Laboratorio di
Ingegneria del Software
a.a. 2013-2014

LEZIONE 2 - PACKAGE DIAGRAM

Catia Trubiani

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica (DISIM) - Università degli Studi dell'Aquila

catia.trubiani@univaq.it

Riepilogo della lezione precedente

Cos'è un **Classe**? A cosa serve?

Qual è la differenza tra classe e istanza??

Car
registration number data speed direction

la classe rappresenta un'entità astratta

l'istanza rappresenta un'entità concreta

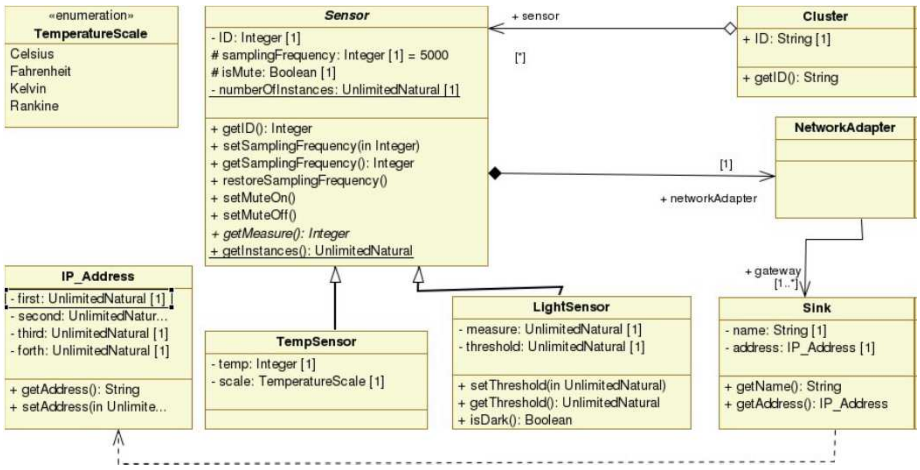
Riepilogo della lezione precedente

Cos'è un **Class Diagram**? A cosa serve?

Facciamo un esempio:

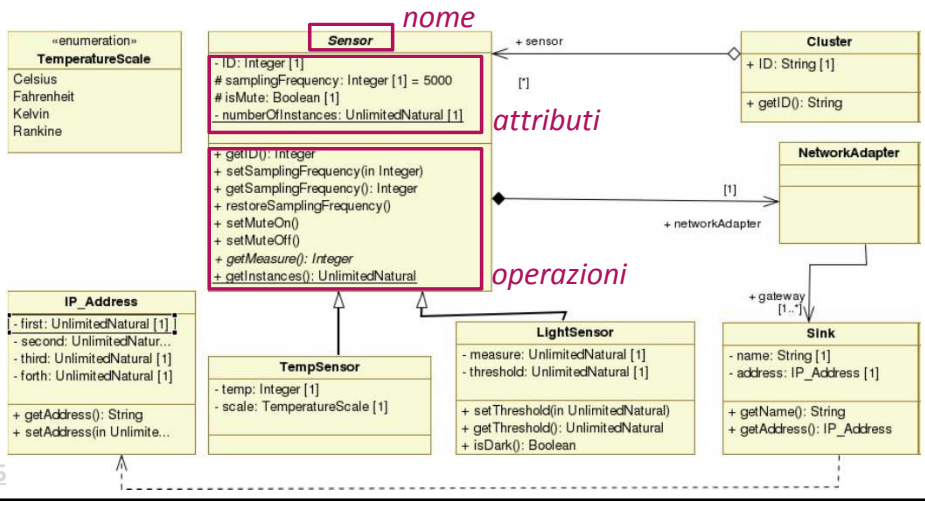
> una Sensor Network è una rete costituita da un insieme di piccoli apparecchi elettronici in grado di ricevere dati dall'ambiente circostante e comunicare tra loro mediante degli elementi per l'interfacciamento alla rete. Ogni sensore invia i dati collezionati ad un gateway. I sensori possono essere organizzati in aree di sensing o cluster.

