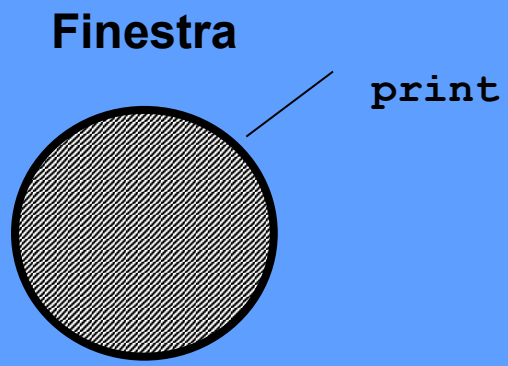
UN PROGETTO A COMPONENTI

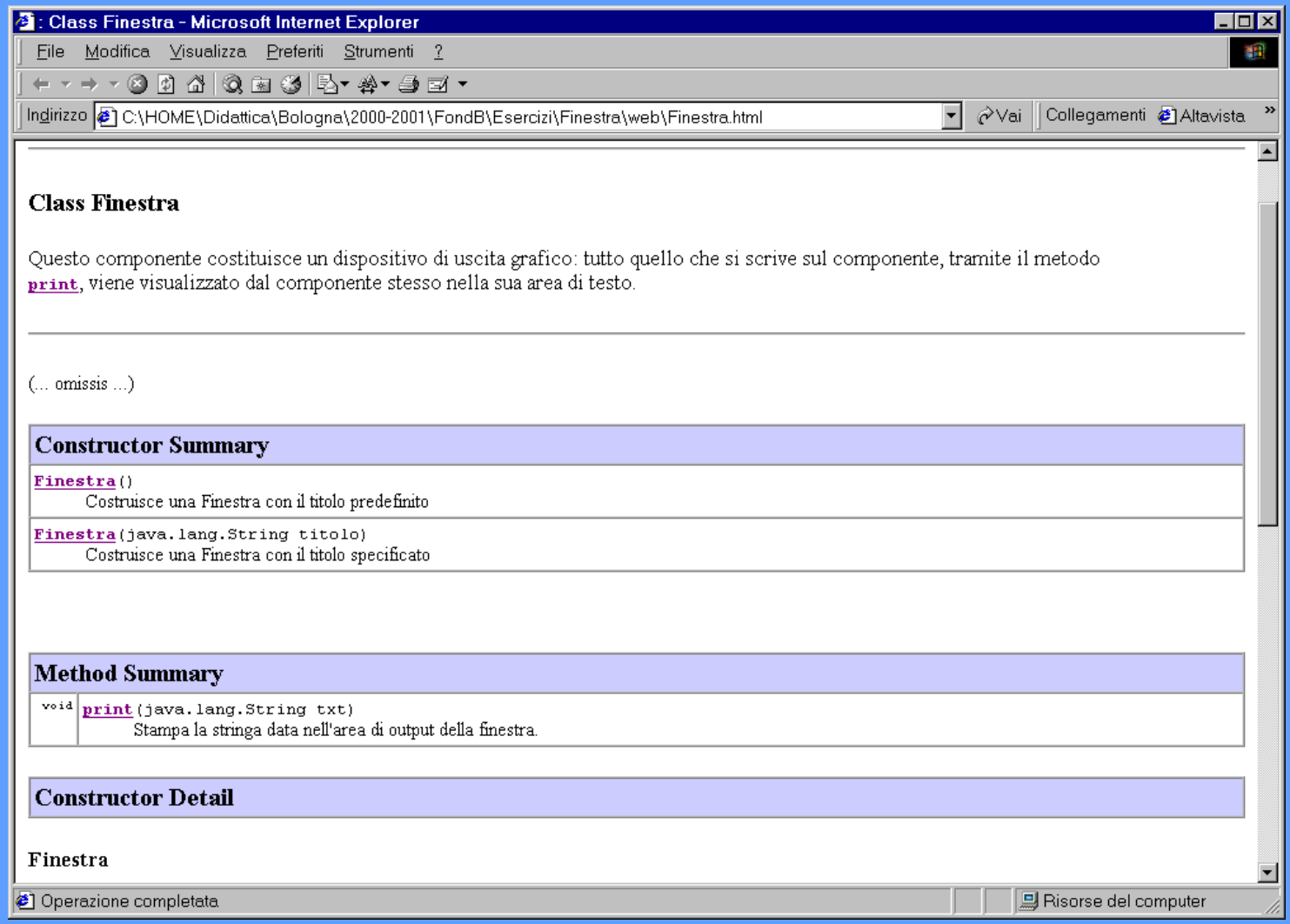
- **Noi non conosciamo alcunché di grafica, quindi non siamo in grado (per ora) di progettare componenti che ne facciano uso.**
- **Però, possiamo *usare* un componente grafico *già pronto…***
- **… e se occorre *specializzarlo.***

