Unified Modeling Language
Package Diagram

Sandro Morasca

Università degli Studi dell'Insubria

Dipartimento di Scienze Teoriche e Applicate

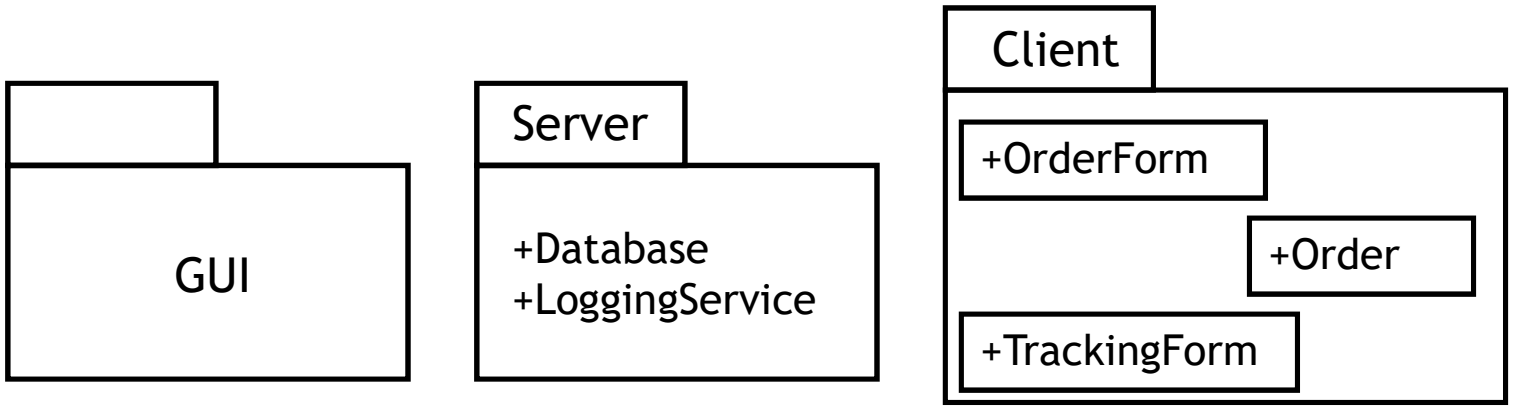
Via Ottorino Rossi 9 – Padiglione Rossi

21100 Varese, Italy

sandro.morasca@uninsubria.it

Package

- I package sono “contenitori” di elementi UML
  - classi
  - use case
  - …
- Un package definisce un namespace
  - i nomi degli elementi all’interno devono essere diversi
  - la visibilità degli elementi all’esterno viene specificata



Motivazioni

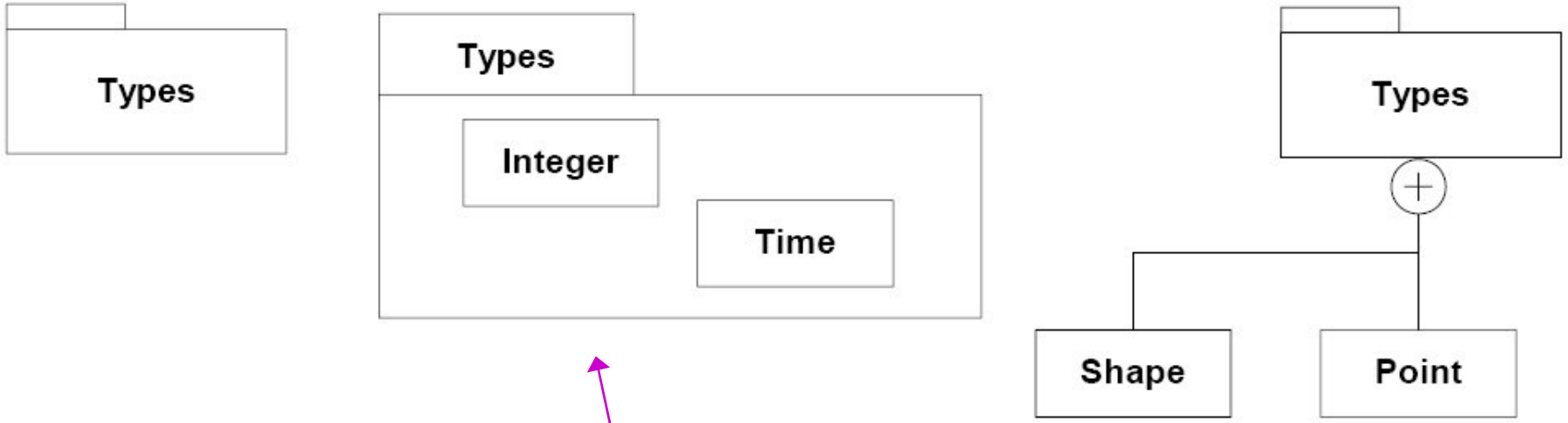
- Sistemi complessi (reali) devono essere modellati gerarchicamente, ad esempio contengono
  - classi
    - in gerarchia di ereditarietà
    - in relazioni di aggregazione/composizione
    - in relazioni d'uso
  - package in relazione fra loro
- Package
  - alta coesione all'interno
  - interfacce precise e limitate