Class Diagram per la sensor network



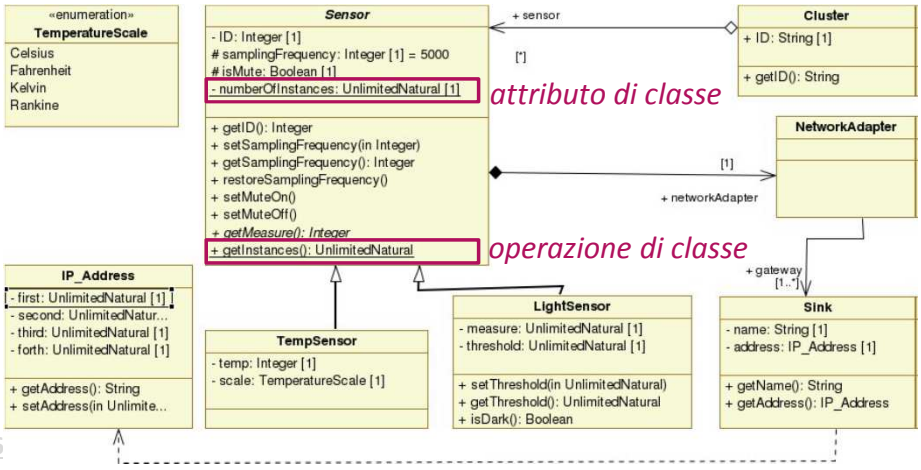
Concetti principali (1/5)

Cos'è una **Classe**? Quali sono gli elementi che la costituiscono?



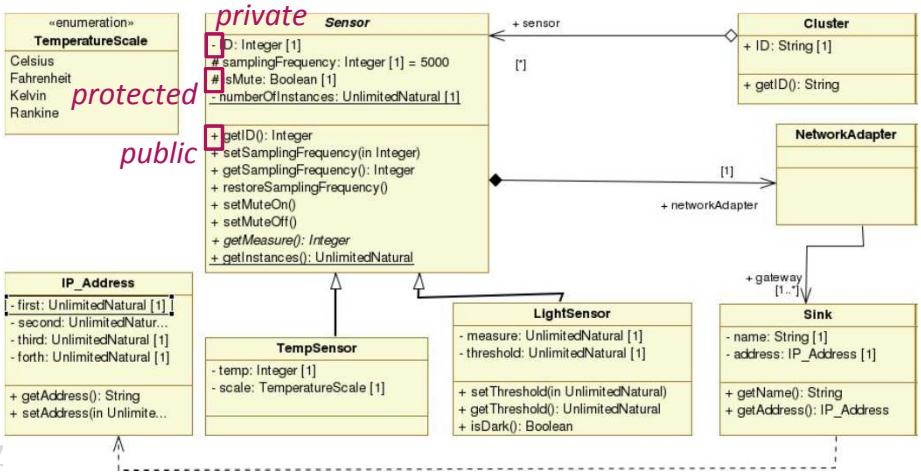
Concetti principali (2/5)

Cosa sono gli attributi e le operazioni di classe? A cosa servono?



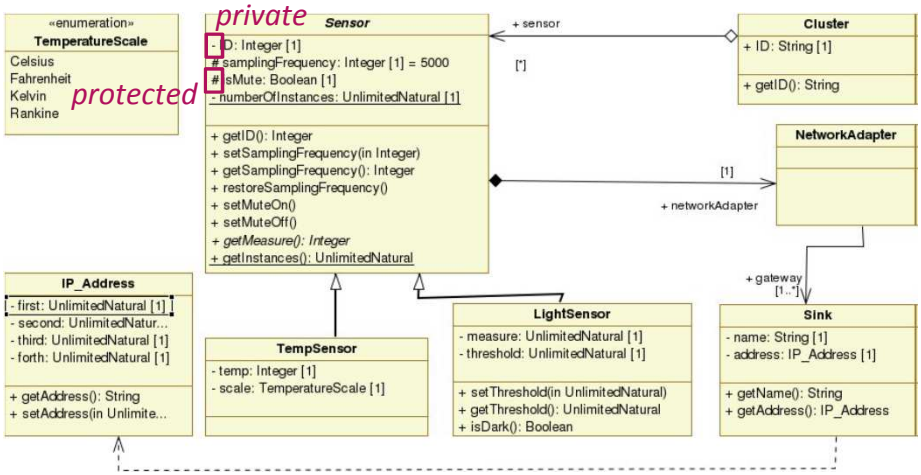
Concetti principali (3/5)

Come può essere definita la visibilità degli attributi e delle operazioni di una classe?



