

OCA CHAPTER 4

Il capitolo 4 del libro per la certificazione ORACLE sul linguaggio java parla di:

- 1) Variabili di istanza
- 2) Parola chiave *final*
- 3) Modificatori di accesso
- 4) Inizializzazioni
- 5) Breve intro alle espressioni lambda

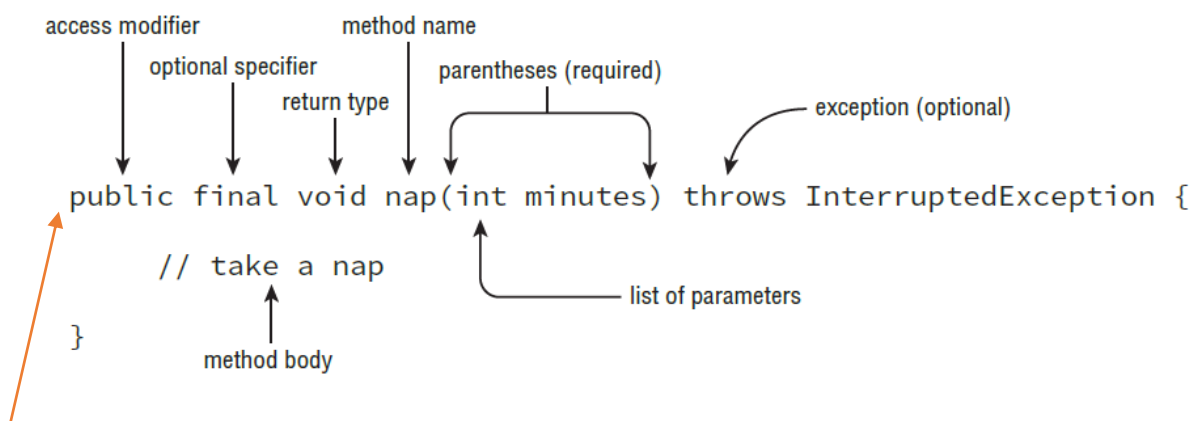
- METODI

Ogni programma java che può essere eseguito a un metodo *main* all'interno di una classe.

In effetti abbiamo visto che il metodo *main* serve al compilatore per capire da dove far partire l'esecuzione, ovvero in gergo si dice che funge da "entry point" per l'applicazione che scriviamo.

Ma il metodo *main* non è l'unico metodo che possiamo scrivere, anzi possiamo scrivere molti altri metodi all'interno di una classe. Vediamo quindi qual è la sintassi per la definizione di un nuovo metodo:

FIGURE 4.1 Method signature



Questa viene chiamata firma del metodo o in inglese signature e corrisponde alla definizione del metodo, ovvero:

- 1) qual è il nome del metodo
- 2) la visibilità del metodo
- 3) qual è l'input che prende il metodo
- 4) qual è l'output che restituisce il metodo
- 5) quale eccezioni può generare il metodo

... ..

In pratica la firma del metodo (che corrisponde alla sua definizione) specifica tutte le informazioni di cui ha bisogno.

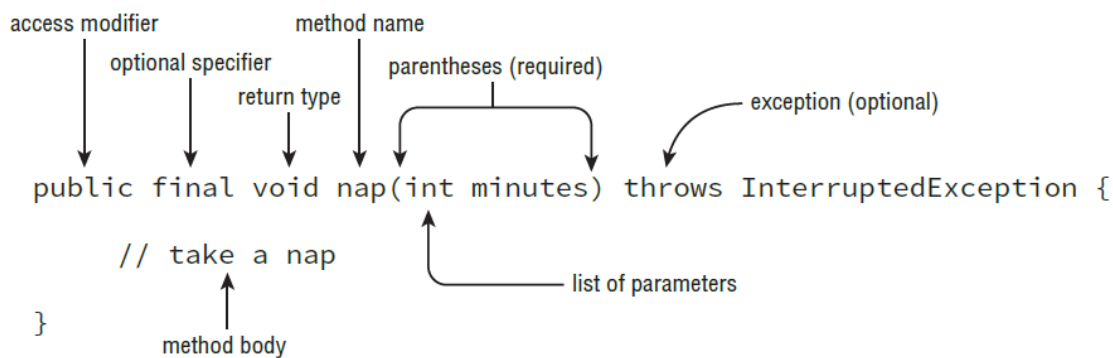
La definizione di un metodo abbiamo visto che quindi è fatta da più parti:

- Nome del metodo
- Visibilità del metodo
- Input che prende il metodo
- Eccezioni che può generare il metodo
- e altre ancora...

Non tutte queste parti però sono obbligatorie quando definiamo un nuovo metodo, o meglio quando specifichiamo la sua firma.

