Corso di Programmazione
Anno Accademico 2013–2014
Prof. Giovanni Pighizzini

Note ed esercizi aggiuntivi

11. Gerarchia delle classi e gerarchia dei tipi

Esercizio 11.1
Si consideri una classe `Sopra` che, oltre a un costruttore privo di argomenti, offre i seguenti metodi:

- `public int doppio(int x)`
  Restituisce il doppio del valore ricevuto tramite l'argomento.
- `public int uno()`
  Restituisce `1`.
- `public int m(int x)`
  Restituisce un valore uguale a quello ricevuto tramite l'argomento.

Si consideri poi una classe `Sotto`, che estende `Sopra`. Anche questa classe fornisce un costruttore privo di argomenti. Inoltre la classe prevede i seguenti metodi, oltre a quelli ereditati da `Sopra`:

- `public int zero()`
  Restituisce `0`.
- `public int m(int x)`
  Restituisce il triplo del valore ricevuto tramite l'argomento.

Si consideri, infine, la seguente classe di prova:

```
import prog.io.*;

class Prova {
  public static void main(String[] args) {
    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    int v = in.readInt();
    Sopra s;
    int x = v;

    if (v > 10)
      s = new Sopra();
    else
      s = new Sotto();
    out.println(s.doppio(v));
```

```
    out.println(s.m(v));
    if (s instanceof Sotto) {
      Sotto st = (Sotto) s;
      x = x + st.zero();
    } else
      x = x + s.uno();
    out.println(x);
  }
}
```

1. Scrivete l'output prodotto dal programma nell'ipotesi che l'utente inserisca il valore `8`.

2. Scrivete l'output prodotto dal programma nell'ipotesi che l'utente inserisca il valore `11`.

Esercizio 11.2
Si considerino quattro classi `Alfa`, `Beta`, `Gamma` e `Delta` tali che:

- `Beta` e `Gamma` estendono `Alfa`, `Delta` estende `Gamma`;
- l'interfaccia `In` è implementata da `Gamma` ma non da `Beta`;
- la classe `Alfa` è astratta, le altre classi sono concrete;
- ognuna delle classi possiede un costruttore privo di argomenti;
- in ognuna delle classi è definito, tra gli altri, un metodo `public String toString()` che restituisce una stringa uguale al nome della classe stessa. Ad esempio, il metodo `toString` definito nella classe `Beta` restituisce la stringa `"Beta"`.

1. Per ciascuna delle seguenti affermazioni, indicate se è vera o falsa:

   (a) I metodi astratti di `Alfa` sono implementati in `Gamma`

   (b) `Delta` è una sottoclasse di `Alfa`

   (c) `Beta` è una sottoclasse di `Object`

   (d) `In` è un supertipo di `Delta`

   (e) `Gamma` è una sottoclasse di `In`

   (f) I metodi astratti di `In` sono implementati in `Gamma`

   (g) `Gamma` è una sottotipo di `Object`

   (h) `In` è un sottotipo di `Alfa`

   (i) `In` è un supertipo di `Alfa`

   (j) `Object` è un supertipo di `Beta`

   (k) `Delta` è una sottoclasse di `Beta`

   (l) `In` è un supertipo di `Alfa`

   (m) `In` possiede un costruttore

   (n) `Alfa` è un tipo riferimento

2. Si considerino le seguenti dichiarazioni di variabile:

```
Alfa p;  Gamma q; Beta t;  Delta r; In s;
```

   Per ciascuno dei seguenti assegnamenti, indicate se è compilato correttamente oppure no:

   (a) `t = (Beta) t`
   (b) `s = new In()`
   (c) `s = new Beta()`
   (d) `t = (Alfa) t`
   (e) `p = new Alfa()`
   (f) `s = new Delta()`
   (g) `q = (Gamma) r`
   (h) `r = q`
   (i) `r = (Delta) q`
   (j) `s = r`
   (k) `s = q`
   (l) `p = (Delta) s`
   (m) `p = t`
   (n) `t = p`

3. Indicate l'output prodotto dal seguente metodo `main`:

```
public static void main(String[] args) {
  Gamma w = new Delta();
  Alfa x = w;
  In y = new Gamma();

  System.out.println(x.toString());
  System.out.println(y.toString());
  System.out.println(w.toString());
}
```