Azzolini Riccardo 2019-10-22

Object e package diagram

1 Object diagram

Un **object diagram** descrive le singole istanze delle classi (cioè gli oggetti) e delle associazioni (cioè i link) rappresentate in un particolare class diagram.

Esso è adatto a descrivere esempi o situazioni specifiche, dato che permette di fare una "fotografia" delle istanze esistenti in un certo istante di tempo.

1.1 Oggetti

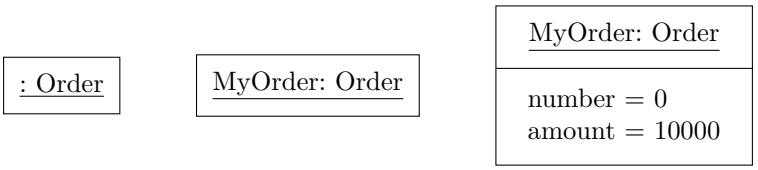
Ciascun oggetto (istanza di una classe) è caratterizzato da:

- nome dell'oggetto;
- classe di cui è istanza;
- valori degli attributi.

Nella rappresentazione grafica dell'oggetto, è obbligatorio indicare la classe di appartenenza, mentre il suo nome e i valori degli attributi sono opzionali.

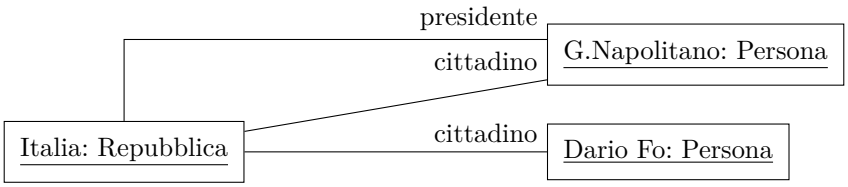
Data una classe,

una sua istanza si può raffigurare come:



1.2 Link

Un **link** è un'istanza di un'associazione, cioè una connessione fisica o concettuale tra due oggetti.

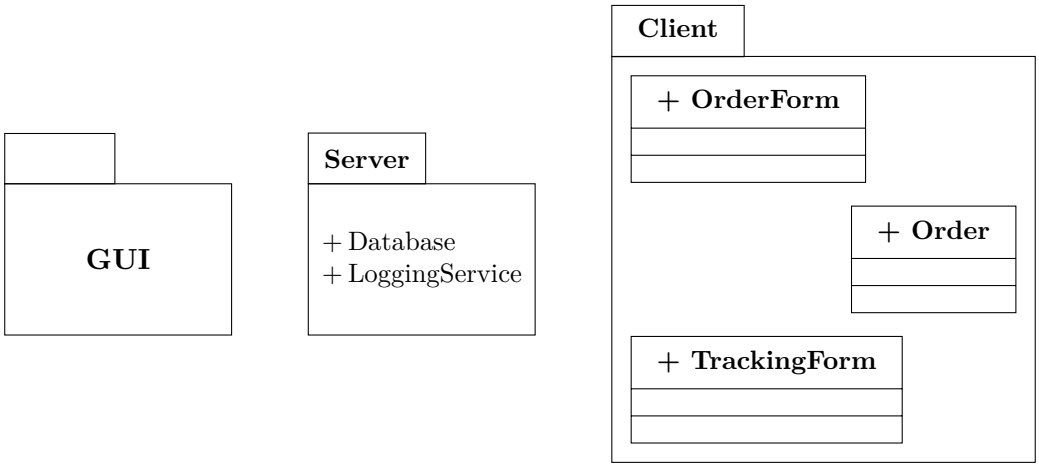


I link presenti in un object diagram devono rispettare le molteplicità delle associazioni di cui sono istanze.

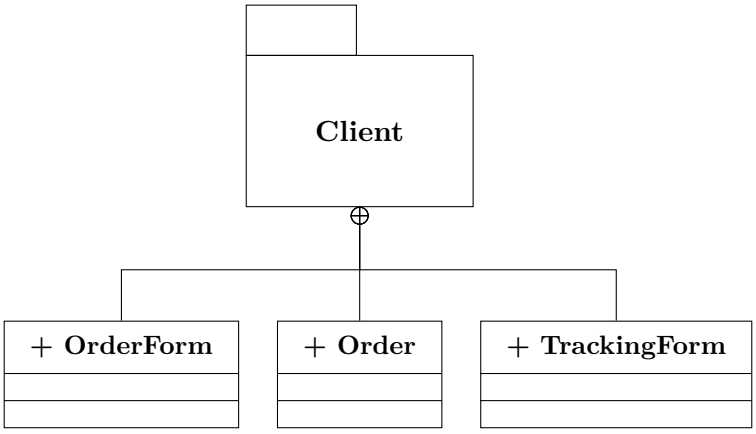
2 Package diagram

I **package** sono contenitori, in generale, di elementi UML (spesso classi). Un package definisce un namespace, quindi i nomi degli elementi al suo interno devono essere diversi.

Graficamente, un package può essere rappresentato con vari livelli di dettaglio, come ad esempio:



I contenuti di un package si possono indicare anche con la notazione delle relazioni di annidamento:



2.1 Motivazioni dei package

I package permettono di modellare sistemi complessi in modo gerarchico.

