // a sinistra del testo
       }
```

Con *file HTML* che fa riferimento al file bytecode dell'applet:

```
<object code="AppletSaluto.class" width="195"height="38"> </object>
```

Si ottiene il seguente effetto con uso di BROWSER (Internet Explorer recenti o FireFox Mozilla)

Ciao alla citta '! *cioè una scritta nella finestra del browser*

(5, 25)

```
// SecondoApplet.java  con passaggio di parametri
import java.awt.*;
import java.applet.*;

public class SecondoApplet extends Applet   {

        public void paint(Graphics g) {
              String nome = getParameter ("Nome");
              g.drawString("Ciao " + nome, 0, 50);
           }
 }
```

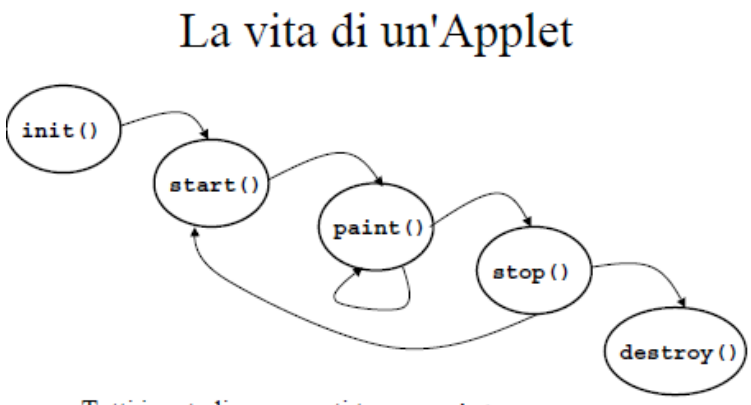
Con *file HTML*

```
<html>
   <head><title> Saluto personalizzato</title></head>
   <body>
        <object code="SecondoApplet.class" width="195"height="38">
            <param name="Nome" value="Nome_proprio">
        </object>
   </body>
</html>
```

Si ottiene il seguente effetto con uso di BROWSER (Internet Explorer):

Ciao *Nome_proprio*

cioè un saluto personalizzato nella finestra del browser



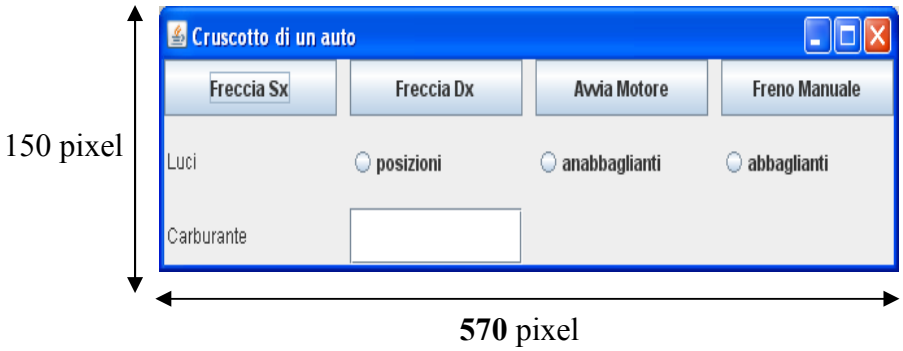
Tutti i metodi sono vuoti tranne `paint` che viene ereditato da `Component`

Esercizio:

Creare un'applet che disegni il cruscotto di un'automobile in una finestra separata.

Nel cruscotto devono apparire

- 4 bottoni con testo: Freccia Sx, Freccia Dx, Freno Manuale, Avvia Motore
- un'etichetta con testo Luci con associate 3 caselle opzione (esclusiva) con testo posizioni, anabbaglianti, abbaglianti , tutte non attivate
- un'etichetta carburante
- ed un JTextField largo 30 pixel



Soluzione:

```
// esempio di JApplet con uso contenitori e componenti GUI
import java.awt.*;
import javax.swing.*;
import java.applet.*;

public class Cruscotto extends JApplet  {

    public void init() {

        JFrame f = new JFrame ("Cruscotto di un auto");
        f.setSize (570, 150);
        f.setLocation (100,100);
        f.setResizable(true);    // con mouse ridimensionabile

         Container  c = f.getContentPane();  // recupera il "muro grezzo" cioè il riquadro senza barra menù

        ButtonGroup luci = new ButtonGroup();
        JRadioButton rb1 = new JRadioButton("posizioni", false);
        JRadioButton rb2 = new JRadioButton("anabbaglianti", false);
        JRadioButton rb3 = new JRadioButton("abbaglianti", false);
        luci.add(rb1);
        luci.add(rb2);
        luci.add(rb3);   // selezione alternativa nel gruppo
        JButton FrecciaSx = new JButton ("Freccia Sx");
        JButton FrecciaDx = new JButton ("Freccia Dx");
        JButton FrenoManuale = new JButton ("Freno Manuale");
        JButton AvviaMotore = new JButton ("Avvia Motore");