IL COMPONENTE `Finestra`

- **Ci piacerebbe poter scrivere *non sulla console*, ma su una vera *finestra grafica***
- **Il componente *`Finestra`* svolge proprio questo compito**
  - **non serve sapere *come* è *fatto* (non avete il sorgente Java…)**
  - **avete solo il file `.class`**
  - **e la documentazione generata dall'autore con Javadoc.**



Finestra: MANUALE D'USO



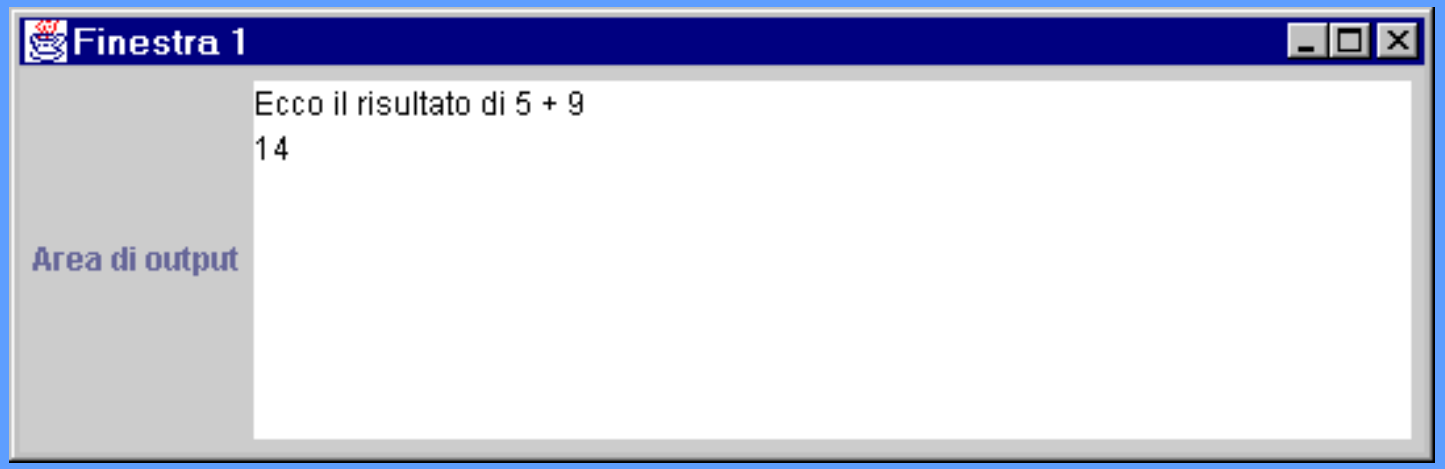
`Finestra`: MANUALE D'USO

Per usare *`Finestra`* bisogna:

- **costruire l'oggetto**
  - **il costruttore di default crea una finestra con titolo predefinito**
  - **il secondo costruttore, con parametro stringa, permette di specificare il titolo desiderato**
- **scriverci sopra mediante il metodo `print`**
  - **come `System.out`, anche `Finestra` permette solo di *aggiungere nuove scritte*, non di cancellare le precedenti.**

UN ESEMPIO D'USO

```
public class Prova1 {
 public static void main(String args[]){
   Finestra f = new Finestra("Finestra 1");
   f.print("Ecco il risultato di 5 + 9\n");
   f.print( "" + (5+9) ); // attenzione al +
 }
}
```



RIFLESSIONI

- **Non è stato necessario avere il sorgente: è bastato il file `.class`**
  - **magari scaricato dalla rete**
  - **magari sviluppato perfino su *diversa piattaforma***
- **Non è stato necessario ricompilare nulla, né linkare alcunché**
  - **il collegamento della classi, in Java, è dinamico**
- **Si sono così potute sfruttare funzionalità che *non saremmo stati in grado di costruirci da soli***
  - **l'architettura di Java favorisce e supporta il *riuso effettivo* dei componenti**

OLTRE `Finestra`

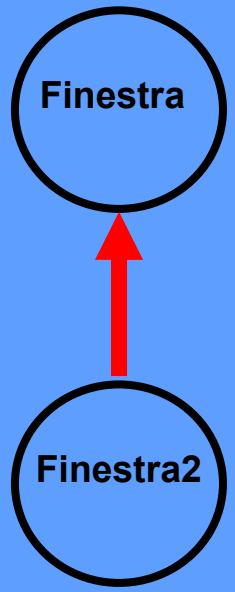
E se il componente `Finestra` *non basta?*

- **`Finestra` fornisce solo un metodo `print` di base**
  - **non va a capo**
  - **stampa solo stringhe (non interi, né float, etc)**
- **Potrebbe farci comodo una *versione arricchita* di `Finestra`, capace anche di**
  - **stampare andando a capo (metodo `println`)**
  - **stampare anche `int`**

Come fare?