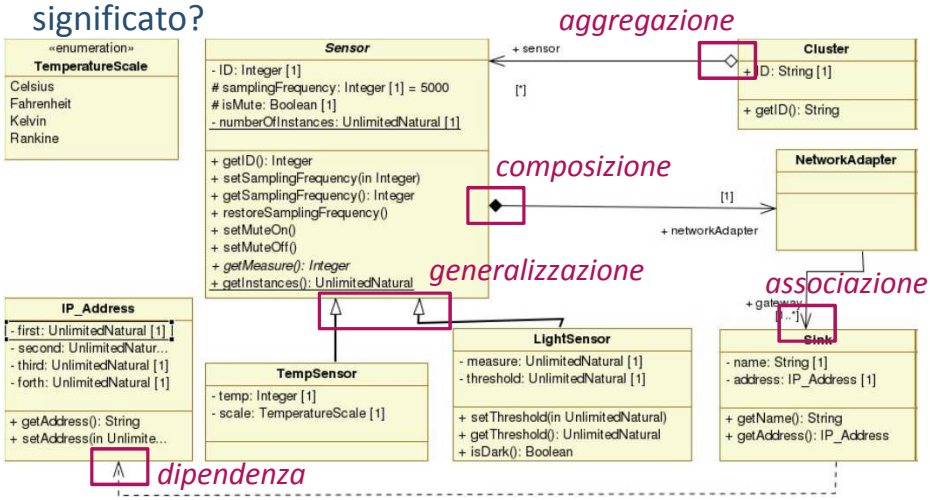
Concetti principali (4/5)

Qual è la differenza tra ‘protected’ e ‘private’?!



Concetti principali (5/5)

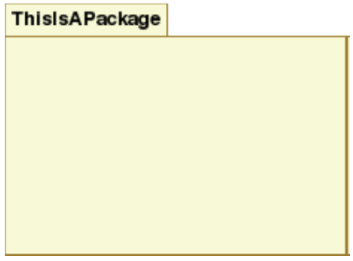
Quali sono le relazioni tra le classi? Qual è il loro significato?



Package Diagram

I package sono costrutti che permettono di raggruppare/organizzare entità da modellare

Rendono i diagrammi più semplici facilitandone la comprensione



Package Diagram

Meccanismo per organizzare elementi in gruppi (principio O.O. della modularità)

Classi, componenti, package, diagrammi, etc.

Meccanismo gerarchico con controllo sulla visibilità

Principali relazioni di dipendenza(/associazione):

- **import** : o "public import", include nella definizione del package elementi definiti in un altro package.

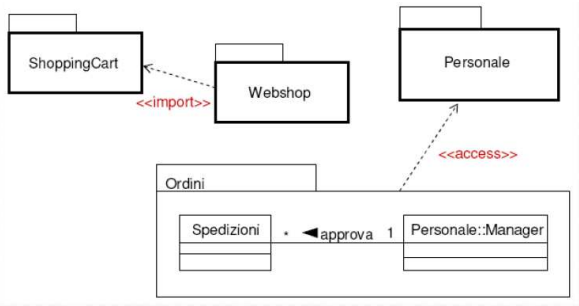
- **access**: o "private import", come import ma gli elementi importati non sono visibili esternamente dal package

- **merge**: gli elementi con lo stesso nome nel package source e nel package target sono fusi nel package risultante. Gli elementi non ridefiniti vengono importati.

Package Diagram - import e access

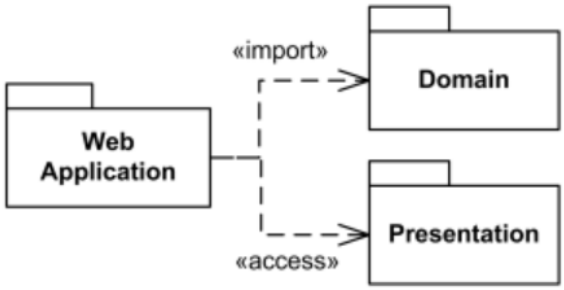
IMPORT: è una relazione di dipendenza tra package, che indica che un package contiene una copia degli elementi appartenenti ad un altro package

ACCESS: è una relazione di dipendenza tra package, che indica che tutti gli elementi del package di origine (nell'es. *Ordini*) possono accedere agli elementi pubblici del package target (nell'es. *Personale*)