Meccanismi

- Modulo: raggruppa classi, associazioni e generalizzazioni
  - fornisce una vista del problema
  - diminuisce la complessità del problema
- Forniscono un costrutto di raggruppamento per gli elementi definiti in un modello UML
- A volte si utilizza il termine *subsystem* per indicare un package

Esempi

- Package
- Relazioni



Indichiamo esplicitamente il contenuto del package

Dipendenze tra Package

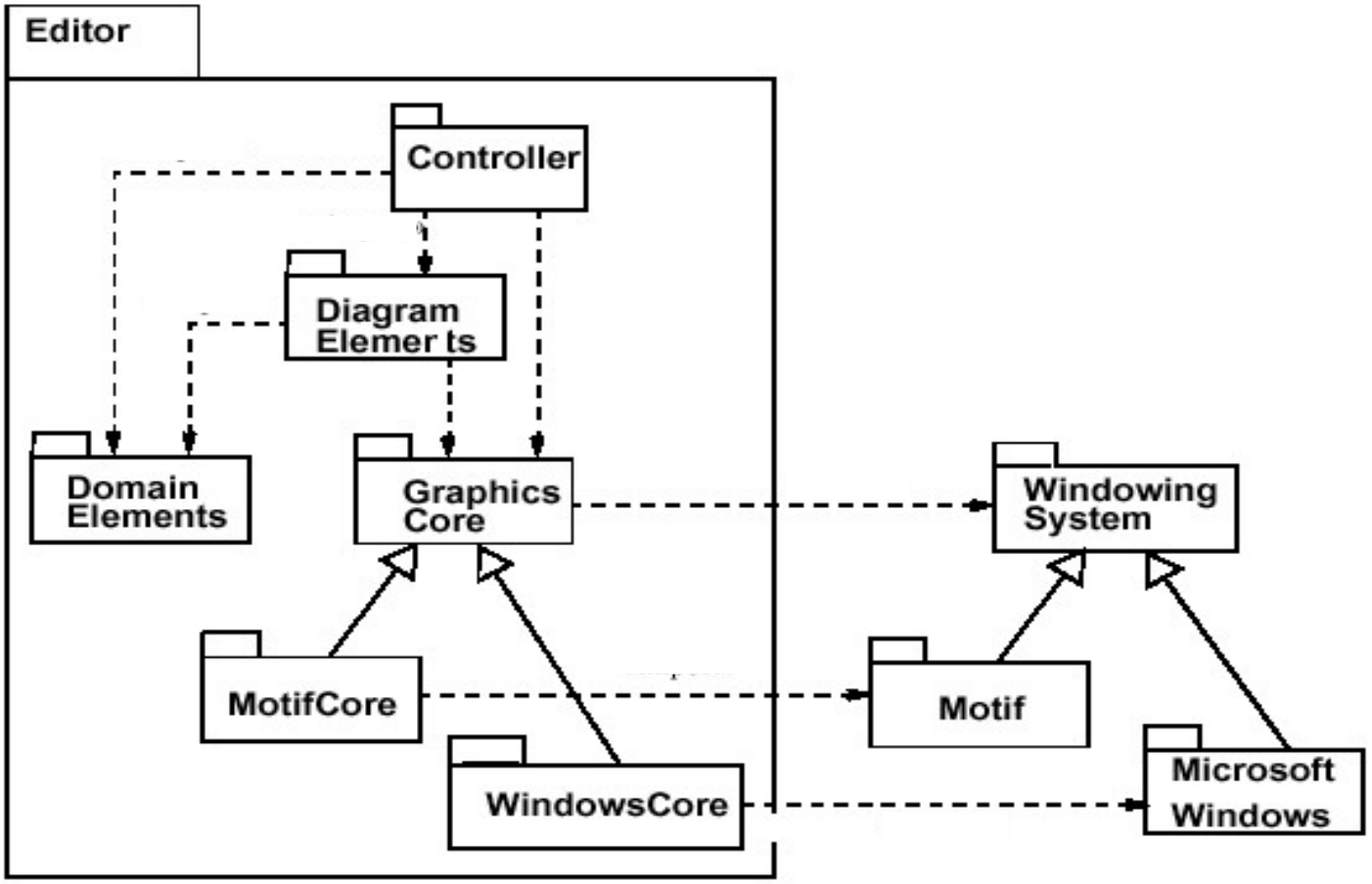
- Descrivono dipendenze esistenti tra elementi contenuti nei singoli package, ad un livello di astrazione più alto
  - dipendenze multiple dello stesso tipo tra singoli elementi (ad es. Classi) appartenenti a package diversi vengono “riassunte” in una singola dipendenza tra i package contenenti i diversi elementi
- Tipologie di relazioni tra package
  - Generalizzazione
    - Esiste almeno una relazione di generalizzazione tra elementi appartenenti a package diversi
  - Dipendenza
    - Esiste almeno una relazione di dipendenza tra elementi appartenenti a package diversi

