Package diagram

moving towards detailed, low level design

Design Levels

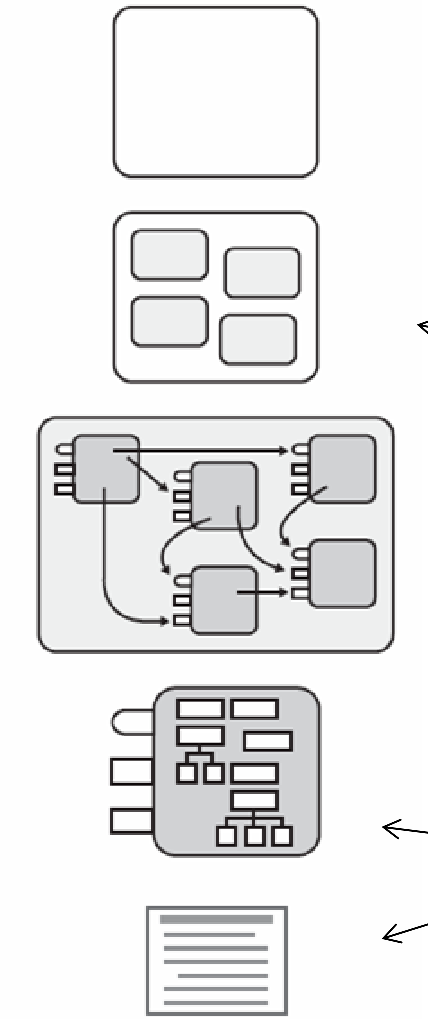
System

Sub-Systems

Packages

Classes

Methods



High-Level Design

Low-Level Design

Package diagram

- **Package** is a **namespace** used to group together elements that are semantically related and might change together.
- It is a general purpose mechanism to organize elements into groups to provide better structure for system model.
- Is the package structure of program or system
- Sometimes a direct translation into OOP languages (package in Java, namespaces in C++)

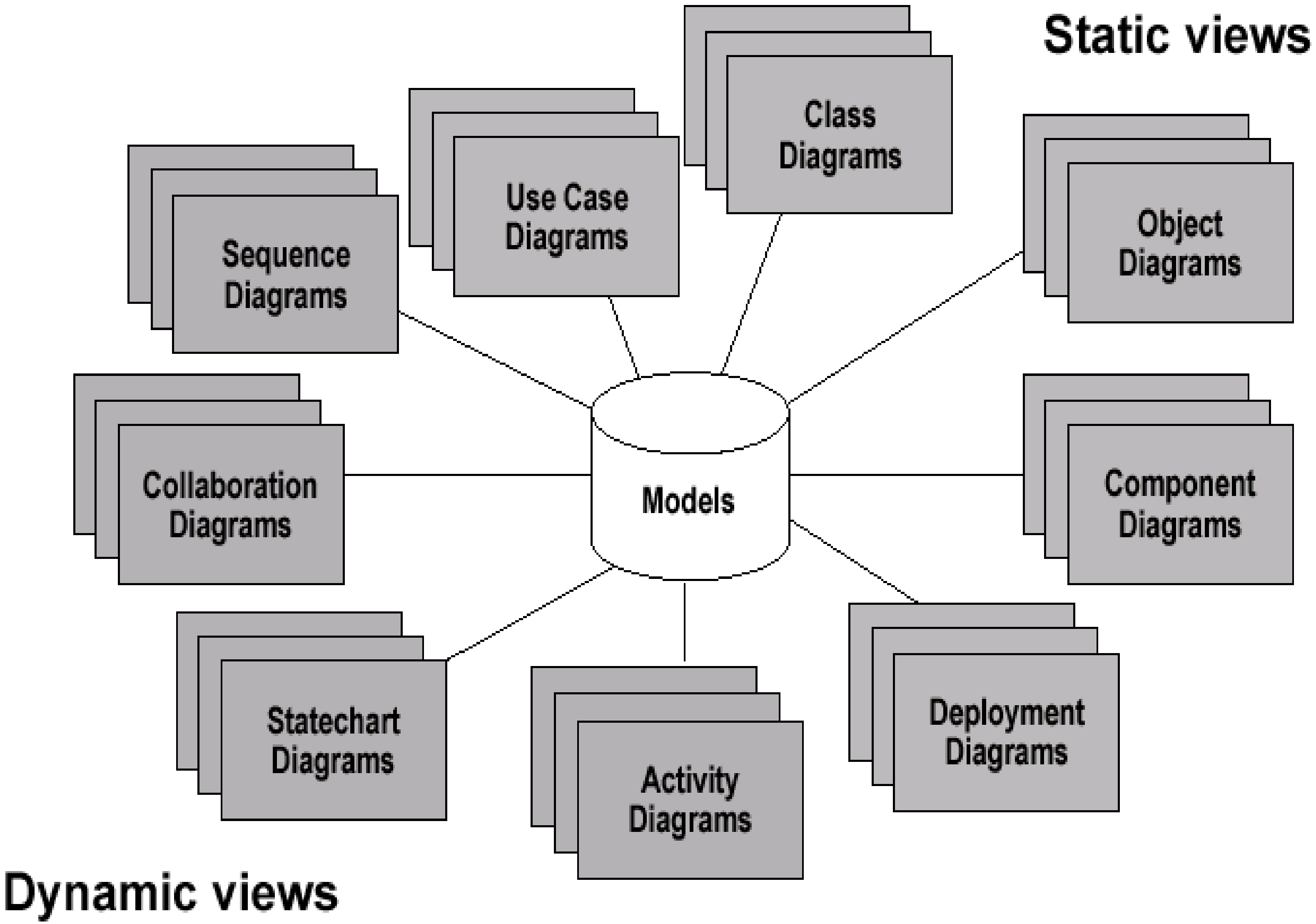
Package Diagrams

- A package diagram can be applied to any UML diagram.
- A package is a construct that enables you to organize model elements, such as use cases or classes, into groups.
- Create a package diagram to:

1. Depict a high-level overview of your requirements (overviewing a collection of UML Use Case diagrams)
2. Depict a high-level overview of your architecture/design (overviewing a collection of UML Class diagrams).
3. To logically modularize a complex diagram.

Models, Views, Diagrams

(Package Diagram)



Package Diagrams

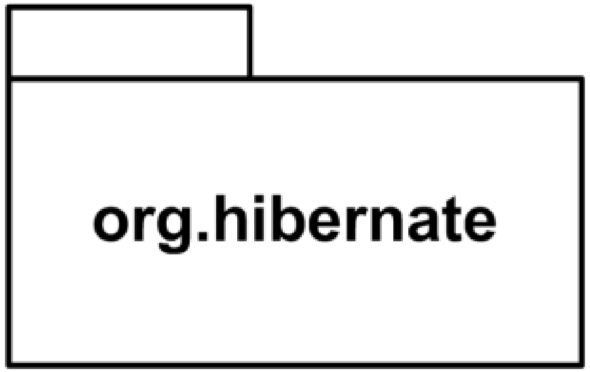
- Packages appear as rectangles with small tabs (file folder) at the top.
- The package name is on the tab or inside the rectangle.
- The dotted arrows are dependencies. ← — — — —
- One package depends on another if changes in the other could possibly force changes in the first. coupling?
- Packages are the basic grouping construct with which you may organize UML models to increase their readability

General Guidelines

- Give packages simple, descriptive names
- Apply packages to simplify diagrams
- Packages should be cohesive
- Indicate architectural layers with stereotypes on packages
- Avoid cyclic dependencies between packages
- Package dependencies should reflect internal relationships

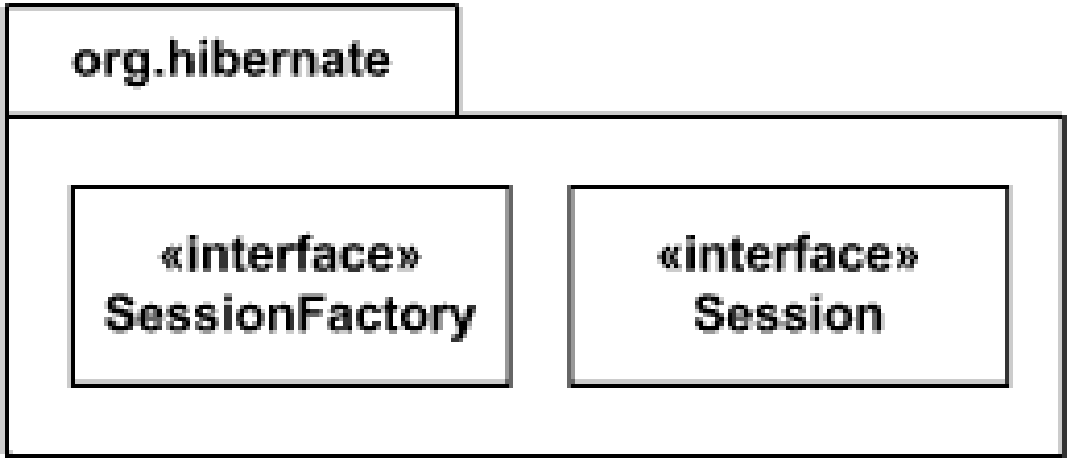
Package:

- Package member(s) are not shown inside the package. but can be shown (see next slide)
- Package org.hibernate



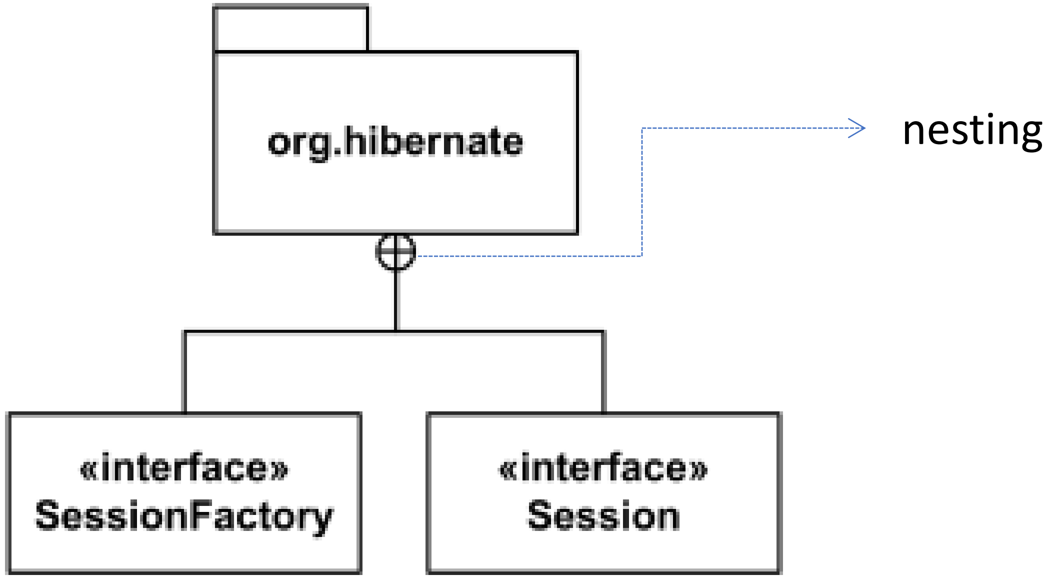
Package:

- Package org.hibernate contains SessionFactory and Session.
- Package Member can be shown inside the package.



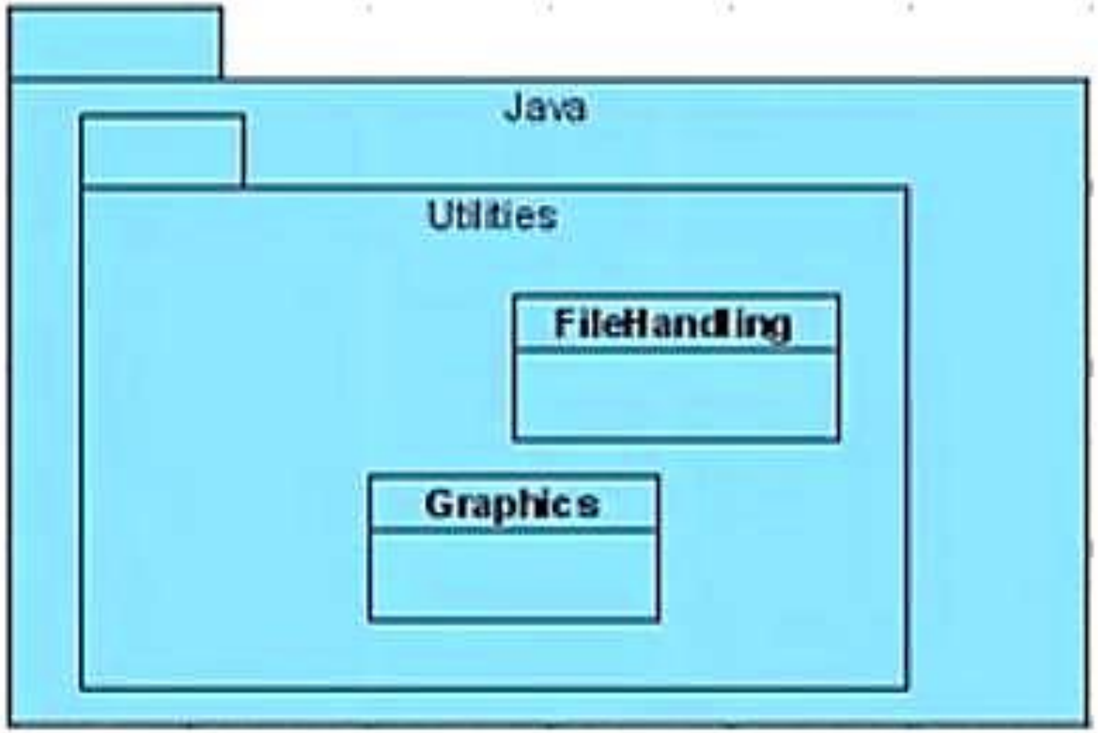
Package:

- Members of the package may be shown **outside** of the package by branching lines.
- Package org.hibernate contains interfaces Session and SessionFactory.



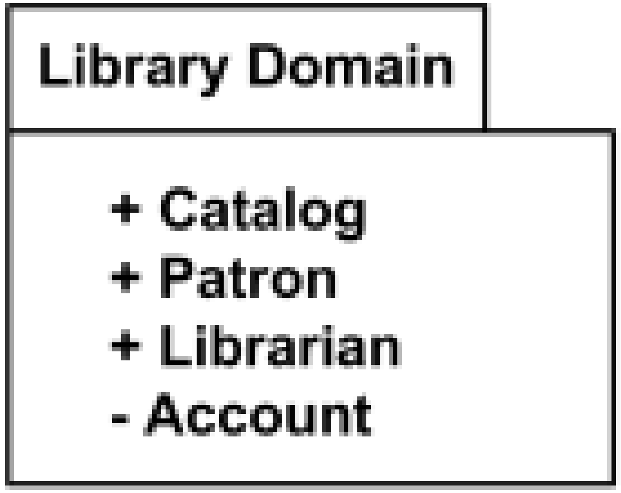
Package:

- Nested packages.
- Qualifier for Graphics class is Java::Utilities::Graphics

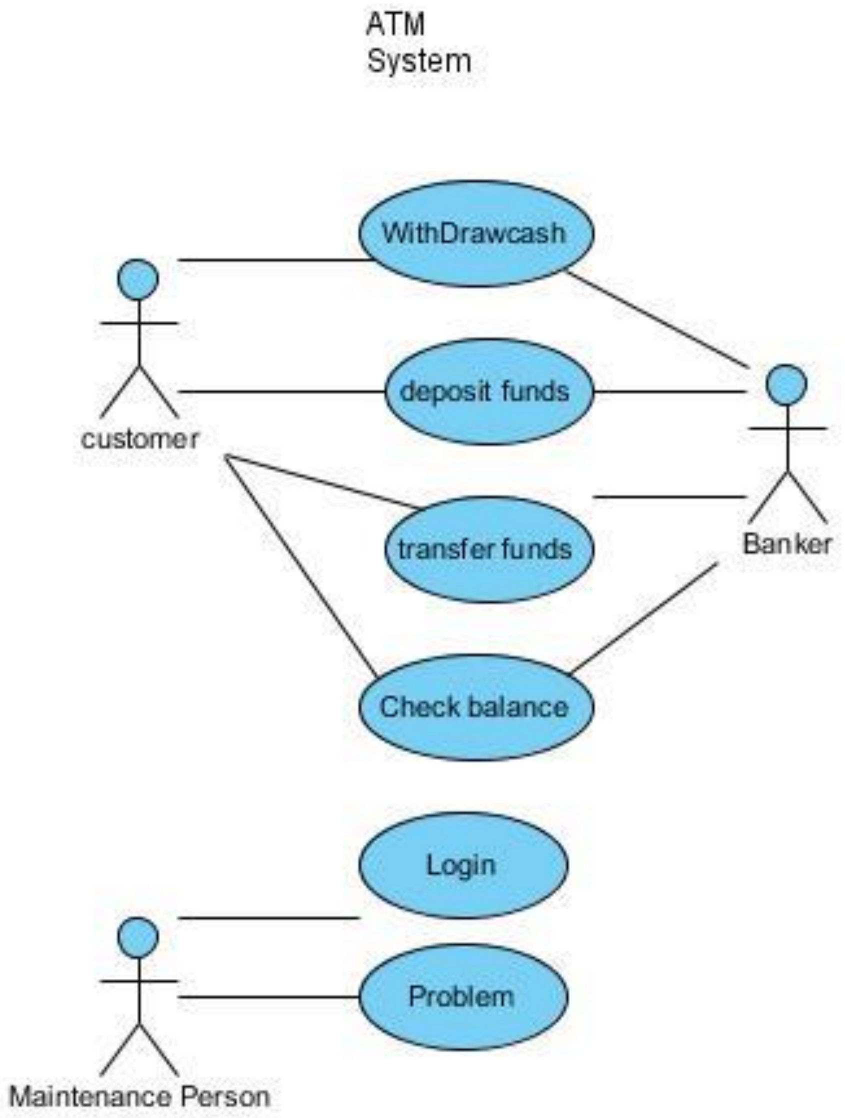


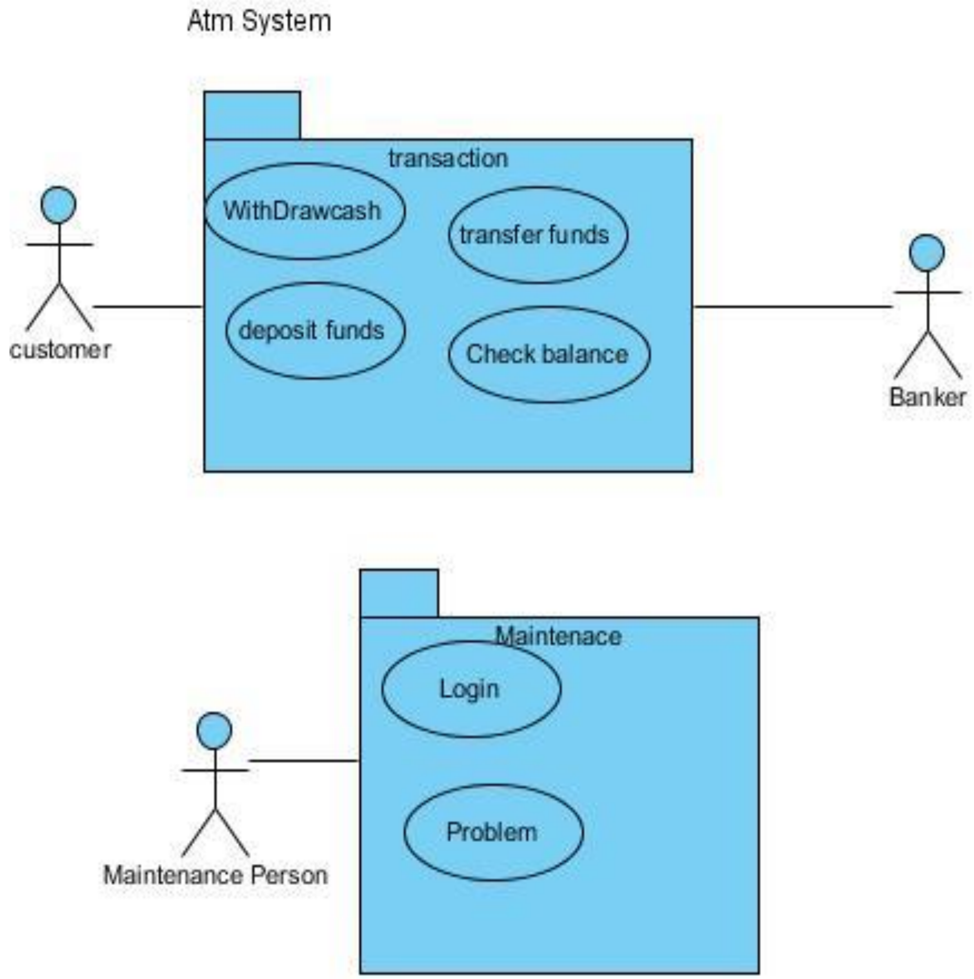
Package:

- **Visibility** of Owned and Import element.
- "+" for public and "-" for private or helper class.
- All elements of Library Domain package are public except for Account.



Use Case Package Diagram(example):