OLTRE `Finestra`

- **Non abbiamo il codice di `Finestra`, quindi *non possiamo modificare il sorgente Java***
  - **d'altronde, non vogliamo neanche farlo!**
- **Possiamo però sfruttare l'ereditarietà per definire una nuova classe `Finestra2` che estenda `Finestra`**
  - **definendo un nuovo metodo `println` che vada automaticamente a capo dopo la stampa**
  - **definendo un nuovo metodo `print(int)` che accetti e stampi direttamente un intero.**



LA CLASSE `Finestra2`

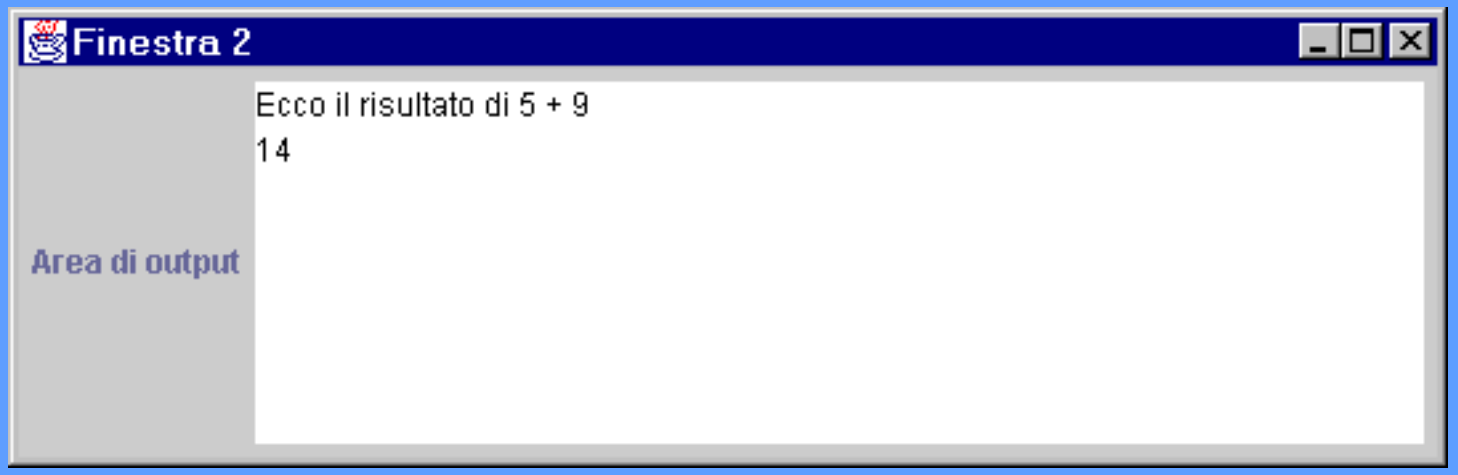
```
public class Finestra2 extends Finestra {
 public Finestra2() { super(); }
 public Finestra2(String titolo) { super(titolo); }
 public void println(String txt){ print(txt + "\n"); }
 public void print(int x){ print("" + x); }
}
```

NOTE

- **il costruttore di `Finestra2` può invocare qualsiasi costruttore di `Finestra` ritenga opportuno e/o cambiare la frase da visualizzare (sfruttando `super`)**
- **i due nuovi metodi di `Finestra2` si appoggiano al metodo `print` di `Finestra`, ereditato da `Finestra2`**

UN ESEMPIO D'USO

```
public class Prova2 {
 public static void main(String args[]){
  Finestra2 f = new Finestra2("Finestra 2");
  f.println("Ecco il risultato di 5 + 9");
  f.print(5+9); // ora il parametro è un int
 }
}
```



UN ESEMPIO D'USO

```
public class Prova2 {
 public static void main(String args[]){
  Finestra2 f = new Finestra2("Finestra 2");
  f.println("Ecco il risultato di 5 + 9");
  f.print(5+9); // ora il parametro è un int
 }
}
```

Domanda: sarebbe stato lecito scrivere

```
Finestra f = new Finestra2(...)   ?
```

Funzionerebbe? Perché?

ULTERIORI RIFLESSIONI

- **Non è stato necessario avere il sorgente *nep-pure per specializzare* quel componente**
  - **anche qui è bastato il file `.class`**
  - **… e un po' di documentazione**
- **È stato possibile perfino adattare un componente *di cui non conosciamo il funzionamento interno e che non avremmo saputo costruire***
  - **l'ereditarietà è un mezzo estremamente potente per *riusare* e *adattare* componenti *anche fatti da altri***
  - **si toccano con mano i vantaggi dell'incapsulamento**

DOPO L'OUTPUT...

… L'INPUT

Sul sito è disponibile il componente `InputField` con relativa documentazione.

- **Esperimenti di funzionamento**
- **Estensioni ?**