Relazioni

- Aggregazioni
  - package possono contenere altri package
- Merge
  - molto simile a una generalizzazione
  - da usare quando elementi definiti in package diversi hanno lo stesso nome e rappresentano lo stesso concetto
    - spesso per scopi diversi
    - con “delta” definiti in package diversi
  - gli elementi base non vengono modificati
  - concettualmente: un’operazione in cui i contenuti di due package vengono uniti in un unico package
- Import
- Access

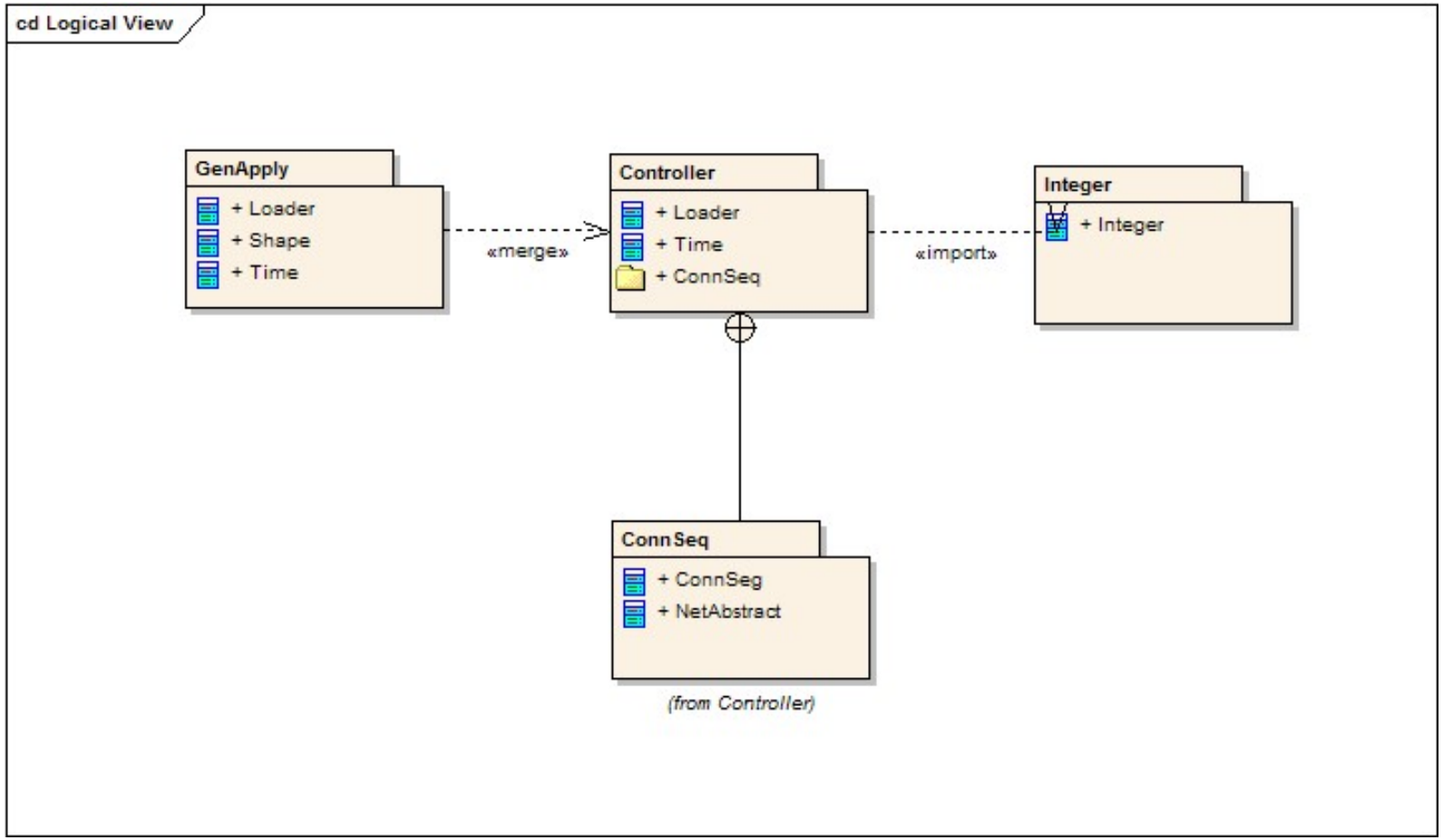
Dipendenze: esempio

- Package
- Relazioni



Esempio

- Package
- Relazioni



Dipendenze tra Package (2)