Ciascun package deve avere:

- alta coesione all'interno;
- interfacce precise e limitate.

In questo modo, esso fornisce una vista relativa a un determinato aspetto del problema, il quale può allora essere analizzato, sviluppato, ecc. in modo (per quanto possibile) indipendente dal resto del problema.

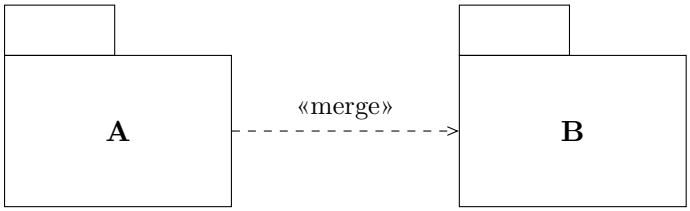
2.2 Relazioni tra package

Una relazione tra due package "riassume" relazioni multiple *dello stesso tipo* tra i singoli elementi che appartengono a tali package. In particolare, si possono definire relazioni di:

- *generalizzazione*, se esiste almeno una generalizzazione tra gli elementi dei due package;
- *dipendenza*, se esiste almeno una relazione di dipendenza tra gli elementi dei due package.

Esistono poi alcune relazioni che riguardano direttamente i package:

- *Annidamento*: un package può contenere altri package.
- *Merge*: è una relazione molto simile alla generalizzazione, con la quale i contenuti di due package vengono combinati in un unico package. Ad esempio, con il merge raffigurato nel diagramma seguente, il package A acquisisce anche le caratteristiche di B:



Esso si può usare quando package diversi contengono elementi che hanno gli stessi nomi e rappresentano gli stessi concetti, ma (spesso) con definizioni diverse per scopi diversi.

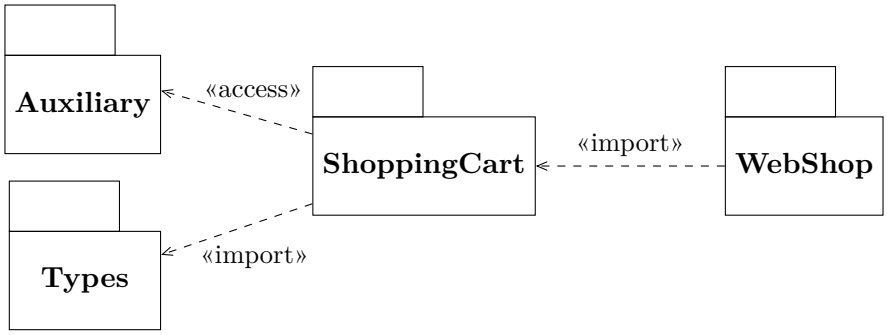
Il merge permette di definire "delta" in package diversi e combinarli per ottenere un singolo package con tutte le caratteristiche necessarie.

2.2.1 Dipendenze

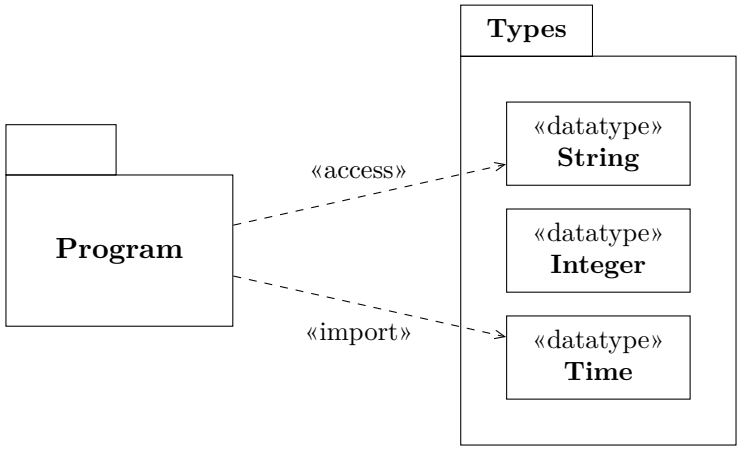
Le relazioni di dipendenza tra package devono essere esplicite: *non* è sufficiente che gli elementi di un package siano pubblici perché questi possano essere usati da altri package. Inoltre, le dipendenze *non* sono transitive: se il package A può accedere a B, e B può accedere a C, non è detto che A possa accedere a C.

UML definisce due stereotipi per le dipendenze tra package:

- «access»: indica che gli elementi contenuti nel package target possono essere referenziati dal package client, ma è necessario usare come prefisso il nome del package target. In altre parole, «access» non modifica il namespace del client.
- «import»: aggiunge al namespace del package client gli elementi contenuti nel package target. Il client può quindi usare gli elementi importati come se fossero stati dichiarati localmente, ma in compenso non ci devono essere conflitti fra i nomi importati e quelli già presenti nel client, altrimenti il modello è mal formato.



Le relazioni di access e import possono riguardare anche i singoli elementi di un package:



In particolare, in un import di un singolo elemento, è possibile specificare un *alias*: