

Note ed esercizi aggiuntivi

13. Struttura delle classi, ereditarietà e costruttori

Esempio.

```
import prog.utili.Importo;

public class Alfa extends Importo {
    private static int x = 2;
    private int y = 5;

    public Alfa(int i) {
        super(i, i + x);
        x++;
        y = y + i;
    }

    public int getCent() {
        return getEuro() * 100 + super.getCent() + y;
    }

    public static int getStatico() {
        return x;
    }
}
```

Note

- Ogni istanza (o oggetto) della classe **Alfa** è formata da un'istanza della classe **Importo** e dal campo **y** definito in **Alfa**, che viene inizializzato a 5 al momento della creazione dell'oggetto.
- Il campo **x** è *statico* e pertanto appartiene alla classe e non ai singoli oggetti. Tale campo viene creato al momento del caricamento della classe.
- Il costruttore di **Alfa** richiama, nella prima istruzione, il costruttore con due parametri della superclasse **Importo** per predisporre la parte di oggetto ereditata da **Importo**. Successivamente il costruttore modifica il campo statico **x** della classe **Alfa** e il campo **y** dell'oggetto appena creato.
- La classe **Alfa**, estendendo **Importo**, ne eredita i metodi. Tra questi vi sono **getEuro** e **getCent**. Il metodo **getCent** è sovrascritto nella classe **Alfa**. Dunque il nuovo metodo **getCent** sostituisce per gli oggetti della classe **Alfa** quello ereditato da **Importo**.

- Quando una chiamata di metodo non è preceduta da riferimenti, è sottinteso `this`. Pertanto la chiamata `getEuro()` all'interno del metodo `getCent` di `Alfa` equivale a `this.getEuro()`. Il metodo chiamato è dunque quello che `Alfa` eredita da `Importo`.
- La chiamata `super.getCent()` all'interno del metodo `getCent` di `Alfa` invoca il metodo `getCent` della superclasse `Importo` (che è stato riscritto in `Alfa`). Se non si utilizzasse il riferimento `super`, sarebbe sottinteso `this`. In tal modo, il metodo `getCent` di `Alfa` si richiamerebbe ricorsivamente, occupando tutta la memoria disponibile e provocando dunque un errore in esecuzione (gli aspetti relativi alle chiamate ricorsive e all'organizzazione della memoria saranno presentati nelle lezioni successive).

Esercizio 13.1

Considerate la seguente applicazione che utilizza la classe `Alfa` dell'esempio precedente.

```
class Prova {
    public static void main(String[] args) {
        System.out.println(Alfa.getStatico());
        Importo imp = new Alfa(7);
        System.out.println(imp.getCent());
        imp = new Importo(100, 30);
        System.out.println(imp.getCent());
        System.out.println(Alfa.getStatico());
        imp = new Alfa(4);
        System.out.println(imp.getCent());
    }
}
```

Simulate l'esecuzione del metodo `main` disegnando l'evoluzione dei vari dati utilizzati (oggetti, campi statici, variabili locali, parametri). Quali risultati vengono visualizzati? Dopo avere risposto, fate eseguire l'applicazione sul vostro computer e confrontate i risultati ottenuti con le risposte che avete fornito.

Esercizio 13.2

Considerate la seguente classe `Alfa`

```
import prog.utili.Intero;

public class Alfa extends Intero {
    private String x = "";
    private static int y = 3;

    public Alfa(String z) {
        this(z.length());
        x = z;
    }

    public Alfa(int i) {
        super(i);
        y = 2 * y + i;
    }

    public String toString() {
        return super.toString() + x;
    }
}
```

```
}

    public static int getStatico() {
        return y;
    }
}
```

Indicate quali campi appartengono alle singole istanze e quali alla classe. Indicate poi la struttura delle istanze della classe. Scrivete poi l'output prodotto dal metodo `main` della seguente applicazione che utilizza la classe `Alfa` (per i dettagli relativi al contratto dei costruttori e dei metodi della classe `Intero` consultate la documentazione):

```
class Prova {
    public static void main(String[] args) {
        System.out.println(Alfa.getStatico());
        Alfa a = new Alfa("cane");
        System.out.println(a.toString().length());
        a = new Alfa(7);
        System.out.println(a.toString().length());
        System.out.println(Alfa.getStatico());
    }
}
```

Esempio.

```
public class K {
    private int x;
    private int y = 2;
    private static int z = 5;

    public void inc() {
        x++;
        y++;
        z++;
    }

    public String toString() {
        return "x == " + x + ", y == " + y;
    }

    public static int getZ() {
        return z;
    }
}
```

Note

- Ogni oggetto della classe `K` possiede due campi `x` e `y`, entrambi di tipo `int`. Il campo `z`, essendo statico, appartiene invece alla classe.
- Poiché non è indicato alcun costruttore, il compilatore aggiunge un costruttore privo di argomenti che inizializza i campi ai valori di default o ai valori indicati insieme alla dichiarazione. In questo caso il campo `x` viene inizializzato a 0, mentre il campo `y` a 5. Il costruttore privo di

argomenti aggiunto dal compilatore, come prima operazione, richiama sempre il costruttore privo di argomenti della superclasse, in questo caso `Object`, per predisporre la parte ereditata (nel caso di `Object` la parte ereditata è sostanzialmente una scatola vuota, pertanto in questo caso la chiamata al costruttore della superclasse appare irrilevante).

```
public class H extends K {
    private int w = 4;

    public void inc() {
        super.inc();
        w++;
    }

    public String toString() {
        return super.toString() + ", w == " + w;
    }
}
```

Note

- Ogni oggetto della classe `H` possiede un campo `w`, oltre alla parte ereditata dalla superclasse `K`.
- Poiché non è indicato alcun costruttore, il compilatore aggiunge un costruttore privo di argomenti. Il costruttore chiama prima di tutto quello della superclasse `K` che predispone la parte di oggetto ereditata come indicato sopra. Successivamente assegna a `w` il valore 4.

Esercizio 13.3

Considerate la seguente applicazione che utilizza le classi dell'esempio precedente.

```
class ProvaH {
    public static void main(String[] a) {
        H r = new H();
        System.out.println(r.toString());
        r.inc();
        System.out.println(r.toString());
        H t = new H();
        System.out.println(r.toString());
        t.inc();
        System.out.println(t.toString());
        System.out.println(K.getZ());
    }
}
```

Simulate l'esecuzione del metodo `main` disegnando l'evoluzione dei vari dati utilizzati (oggetti, campi statici, variabili locali, parametri). Quali risultati vengono visualizzati? Dopo avere risposto, fate eseguire l'applicazione sul vostro computer e confrontate i risultati ottenuti con le risposte che avete fornito.