Le classi in java

Un semplice programma java, formato da una sola classe, assume la seguente struttura:

```
class Domanda
{
   public static void main(String args[])
   {
     System.out.println(“Quanti anni hai?”);
   }
}
```

La classe dichiarata sopra contiene solo il metodo main. Il metodo main viene dichiarato usando le parole chiavi **public, static e void** che specificano alcune proprietà del metodo:

- **public** indica che il metodo è pubblico ed è visibile;
- **void** indica che non ci sono valori di ritorno;
- **static** indica che il metodo è associato alla classe e non può essere richiamato dai singoli oggetti della classe.

Dopo il nome del metodo, tra parentesi, sono indicati i parametri. Il metodo main possiede come parametro un array di stringhe (indicato con **args[]**) che corrisponde ai parametri passati alla riga di comando quando viene eseguita l’applicazione.

Nella classe domanda viene utilizzato il metodo main che è formato da una istruzione singola, precisamente una chiamata del metodo println(). Più precisamente nella OOP, si può dire che viene inviato all’oggetto **System.out** il messaggio **println**. L’oggetto System.out indica lo standard output a video.

Livelli di visibilità

Per specificare alcune caratteristiche delle classi, tra cui il livello di visibilità, bisogna inserire nella dichiarazione della classe una delle seguenti parole chiave **(modificatori):**

- **public**
- **abstract**
- **final**

Se una classe viene dichiarata **public**, allora, può essere usata da tutte le classi, anche fuori dalla libreria dove è stata definita.

```
public class Domanda
{
   public static void main(String args[])
   {
     System.out.println(“Quanti anni hai?”);
   }
}
```

Una classe dichiarata **abstract**, rappresenta una classe astratta, cioè una classe incompleta in cui alcuni metodi sono a loro volta astratti. Una classe definita astratta non può essere istanziata. Deve essere usata per creare le sottoclassi in cui vengono implementati i metodi astratti.

```
abstact class Domanda
{
}
```

Una classe dichiarata **final** non può originare sottoclassi.

```
Final class Domanda
{
}
```

Dichiarazione degli attributi

Gli attributi corrispondono alle variabili dichiarate nel blocco di una classe. Vengono anche chiamati **variabili di istanza** perché ogni oggetto (istanza di una classe) possiede le proprie variabili nelle quali memorizzare il valore dei propri attributi.

I seguenti esempi mostrano dichiarazioni di attributi con visibilità diverse:

```
private int a;
public int b;
protected int c;
int d;
```

Nella dichiarazione di un attributo si può aggiungere un livello di visibilità. I livelli di visibilità stabiliscono se l'attributo è accessibile da altre classi, cioè se dall'esterno della classe che lo contiene si può leggere o modificare il suo valore.
I livelli di visibilità possono essere:

- **public**
- **private**
- **protected**

Con la visibilità **public** l'attributo è accessibile da qualsiasi altra classe.

Al contrario, l'uso di **private** permette di nascondere l'attributo all'interno della classe. Non può essere visto da nessun'altra classe. In questo modo solo la classe che lo contiene può decidere di modificarne il valore.

Un attributo dichiarato come **protected** è visto all'esterno solo dalle classi che appartengono alla stessa libreria oppure alle sottoclassi della classe in cui è stato dichiarato l'attributo.

Se non viene specificato nessun livello di visibilità, l'attributo è visibile solo dalle classi che appartengono alla stessa libreria.

Information hiding

Se si vuole applicare il concetto di **Information hiding (nascodere le informazioni),** secondo cui devono essere tenuti nascosti i dettagli implementativi della classe, conviene dichiarare tutti gli attributi (variabili di istanza) con il livello **private.**

Una classe che possiede attributi dichiarati come private, può comunque permettere la loro lettura e modifica definendo opportuni metodi.

In aggiunta ai livelli di visibilità si può specificare per l'attributo un'altra caratteristica:

- **static**
- **final**

La parola **static** indica un attributo legato alla classe, nel senso che, se vengono creati più oggetti di quella classe, esiste solo una copia dell'attributo.

Un attributo è invece dichiarato **final** se lo si vuole rendere una costante, in modo che possa assumere solo un unico valore. Solitamente gli attibuti final vengono anche dichiarati pubblici, perché non esiste il rischio che possano venire modificati dall'esterno in modo indesiderato.

```
public final int DIECI=10;
```

Dichiarazione dei metodi

I metodi rappresentano la parte dinamica di una classe. Servono per implementare le operazioni che una classe può eseguire: i metodi possono modificare gli attributi, cioè lo stato interno di un oggetto oppure calcolare i valori che verranno restituiti al richiedente.