        JTextField carburante = new JTextField (30);

        c.setLayout (new GridLayout (3,4,10,10)); // 3 righe, 4 colonne e spaziatura
        c.add(FrecciaSx);
        c.add(FrecciaDx);
        c.add(AvviaMotore);
        c.add(FrenoManuale);
```

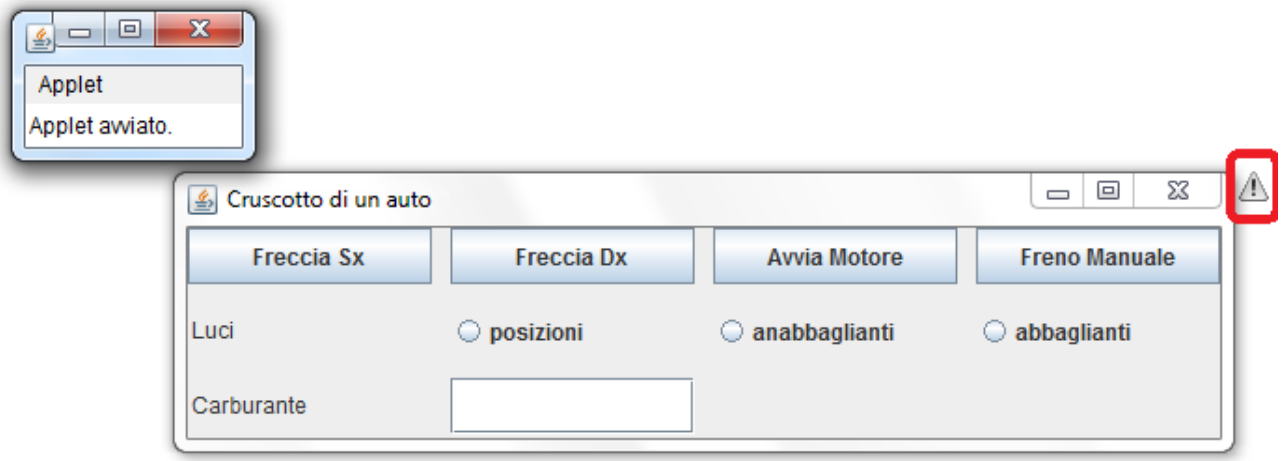
```
        c.add(new Label("Luci"));
        c.add(rb1);
        c.add(rb2);
        c.add(rb3);
        c.add(new Label("Carburante"));
        c.add(carburante);

        f.setVisible(true);
      }
}
```

Nel codice HTML: <**object** code ="Cruscotto.class" **width= "0" height= "0"**></object>

Con **dimensioni obbligatorie** pur se nulle. In ambiente **JCreator**, il visualizzatore della *Java Applet Window* inserisce messaggio di Applet avviato o non avviato.

NB: Un' applet dunque può aprire un'altra finestra *ma compare automaticamente un avviso* anche solo come *piccola icona di allerta* in alto a sinistra della nuova finestra che ricorda come la ***Java Applet Window*** non sia mai completamente invisibile.

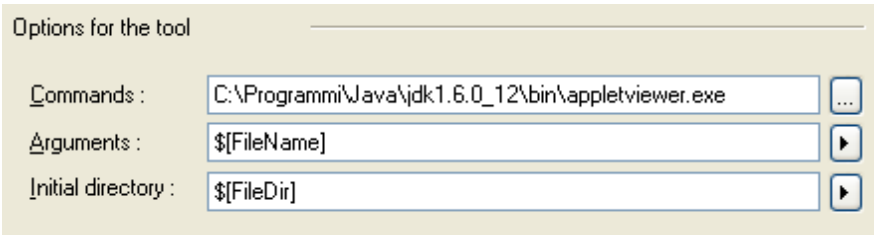


nb: per eseguire la **pagina html** con **opzione Run** in JCreator, salvare tale pagina web nella stessa cartella del bytecode (**classes**). Si noti che,scegliendo come progetto **Basic Java Applet**
si crea nella sottocartella **classes** la seguente pagina


```
<html>
    <head> </head>
    <body bgcolor="000000">
    <center><applet code  = "NomeClasse.class"
                    width= "500"
                     height= "300">
            </applet>
    </center>
    </body>
</html>
```

con uso di **tag obsoleto** <applet></applet>

nb: per eseguire la pagina html con programma **appletviewer**, inserito come ***tools*** in JCreator, impostare come opzioni:



APPLET e SICUREZZA

Un'applet non può fare tutto quello che fa una applicazione. Poiché può essere scaricata dalla rete, *sarebbe troppo pericoloso* permettere a un'applet di fare qualunque cosa ed è *costretta* a rispettare un ben preciso *modello di sicurezza (**"sandbox"**)*

– è eseguita in una "scatola" da cui non può uscire
– non può contaminare (o spiare) i dati del computer dell'utente

Un'applet di norma *non può*:

- accedere al file system locale (neppure per leggere un file)
- eseguire un altro programma
- ottenere informazioni sull'utente
- connettersi via rete a un computer *diverso da quello da cui è stata scaricata*[2]
- caricare la libreria Java, chiamare System.exit()

In molte situazioni, questi vincoli sono *troppo rigidi* e rischierebbero di *rendere impossibile* la costruzioni di applet *utili.*

APPLET FIRMATE

Attraverso tecnologie di cifratura, un'applet può essere *firmata*, ossia a essa può essere allegato un *certificato* che ne garantisce l'origine. A tali applet firmate, cui si attribuisce maggiore fiducia, *l'utente può consentire di svolgere alcune o tutte le operazioni sottoposte a vincolo.*

Ogni browser può essere configurato per gestire le applet firmate.

POLITICHE DI SICUREZZA

A partire da Java 2, l'utente può decidere *caso per caso* quali politiche di sicurezza applicare, con una *granularità molto fine*

Esiste il concetto di *policy file*, che elenca le politiche locali e si può stabilire che *una certa applet*, proveniente da *un ben preciso sito*, ha diritti particolari. Un tale file può essere fornito da chi sviluppa l'applet, o modificato dall'utente con lo strumento *PolicyTool.*

[2] Per questo motivo si preferisce lavorare con applicazioni quando ci si connette a Internet e se ne usano le risorse.