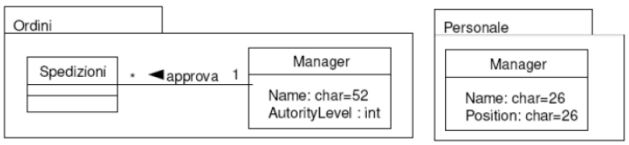
Un esempio di package import e access

Il package Web Application effettua un import privato (i.e., **access**) del package Presentation e un import pubblico (i.e., **import**) del package Domain

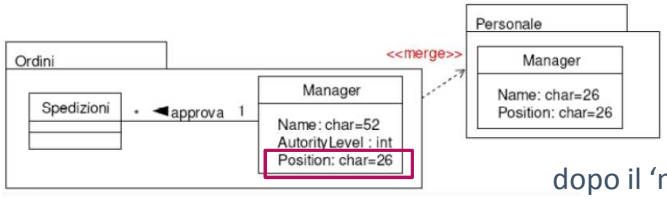


Package Diagram - merge

MERGE: è una relazione di dipendenza tra package, dove i contenuti del package target (nell'es. *Personale*) sono "uniti" a quelli del package origine (nell'es. *Ordini*) tramite specializzazione e ridefinizione, dove applicabile



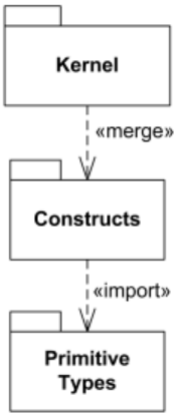
prima del 'merge'



dopo il 'merge'

Un esempio di package merge

Il package Kernel unisce il package Constructs che a sua volta effettua un import pubblico del package Primitive Types



Package Diagram (per modelli funzionali)

Quando si organizzano modelli funzionali (use case, workflow, ecc.) si usano i package per modellare la struttura modulare del sistema da applicare nel mondo reale.

Quando si organizza il codice sorgente, si usano i package per rappresentare i differenti strati di un codice sorgente.

Per esempio:

- logica di presentazione
- controller layer
- data access layer
- integration layer
- business services layer

Package Diagram (per modelli a componenti)

Quando si organizzano modelli con componenti software, si usano i package per raggruppare i componenti in base alla proprietà (possesso) e/o alle possibilità di riuso. Per esempio:

- commercial-off-the-shelf products
  - componenti disponibili sul mercato
- componenti framework open source
  - componenti che servono per coordinare le attività di un sistema software ed è possibile modificarle perché viene fornito il codice sorgente
- componenti framework "custom-built"
  - componenti framework selezionate dall'utente finale
- componenti applicazione "custom-built"
  - componenti applicative selezionate dall'utente finale

Package Diagram (per modelli di deployment)

Quando si organizzano modelli deployment, si usano i package per rappresentare i differenti tipi di ambienti di distribuzione (ambienti deployment) che si modelleranno. Per esempio:

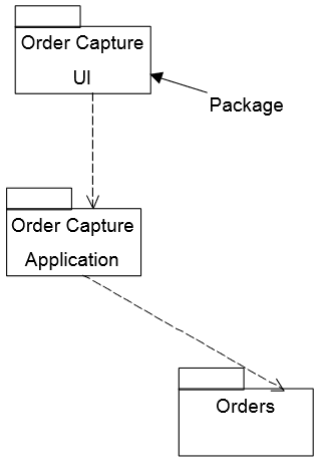
- ambiente di produzione
- ambiente di pre-produzione
- ambiente di test d'integrazione
- ambiente di test di sistema
- ambienti di sviluppo

Dipendenze tra packages

Una dipendenza tra due package sussiste se esiste una dipendenza tra almeno due classi appartenenti a ciascuno dei package in questione.

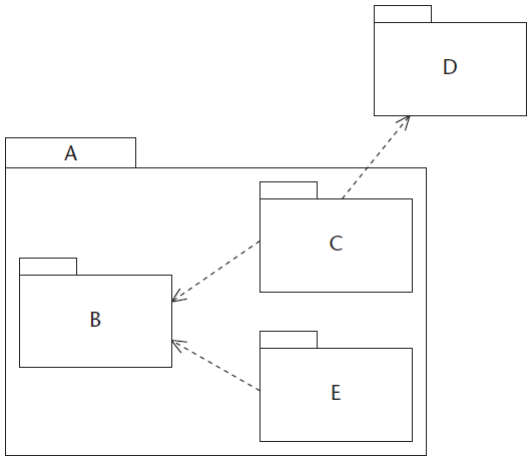
Se qualche classe in Orders cambia è opportuno verificare se qualche classe di Order Capture Application deve essere a sua volta modificata.

Teoricamente solo delle modifiche all'interfaccia di una classe dovrebbe interessare (in modifica) altre classi



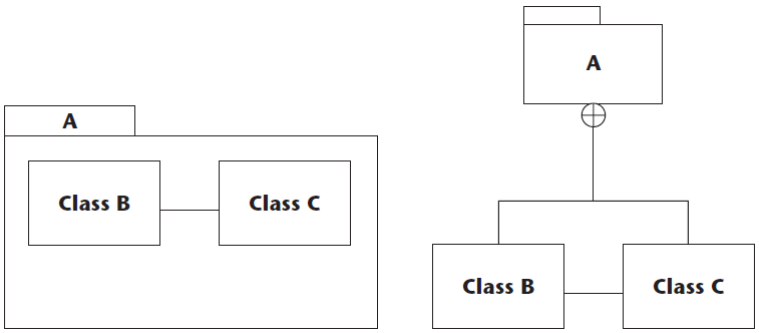
Esempi di package diagrams

Package E è in relazione di dipendenza con il package B. Package C è in relazione di dipendenza con i packages B and D. Packages B, C, and E sono inclusi nel package A.



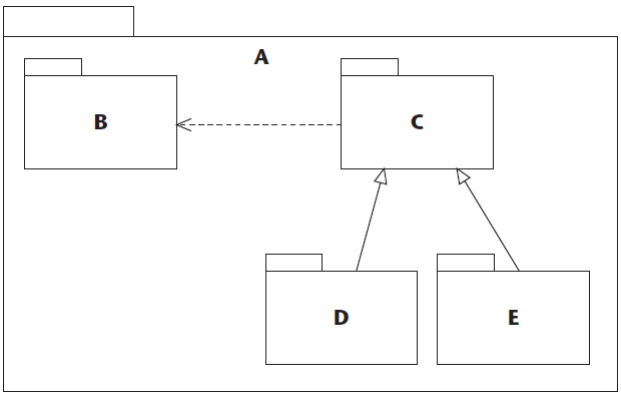
Esempi di package diagrams

Due differenti notazioni grafiche per indicare che il package A contiene le classi B e C.



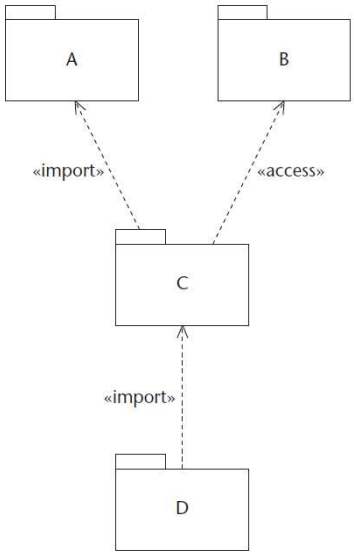
Esempi di package diagrams

Packages D and E sono in relazione di generalizzazione con il package C. Packages B, C, D, and E sono inclusi nel package A. Package C è in relazione di dipendenza con il package B.



Esempi di package diagrams

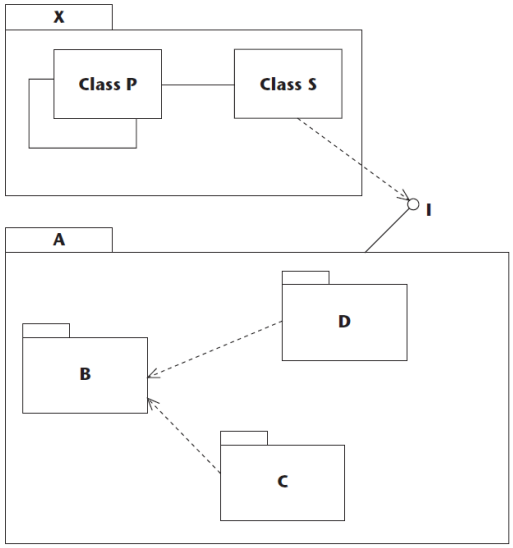
Gli elementi definiti nel package A sono importati con visibilità pubblica nel package C. Gli elementi definiti nel package B sono importati con visibilità private nel package C. Gli elementi definiti nei packages A and C sono importati con visibilità pubblica e sono disponibili nel package D.