Rendiamo tutto più chiaro con un esempio e per farlo riprendiamo l'esempio di prima:

FIGURE 4.1 Method signature



Le tabelle sottostanti illustrano le parti obbligatorie e non per la definizione del metodo di sopra:

TABLE 4.1 Parts of a method declaration

Element	Value in nap () example	Required?
Access modifier	public	No
Optional specifier	final	No

Element	Value in <code>nap()</code> example	Required?
Return type	<code>void</code>	Yes
Method name	<code>nap</code>	Yes
Parameter list	<code>(int minutes)</code>	Yes, but can be empty parentheses
Optional exception list	<code>throws InterruptedException</code>	No
Method body	<pre>{ // take a nap }</pre>	Yes, but can be empty braces

- MODIFICATORI DI ACCESSO

Il linguaggio java mette a disposizione 4 tipi di modificatori di accesso:

- 1) Modificatore di accesso ***public***
specificando questo tipo di modificatore di accesso in un metodo, l'effetto è che tale metodo è visibile ovunque e quindi può essere richiamato in qualsiasi altra classe.
- 2) Modificatore di accesso ***protected***
specificando questo tipo di modificatore di accesso in un metodo, l'effetto è che tale metodo è visibile e quindi richiamabile solo all'interno dello stesso package oppure dalle sottoclassi che estendono la classe che ospita tale metodo
- 3) Modificatore di accesso ***default***
specificando questo tipo di modificatore di accesso in un metodo, l'effetto è che tale metodo è visibile e quindi richiamabile solo all'interno dello stesso package. Quindi se richiamo tale metodo all'interno di una classe che si trova in un package diverso dalla classe dove è stato definito il metodo ottengo un errore in compilazione.

Notiamo che in realtà non esiste una parola chiave per il modificato di accesso default. Infatti per usare questo modificatore basta non specificare nessun modificatore di accesso prima del metodo.

Ad essere proprio sinceri, esiste una parola chiave (ovvero riservata dal linguaggio) ***default*** ma non centra niente con il modificatore di accesso di default.


- 4) Modificatore di accesso ***private***
specificando questo tipo di modificatore di accesso in un metodo, l'effetto è che tale metodo è visibile e quindi richiamabile solo all'interno della classe dove è stato definito.

STAI MOLTO ATTENTO PERCHE' NELE DOMANDE DI ESAME PIACE FARE GLI SCHERZI SULLA CONTENUTO SINTATTICO DELLA FIRMA DEI METODI.

QUINDI TU RICORDA QUELLO CHE OBLIGATORIO E NON NELLA FIRMA DI UN METODO.

ESERCIZIO:

```
public void walk1() {}  
default void walk2() {} // DOES NOT COMPILE  
void public walk3() {} // DOES NOT COMPILE  
void walk4() {}
```



Perché la riga 2 e 3 non compilano?

- La riga 2 sembra sintatticamente scritta correttamente a parte che non esiste una parola chiave **default** per il modificatore di accesso di default.
La parola chiave default viene usata all'interno del costrutto switch e non nella firma dei metodi e quindi il compilatore segnalerà questo errore e la compilazione del programma non andrà a buon fine.
- La riga 3 invece non è sintatticamente corretta in quanto sono state invertite le parole chiavi per il Tipo di ritorno (void) e per il modificatore di accesso (public).
Il modificatore di accesso se presente deve sempre essere specificato in prima posizione, quindi in questo caso poiché la firma del metodo è scorretta sintatticamente verrà generato un errore dal compilatore.
La riga 3 sarebbe stata sintatticamente corretta se fosse stata: `public void walk3() {}`

Notiamo che se non specifichiamo nessun modificatore di accesso nella firma di un metodo, questo implica che stiamo usando il modificatore di default.

- SPECIFICATORI OPZIONALI

Prima abbiamo visto i modificatori di accesso anche chiamati come specificatori obbligatori.

I specificatori opzionali sono invece quelle parti della definizione del metodo che non sono obbligatorie.

Nel senso che se non ci sono nella firma del metodo, allora il metodo è ancora sintatticamente corretto e quindi il compilatore non darà alcun errore nella fase di compilazione.

Elenchiamo adesso alcuni dei specificatori opzionali più noti che possono essere inclusi nella definizione di un metodo:

- **static**

Questa parola chiave (o specificatore opzionale) viene usata nella definizione di un metodo per far capire al compilatore che questo metodo può essere richiamato anche senza istanziare alcun oggetto della classe che contiene tale metodo static.

Notiamo che questo specificatore lo abbiamo sempre visto all'interno della definizione del metodo main. Questo perché quando viene richiamato il metodo main dalla JVM non esiste ancora nessun oggetto in memoria.

- **abstract**

Questa parola chiave (o specificatore opzionale) viene usata per far capire al compilatore che il metodo che possiede tale specificatore non verrà implementato nella classe dove risiede la firma di tale metodo e quindi non avrà nessun corpo del metodo.

L'implementazione di tale metodo abstract spetterà alle classi concrete (quindi non astratte) che estenderanno la classe che contiene tale metodo astratto.

- **final**

Questa parola chiave (o specificatore opzionale) viene usata per far capire al compilatore che il metodo che possiede tale specificatore non potrà essere sovrascritto (cioè fare l'override) dalle classi che estendono la classe che contiene questo metodo.

Se si provasse a sovrascrivere (fare l'override) nelle sottoclassi questo metodo che contiene nella sua definizione la parola chiave final, allora verrà generato un errore in fase di compilazione.

- **synchronized**

- **native**

- **strictfp**

Gli ultimi 3 specificatori opzionali non sono compresi tra le domande per l'esame OCA e quindi non verranno trattati.

ESERCIZIO:

```
public void walk1() {}  
public final void walk2() {}  
public static final void walk3() {}  
public final static void walk4() {}  
public modifier void walk5() {} // DOES NOT COMPILE  
public void final walk6() {} // DOES NOT COMPILE  
final public void walk7() {}
```

**FAI
ATTENZIONE**

Perché la riga 5 e la riga 6 non compilano?

- La riga 5 non compila perché non esiste nessun specificatore ne opzionale ne obbligatorio da includere nella firma del metodo che si chiama modifier. Quindi l'inclusione di tale provocherà un errore sintattico nella firma del metodo e quindi un errore in fase di compilazione.
- La riga 6 invece non compila in quanto sono state invertite le parole chiavi void (ovvero il tipo di ritorno che ricordiamo essere uno specificatore obbligatorio) e final (specificatore opzionale) provocando quindi un errore sintattico nella firma del metodo e quindi un errore in fase di compilazione.

Nota tipo di ritorno nella definizione del metodo va essere messo sempre prima del nome del metodo.

- La riga 1 compila senza problemi in quanto è sintatticamente corretta. Essa include
 - o Modificatore di accesso public all'inizio che è uno specificatore opzionale e quindi può anche non essere incluso nella firma di un metodo.
 - o Il tipo di ritorno che è uno specificatore obbligatorio e quindi bisogna assolutamente includerlo nella firma di un metodo.
 - o Il nome del metodo che è uno specificatore obbligatorio e quindi bisogna assolutamente includerlo nella firma del metodo.
 - o Parentesi (e) che sono specificatori obbligatori nella firma del metodo e quindi obbligatorie.
 - o Parentesi { e } che sono specificatori obbligatori nella firma del metodo e quindi obbligatorie.
- La riga 2 compila senza problemi in quanto sintatticamente corretta. Essa include
 - o Modificatore di accesso public all'inizio che è uno specificatore opzionale e quindi può anche non essere incluso nella firma di un metodo.
 - o Specificatore opzionale final
 - o Il tipo di ritorno che è uno specificatore obbligatorio e quindi bisogna assolutamente includerlo nella firma di un metodo.
 - o Il nome del metodo che è uno specificatore obbligatorio e quindi bisogna assolutamente includerlo nella firma del metodo.
 - o Parentesi (e) che sono specificatori obbligatori nella firma del metodo e quindi obbligatorie.
 - o Parentesi { e } che sono specificatori obbligatori nella firma del metodo e quindi obbligatorie.
- La riga 3 compila senza problemi in quanto sintatticamente corretta. Essa include
 - o Modificatore di accesso public all'inizio che è uno specificatore opzionale e quindi può anche non essere incluso nella firma di un metodo.
 - o Specificatore opzionale static
 - o Specificatore opzionale final
 - o Il tipo di ritorno che è uno specificatore obbligatorio e quindi bisogna assolutamente includerlo nella firma di un metodo.
 - o Il nome del metodo che è uno specificatore obbligatorio e quindi bisogna assolutamente includerlo nella firma del metodo.

- Parentesi (e) che sono specificatori obbligatori nella firma del metodo e quindi obbligatorie.
- Parentesi { e } che sono specificatori obbligatori nella firma del metodo e quindi obbligatorie.
- La riga 4 compila senza problemi in quanto sintatticamente corretta. Essa include
 - Modificatore di accesso public all'inizio che è uno specificatore opzionale e quindi può anche non essere incluso nella firma di un metodo.
 - Specificatore opzionale static
 - Specificatore opzionale final
 - Il tipo di ritorno che è uno specificatore obbligatorio e quindi bisogna assolutamente includerlo nella firma di un metodo.
 - Il nome del metodo che è uno specificatore obbligatorio e quindi bisogna assolutamente includerlo nella firma del metodo.
 - Parentesi (e) che sono specificatori obbligatori nella firma del metodo e quindi obbligatorie.
 - Parentesi { e } che sono specificatori obbligatori nella firma del metodo e quindi obbligatorie.

Notiamo che i specificatori opzionali possono essere inclusi nella firma del metodo senza tenere in considerazione il loro ordine relativo. Infatti nella riga 3 e 4 i due modificatori opzionali (static e final) sono stati invertiti di posizione ma la firma del metodo è ancora sintatticamente corretta.

- La riga 7 potrebbe ingannare l'esaminante ad errori dicendo che non compilerebbe ma invece compila. Infatti java permette ai specificatori opzionali di apparire prima dei modificatori di accesso se sono presenti. Quindi presta attenzione a casi come questo.

- TIPI DI RITORNO