Azzolini Riccardo 2018-11-30

Package

1 Unità di compilazione

Un'**unità di compilazione** è un file contenente codice sorgente Java.

Una singola unità di compilazione può contenere diverse classi e interfacce, ma al massimo una di esse può essere dichiarata `public` (e in tal caso determina il nome del file).

2 Package

I **package** permettono di raggruppare in unità logiche classi e interfacce correlate tra loro.

Per aggiungere un'unità di compilazione a un package, è necessario scrivere

```
package nome_package;
```

come prima istruzione nell'unità.

Perché la JVM riesca a trovare il bytecode, il nome del package deve corrispondere al percorso, relativo al `CLASSPATH`, del file `.class`. Ad esempio, se la directory `/myjavalib` fa parte del `CLASSPATH` e il file `Frazione.class` appartiene al package `prog.utili`, esso si potrebbe trovare al percorso `/myjavalib/prog/utili/Frazione.class`.

3 Nomi completi

È possibile usare una classe (o interfaccia) appartenente a un package senza bisogno di direttive di importazione, specificandone il **nome completo**:

```
nome_package.nome_classe
```

In questo modo, si possono utilizzare anche classi diverse con lo stesso nome.

4 Direttive di importazione

Una **direttiva di importazione** aggiunge il nome di una classe o interfaccia appartenente a un package allo **spazio dei nomi** che il compilatore prende in considerazione, permettendo quindi di riferirsi a tale classe/interfaccia senza bisogno di usare il nome completo.

```
import nome_package.nome_classe;
```

L'**importazione su richiesta** aggiunge i nomi di tutte le classi/interfacce al di sotto di un package allo spazio dei nomi del compilatore.

```
import nome_package.*;
```

Se si importano su richiesta due package contenenti classi/interfacce con lo stesso nome, si può verificare un conflitto di nomi, ma ciò accade solo se il nome ambiguo viene effettivamente utilizzato nel codice. In caso di conflitto, è necessario ricorrere ai nomi completi.

5 Package importati automaticamente

Il compilatore importa automaticamente i contenuti di `java.lang` e del **package di default**.

Quest'ultimo è un package senza nome, creato automaticamente, che include tutte le classi/interfacce definite nella directory di compilazione che non sono esplicitamente incluse in un package.

6 Livelli di visibilità

I package forniscono un ambiente di protezione in cui sviluppare codice nascondendone i dettagli all'esterno.

Esistono infatti diversi livelli di visibilità, specificati da appositi modificatori:

- `private` (membri): visibili solo all'interno del codice della classe
- `public` (membri, classi e interfacce): visibili ovunque
- **amichevole** (membri, classi e interfacce): visibili solo all'interno del package (questo livello si ottiene omettendo gli altri modificatori)
- `protected`: legato all'estensione delle classi