Un metodo è caratterizzato dai diversi elementi che devono essere dichiarati:

- **un nome,**
- **un blocco di istruzioni,**
- **un tipo di valore di ritorno,**
- **un elenco di parametri,**
- **un livello di visibilità.**

La **dichiarazione dei metodi** deve essere fatta all'interno del blocco di una classe. La struttura generale di un metodo è la seguente:

Gli elementi che sono obbligatori sono il tipo restituito e il nome del metodo, mentre il livello di visibilità e i parametri sono facoltativi e possono anche non essere presenti nella dichiarazione.

Esempio:

Scrivere la dichiarazione di un metodo che esegue la somma di due numeri passati come parametri.

```
public int somma (int a, int b)
{
 int sum;
 sum=a+b;
 return sum;
}
```

Il **nome** del metodo per convenzione lo si fa iniziare con la lettera minuscola. Le variabili usate all'interno del metodo sono chiamate variabili locali. Queste variabili, a differenza delle variabili di istanza, non

permettono di indicare il livello di visibilità perché sono locali ai metodi. Vengono create quando viene eseguito il metodo e vengono distrutte non appena il metodo termina l'esecuzione. La loro durata corrisponde alla durata del metodo.
Il **blocco di istruzioni** è identificato da una coppia di parentesi graffe, all'interno delle quali vengono specificate le istruzioni, intese come dichiarazioni di variabili e istruzioni operative.

Le variabili usate all'interno dei metodi sono chiamate **variabili locali**. Queste variabili, a differenza delle variabili di istanza, non permettono il livello di visibilità perché sono locali ai metodi. Vengono create quando viene eseguito il metodo e vengono distrutte non appena il metodo termina l'esecuzione. Non possono esistere le istruzioni all'esterno dei blocchi definiti dai metodi.
Per esempio la seguente dichiarazione di classe non è corretta a causa di un errore in fase di compilazione:

```
class Coppia
{
  int a;
  int b;
  System.out.println("coppia ="+a+","+b);
}
```

La dichiarazione di classe corretta per visualizzare la coppia di numeri è al seguente:

```
class Coppia
{
  int a;
  int b;
  void stampa()
  {
     System.out.println("coppia ="+a+","+b);
  }
}
```

Il **tipo di valore di ritorno** viene specificato nella dichiarazione del metodo e indica quale sarà il valore restituito al termine dell'esecuzione del metodo. Il tipo può essere qualunque dei tipi di dato che sono usati in java e deve essere indicato prima del nome del metodo.
Es:

```
public int totale()
public void accelera()
```

Il primo metodo restituirà un valore di tipo int, mentre il secondo non restituirà alcun valore in quanto dichiarato con void.

I metodi, dichiarati con void ricordano le procedure del linguaggio Pascal, mentre i metodi che restituiscono un valore sono associabili alle funzioni del Pascal.

L'elenco dei parametri usati da un metodo vengono elencati dopo il nome del metodo, all'interno delle parentesi tonde. I parametri sono separati con la virgola. Se un metodo non presenta dei parametri, viene dichiarato usando solo le due parentesi tonde.
In java, i parametri possono essere passati solo per valore.

```
public int somma (int a, int b)
```

Ogni volta che viene richiamato un metodo, I parametric assumono il ruolo di variabili locali.

Per quanto riguarda **i tipi di riferimento**, cioè gli array e gli oggetti, viene creata una copia del riferimento e non dell'oggetto. L'array quindi non viene duplicato, viene creata una variabile locale che contiene una

copia di riferimento dello stesso array. Ne segue che una modifica all'interno del metodo, fatte usando il parametro, modifico anche l'array originale.

I livelli di visibilità di un metodo

I livelli di visibilità dei metodi, come per gli attributi, specificano se il metodo può essere visto e richiamato da altri oggetti. I livelli di visibilità che possono essere specificati sono:

- **public**
- **private**
- **protected**

Questi tre livelli di visibilità applicati ai metodi, hanno lo stesso significato già visto per i livelli riferiti agli attributi: un metodo dichiarato **public** può essere richiamato da qualunque altra classe; un metodo **private** non può essere richiamato esternamente alla classe in cui è dichiarato; un metodo dichiarato **protected** è visibile solo dalla classi che appartengono alla stessa libreria oppure dalle sottoclassi della classe in cui è dichiarato.

Solitamente i metodi vengono dichiarati usando la parola chiave public. Il livello private viene scelto quando il metodo è usato solo all'interno della classe. Questo si verifica quando un'operazione può essere scomposta in diverse azioni e ad ognuna viene associato un metodo privato.