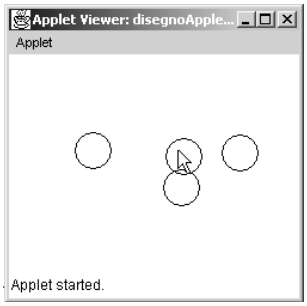
Esempi di programmazione

Applet che disegna cerchi

- Costruire un'applet **disegnoApplet** che ad ogni pressione del mouse disegna un cerchio



`disegnoApplet`

1. Deve usare la classe Applet? NO
2. Deve estendere la classe Applet? SÌ
3. Deve importare java.applet.*? SÌ
4. Deve essere pubblica? SÌ
5. Deve avere il metodo main? NO
6. Deve avere il metodo init? S/N

disegnoApplet

```
import java.applet.*;
public class disegnoApplet
             extends Applet {
}
```

Di cosa abbiamo bisogno

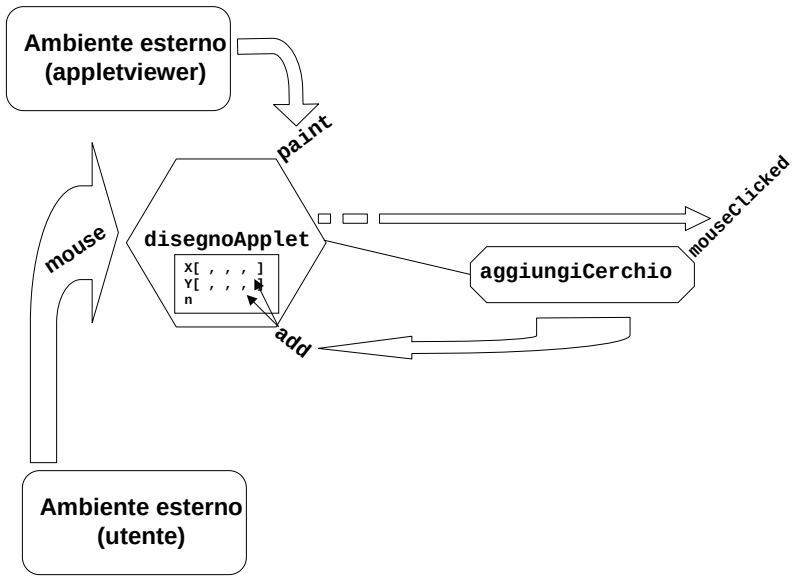
1. Panel? NO
2. Canvas? S/N
3. Button? NO
4. ActionListener? NO
5. MouseListener? SÌ
6. MouseMotionListener? NO

```
/*<applet code="disegnoApplet.class" width="400"
height="400"></applet>*/
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class disegnoApplet extends Applet {
  public void init() {
    this.addMouseListener(new aggiungiCerchio());
  }
}

class aggiungiCerchio extends MouseAdapter  {
  public void mouseClicked(MouseEvent e) {}
}
```

Come fare??

- L'unico modo che conosciamo per disegnare un cerchio è usare il metodo **`drawOval`** della classe **`Graphics`**
- Per far disegnare sull'applet sovrascriviamo il metodo **`paint`**
- L'applet deve disegnare un cerchio in ogni punto in cui è stato cliccato
  - Deve "tenere a mente" i punti in cui si clicca
  - Ogni volta che si fa click occorre aggiungere un nuovo punto e richiamare **`paint`**

```
public class disegnoApplet extends Applet {
   private final int MAX_ELEM = 10;  //massimo numero di cerchi
   private final int R = 15;         //raggio
   int[] x = new int[MAX_ELEM];      //coordinate dei cerchi da disegnare
   int[] y = new int[MAX_ELEM];
   int n = 0;                        //numero cerchi da disegnare
...
   public void paint(Graphics abc) {
      for (int i = 0; i<n; i++)
         abc.drawOval(x[i]-this.R,y[i]-this.R,this.R*2,this.R*2);
   }
   void add(int x, int y) {
      if (this.n < this.MAX_ELEM) {
         this.x[n] = x; this.y[n] = y; this.n++;
      }
   }
}
class aggiungiCerchio extends MouseAdapter  {
   public void mouseClicked(MouseEvent e) {
      disegnoApplet dA = (disegnoApplet)e.getSource();
      dA.add(e.getX(),e.getY()); dA.repaint();
   }
}
```



Memorizzare una coppia di interi

- Per memorizzare una coppia possiamo usare 2 variabili **`int`**
- Possiamo definire una classe (meglio!)

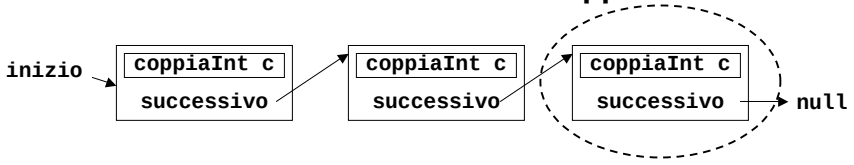
```
class coppiaInt {
  int x;     int y;
  coppiaInt(int x, int y) {
   this.x = x;
   this.y = y;
  }
}
```

Memorizzare 2 coppie di interi

- Possiamo usare 4 variabili **`int`**
- Possiamo usare 2 vettori di 2 **`int`**
  - ```
    int[] x = new int[2];
    int[] y = new int[2];
    x[0] = 2; y[0] = 3;
    System.out.println(x[0]);
    ```
- Possiamo usare 1 vettore di coppie di interi **`coppiaInt`**
  - ```
    coppiaInt[] c = new coppiaInt[2];
    c[0] = new coppiaInt(2,3);
    System.out.println(c[0].x);
    ```

Per memorizzare "tante" coppie di interi?

- Possiamo usare "tante" variabili **`int`**? Non sappiamo quante.
- Possiamo usare 2 vettori di **`int`**? Non sappiamo quanto grandi.
- Possiamo usare un vettore di coppie di interi **`coppiaInt`**? Non sappiamo quanto grande.
- Creiamo una nuova classe **`coppiaIntPerLista`**



coppiaIntPerLista

```
class coppiaIntPerLista {
 coppiaInt c;
 coppiaIntPerLista succ;

 coppiaIntPerLista(coppiaInt c) {
   this.c = c;
   this.succ = null;
 }
}
```

listaCoppie

```
class listaCoppie {
  coppiaIntPerLista inizio;
  listaCoppie(coppiaIntPerLista prima) {
     this.inizio = prima;
  }
  public void aggiungi(coppiaIntPerLista c) {
     c.succ = this.inizio;
     this.inizio = c;
  }
  public coppiaIntPerLista primaCoppia() {
     return inizio;
  }
  public void togli() {this.inizio = this.inizio.succ;}
}
```

Memorizzare 4 coppie di interi

```
coppiaInt c = new coppiaInt(2,3);
coppiaIntPerLista cpl = new coppiaIntPerLista(c);
listaCoppie lc = new listaCoppie(cpl);

lc.aggiungi(new coppiaIntPerLista(new coppiaInt(1,2)));
lc.aggiungi(new coppiaIntPerLista(new coppiaInt(1,7)));
lc.aggiungi(new coppiaIntPerLista(new coppiaInt(9,5)));
System.out.println(lc.primaCoppia().c.x);
                            //stampa 9
System.out.println(lc.primaCoppia().succ.c.y);
                            //stampa 7
```

Memorizzare 100 coppie di interi?

- Sempre allo stesso modo con **`listaCoppie`**
- In realtà la struttura dati assomiglia, come funzionamento, più ad una…
- Attenzione! Il metodo **`togli`** può dare errore. Quando?