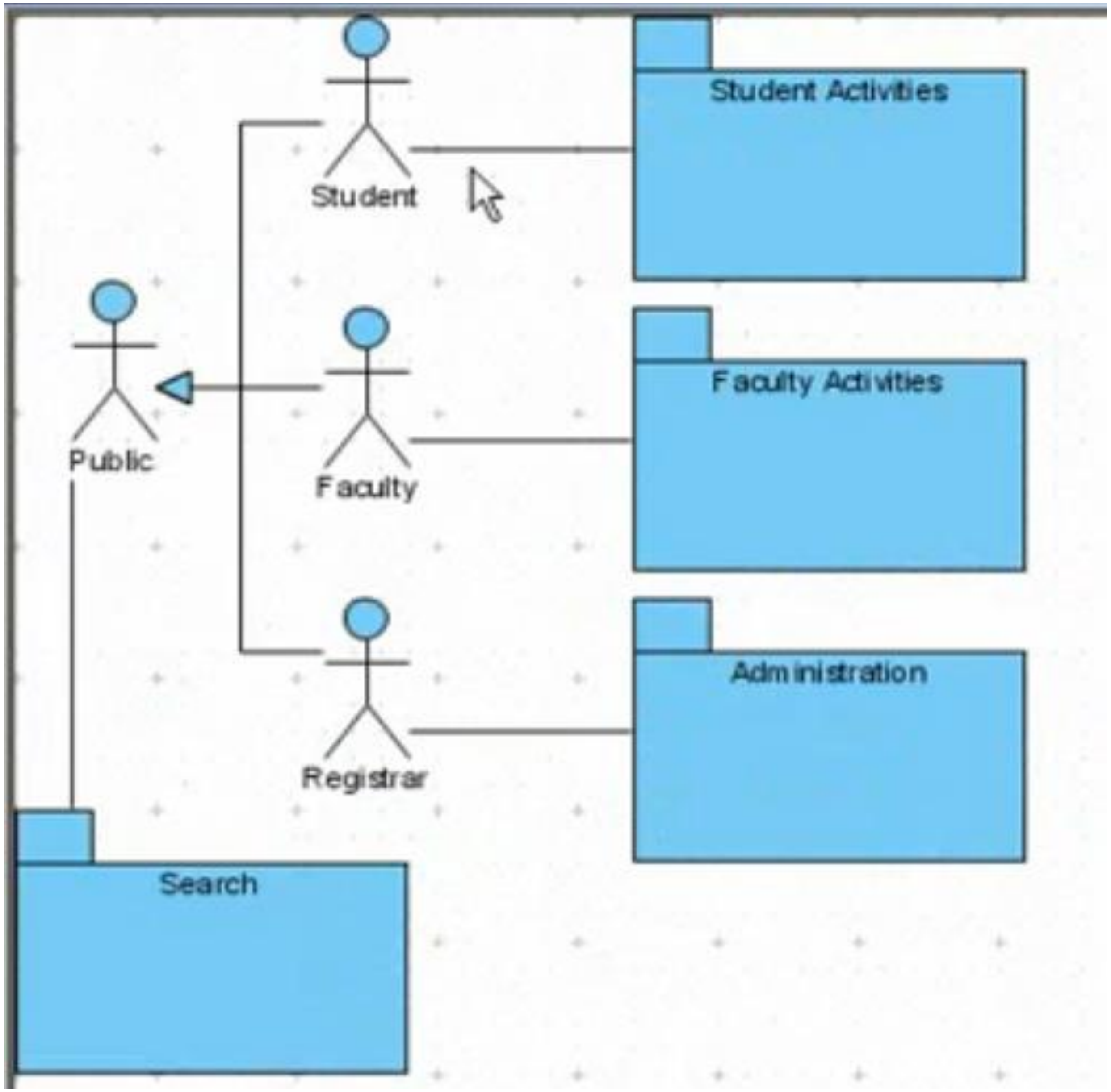




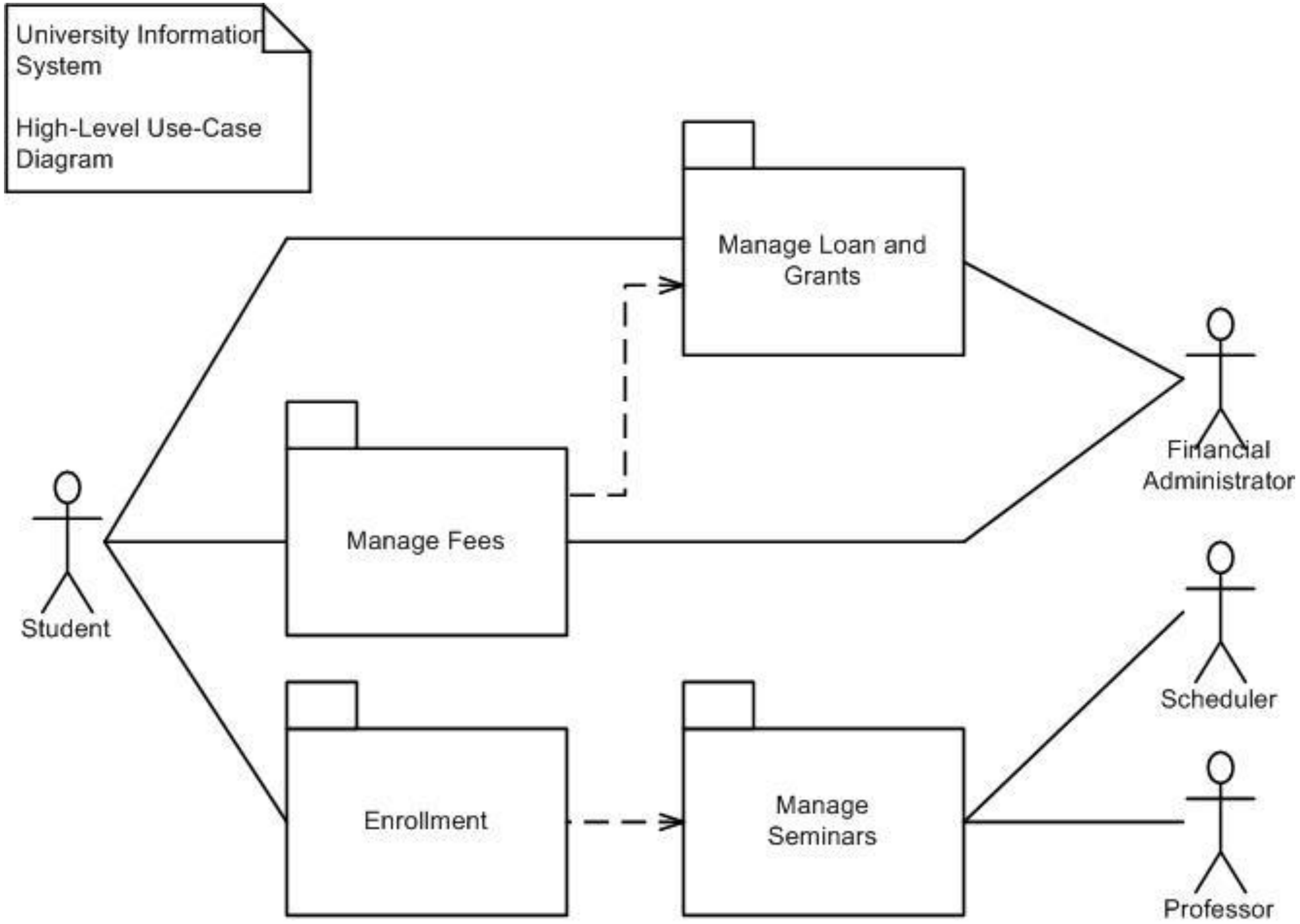
Use case Diagram ➔➔