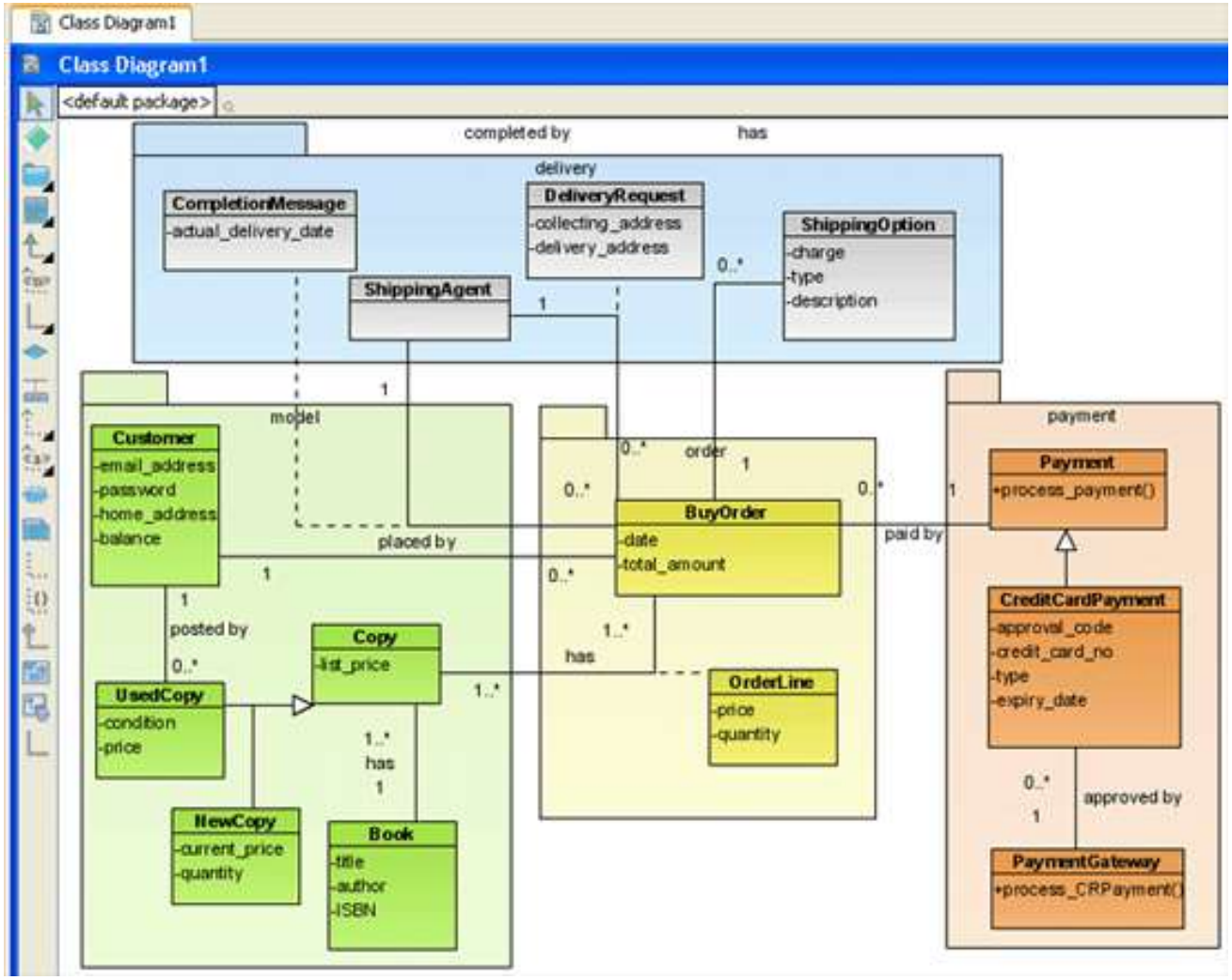
- In generale, se non si esplicita una relazione di dipendenza, un package non può accedere agli elementi contenuti in un altro package
  - le relazioni di dipendenza definiscono delle “permission” per l’accesso al contenuto di altri packages (limitatamente agli elementi con visibilità public o protected presenti nel package target)
- Le relazioni di dipendenza tra packages non sono transitive
  - se il package A può vedere B e B può vedere C, non è detto che A possa vedere C

Uso

- Top-down
  - come primo strumento per abbozzare il sistema e identificare i “componenti” principali
  - si parte con semplici “cartellette” vuote per poi riempirle con classi
- Bottom-up
  - come strumento per “gerarchizzare” e rendere leggibile un modello troppo complesso
  - si parte dalle classi e si raggruppano in package