Esempi di package diagrams

Package X contiene le classi P and S. Package A ha un'interfaccia I.

Class S all'interno del package X è in relazione di dipendenza con l'interfaccia I appartenente al package A.



Interfacce

Collezioni di operazioni che sono utilizzate per specificare un servizio di una classe o di un componente

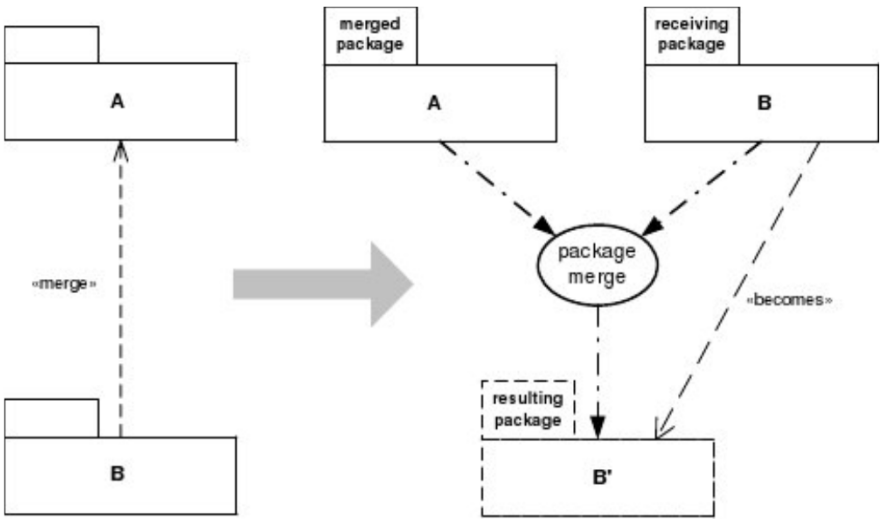
Definisce solo la segnatura delle operazioni

Le operazioni possono avere attributi di visibilità (come nelle classi)

Non tutti i linguaggi hanno interfacce

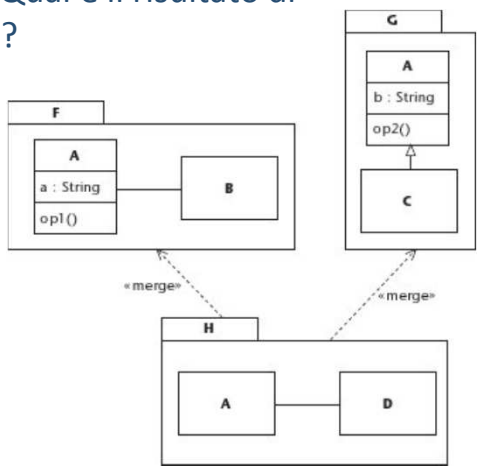
C++ NO , Java SI

Package Diagram merge



Package Diagram merge: example 1

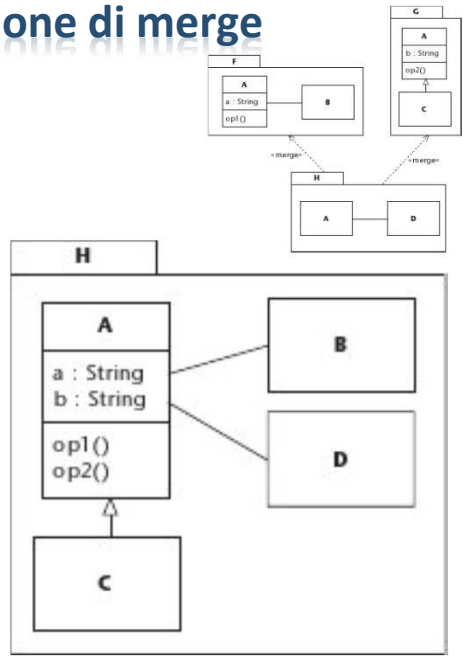
Package H unisce tutti i contenuti dei packages F and G. Qual'è il risultato di questa operazione?



Risultato dell'operazione di merge

Package H unisce tutti i contenuti dei packages F and G.

Il package H è trasformato. La classe A ha unito le classi F::A e G::A in H::A. Le classi B e C sono incluse nel package H.



