Programmazione in Java
Parte II

Lezione 6

Dott. Marco Faella

Il modificatore static

- Il modificatore static si può applicare a campi (attributi) e metodi
- In entrambi i casi, esso indica che quel membro appartiene alla classe, e non alle singole istanze della classe

- Ad esempio, un campo statico è una variabile di cui esiste una sola copia, legata alla classe a cui appartiene, e non alle singole istanze
- Un campo statico esiste in memoria anche se la sua classe non è mai stata istanziata
- Per un campo statico valgono le stesse regole di visibilità che si applicano ai campi non statici

- Quindi, i campi si dividono in campi istanza e campi di classe

- Una combinazione utile è costituita dai modificatori final e static
  - applicati ad un campo, essi danno luogo ad una *costante di classe*

- Esempio:

```
class A {
   public static int n;
   public static void f() { ... }
}
```

- Ad un membro statico si può accedere usando il nome della classe, come in

```
A.n = 7;
A.f();
```

- Se ci si trova all'interno della classe stessa, non è necessario utilizzare il nome della classe

Memory layout di un campo statico

Esempio:

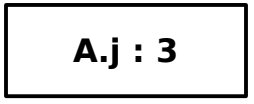
```
class A {
     private int i;
     public static int j = 10;

     public A(int n) {
          i = n;
          j = n;
     }
}

...

main() {
   A.j = 3;
   A a = new A(4);
   A b = new A(5);
}
```

Memory layout:



Memory layout di un campo statico

Esempio:

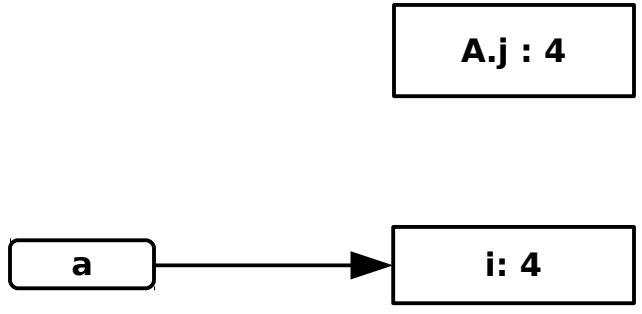
```
class A {
     private int i;
     public static int j = 10;

     public A(int n) {
          i = n;
          j = n;
     }
}

...

main() {
   A.j = 3;
   A a = new A(4);
   A b = new A(5);
}
```

Memory layout:



Memory layout di un campo statico

Esempio:

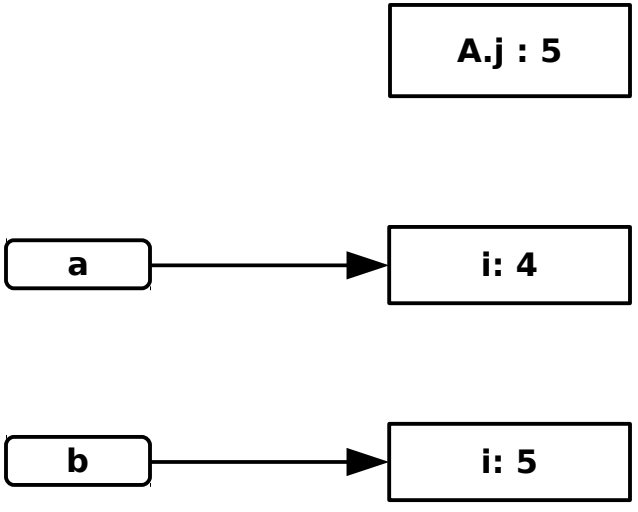
```
class A {
     private int i;
     public static int j = 10;

     public A(int n) {
          i = n;
          j = n;
     }
}

...

main() {
   A.j = 3;
   A a = new A(4);
   A b = new A(5);
}
```

Memory layout:



Metodi statici e riferimento `this`

- Il riferimento `this` punta sempre all'oggetto corrente
  - cioè, all'oggetto sul quale è stato chiamato il metodo corrente
- `this` è presente in tutti i metodi non statici e nei costruttori

- Un metodo statico si riferisce all'intera classe e non ad un oggetto in particolare
- Quindi, esso non possiede il riferimento `this`
- Di conseguenza, esso non ha nemmeno accesso agli attributi non statici della classe

- Esercizi

- Un'interfaccia è un tipo particolare di classe, che ha tutti i metodi pubblici e astratti
- Inoltre, gli unici campi che un'interfaccia può contenere sono le costanti di classe
    - ovvero, campi statici e final

- La sintassi è la seguente:

```
interface Test {
   ...
}
```

- Tutti i metodi di un'interfaccia sono implicitamente public e abstract
    - indicare questi due modificatori esplicitamente è facoltativo
- Tutti i campi di un'interfaccia sono implicitamente public, final e abstract
    - indicare questi tre modificatori esplicitamente è facoltativo

- Naturalmente, non è possibile istanziare un'interfaccia

- Un'interfaccia non contiene codice, è solo un contratto che le sottoclassi si impegnano a rispettare
- Una classe può “estendere” un'interfaccia con la parola chiave **implements**
  - difatti, si preferisce dire che la classe *implementa* o *realizza* l'interfaccia
  - se la classe in questione non è astratta, è obbligata ad implementare tutti i metodi presenti nell'interfaccia

- Esempio:

```
class A implements Test {
   ...
}
```

Ereditarietà singola e multipla

- Un'interfaccia è un costrutto più restrittivo di una classe astratta
- Dunque, perché preferirla ad una classe astratta?

- **Una classe può estendere una sola altra classe, ma può implementare diverse interfacce**

- Quindi, se la progettazione ci porta ad avere una classe con tutti metodi astratti, è opportuno convertirla in interfaccia

- Esercizio: aggiungere cerchio