Esempio

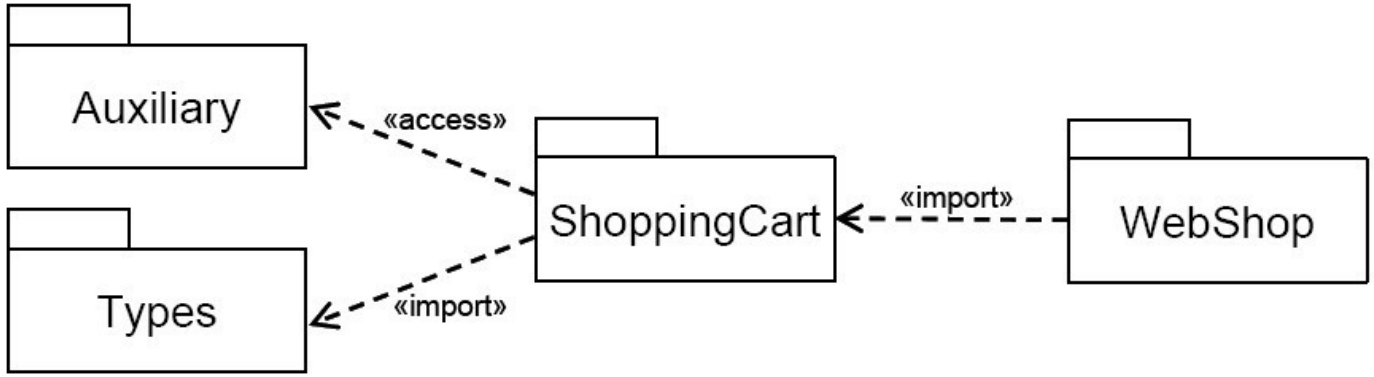
- Package
- Relazioni

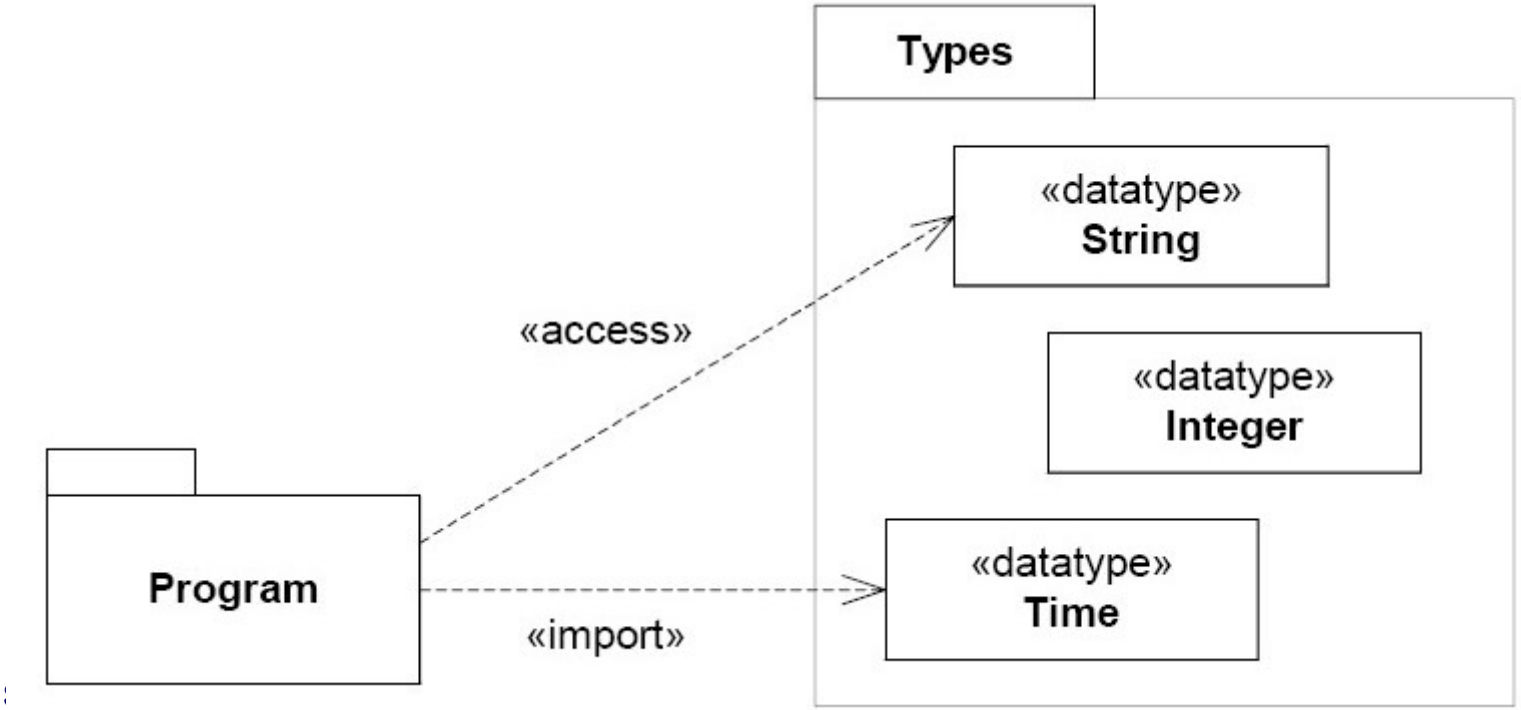


<<import>> e <<access>>

- Dipendenza con stereotipo <<access>>
  - gli elementi contenuti nel package target possono essere referenziati dagli elementi contenuti nel package client (o dai sub-package in esso contenuti)
  - una dipendenza di tipo <<access>> non modifica il namespace del client e non crea riferimenti di alcun tipo
    - Garantisce solo la possibilità di creare references
- Dipendenza con stereotipo <<import>>
  - i nomi degli elementi presenti nel namespace del package target vengono aggiunti al namespace del package client (con le stesse regole di visibilità valide per <<access>>)
  - se ci sono conflitti fra i nomi importati e nomi già presenti nel namespace del client, il modello è mal formato (ill formed)