Package Diagrams - conclusioni

L'organizzazione di (classi, oggetti, use cases, componenti, nodi, istanze di nodi, etc.) in Packages per un sistema software è necessaria per gestire la complessità dello stesso.

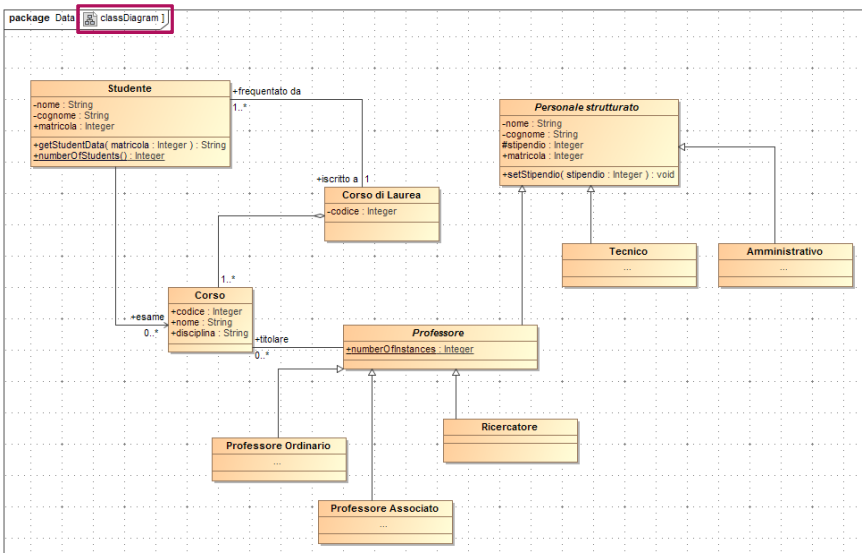
Un Package Diagram consente alle diverse entità coinvolte nello sviluppo del sistema una rapida comprensione.

In fase di Manutenzione e Testing le dipendenze definite tra i vari packages diventano di vitale importanza.

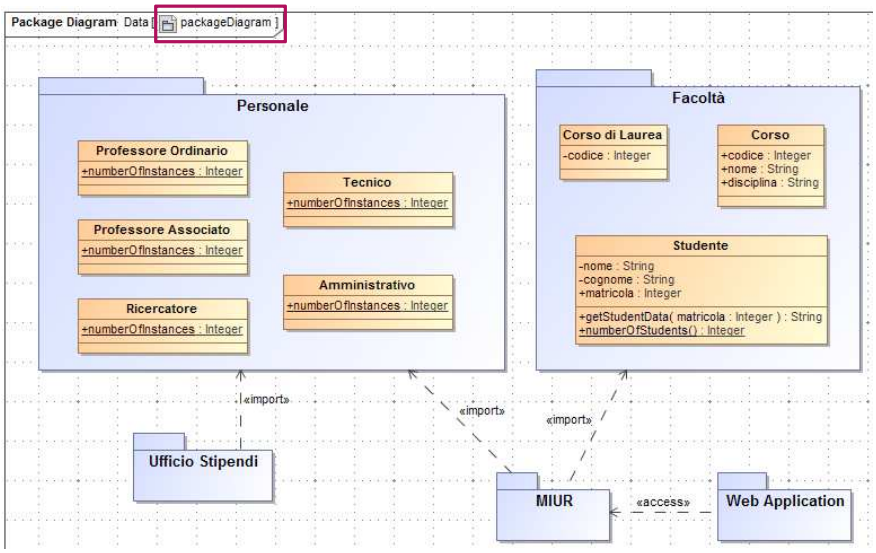
Class e Package Diagrams… proviamoci!

Si vogliono modellare gli studenti (con nome, cognome, numero di matricola), il corso di laurea in cui sono iscritti, ed i corsi di cui hanno sostenuto l'esame. Di ogni corso interessa il codice, il nome, e la disciplina a cui appartiene (ad esempio: matematica, fisica, informatica, ecc.). Di ogni professore interessa nome, cognome, il codice del corso di cui è titolare. Di ogni corso di laurea interessa il codice, il numero di professori ordinari, professori associati, ricercatori. Infine è necessario modellare il personale tecnico e amministrativo allo scopo di poter fare statistiche sui bilanci di ogni ateneo.

Class Diagram



Package Diagram



Questions?

catia.trubiani@univaq.it