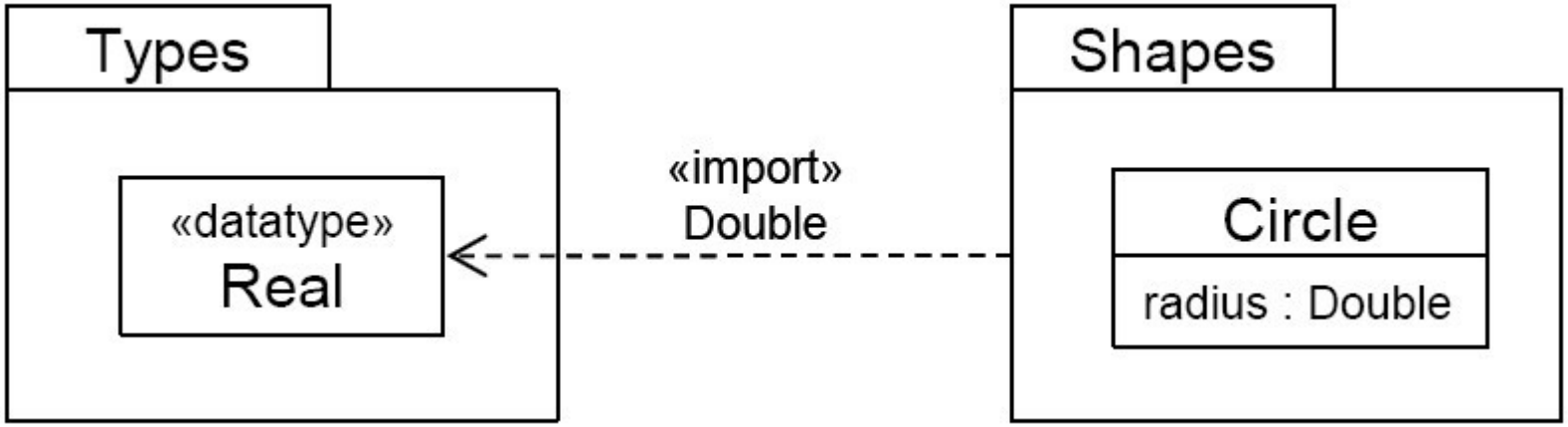
<<import>> e <<access>>: esempi





Import con aliasing

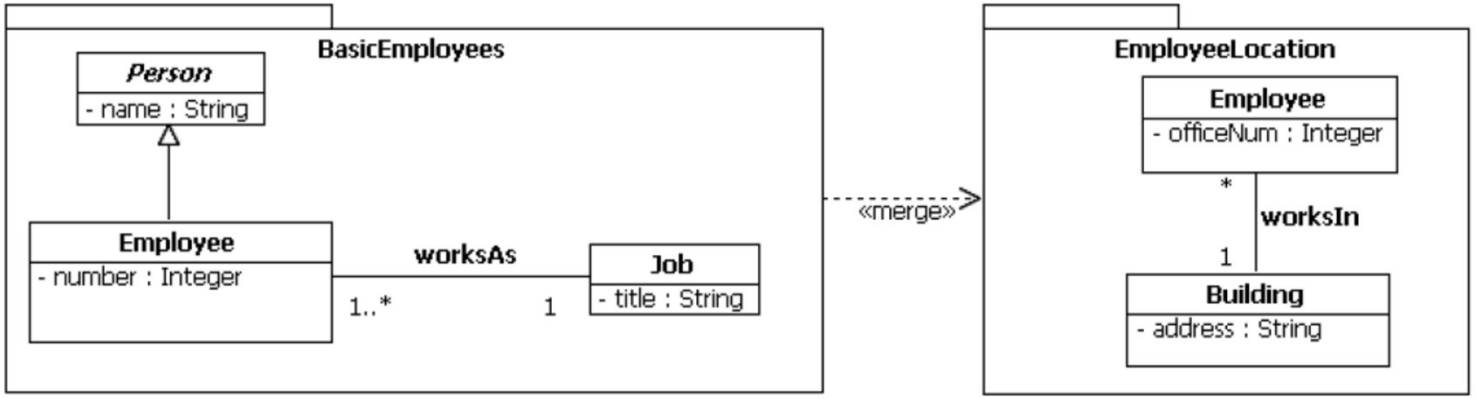
- Package
- Relazioni



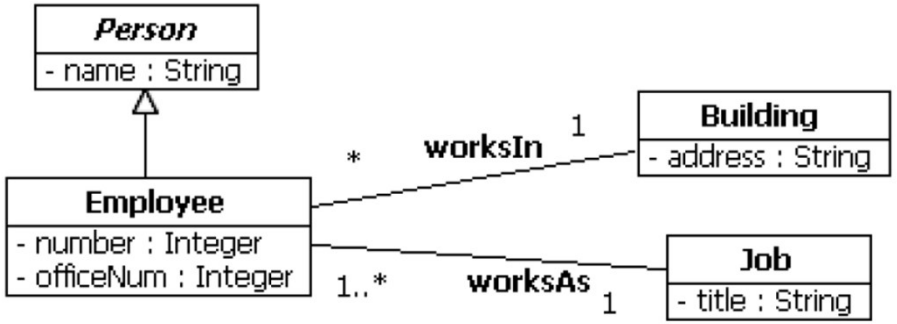
Package merge

- Una relazione tra package che indica che i contenuti devono essere combinati
  - simile alla generalizzazione, nel senso che l'elemento sorgente aggiunge le caratteristiche dell'oggetto target alle proprie, col risultato di fornire una combinazione dei due
  - usato quando elementi di package differenti hanno lo stesso nome e significato concettuale
  - si può usare per dare definizioni diverse, per scopi diversi, dello stesso concetto
    - il concetto base viene esteso incrementalmente, con ciascun incremento definito in un merge package separato

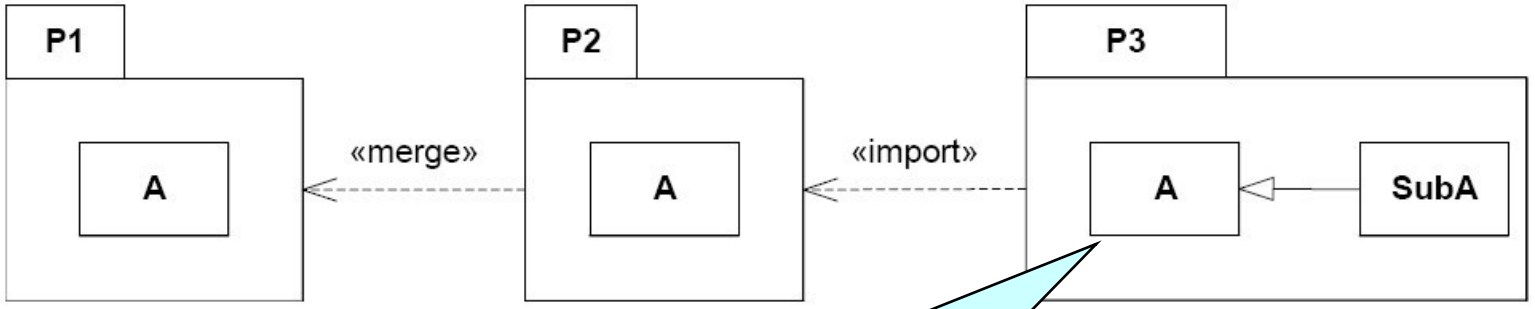
Package merge: esempio



- Risultato del merge



Package merge: uso



L'elemento A nel package P3 (P3::A) rappresenta il risultato del merge di P1::A in P2::A e non semplicemente P2::A.

Package merge: semantica