Use case Package Diagram

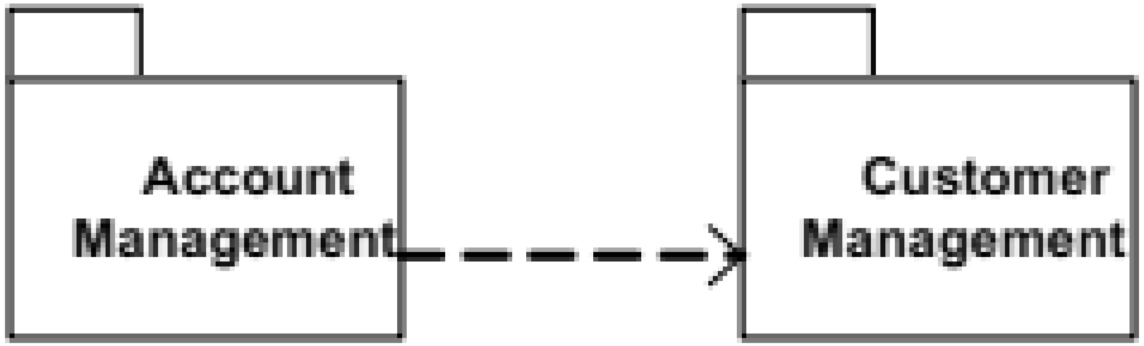
Use Case Package Diagram (example):



Use Case Package Example

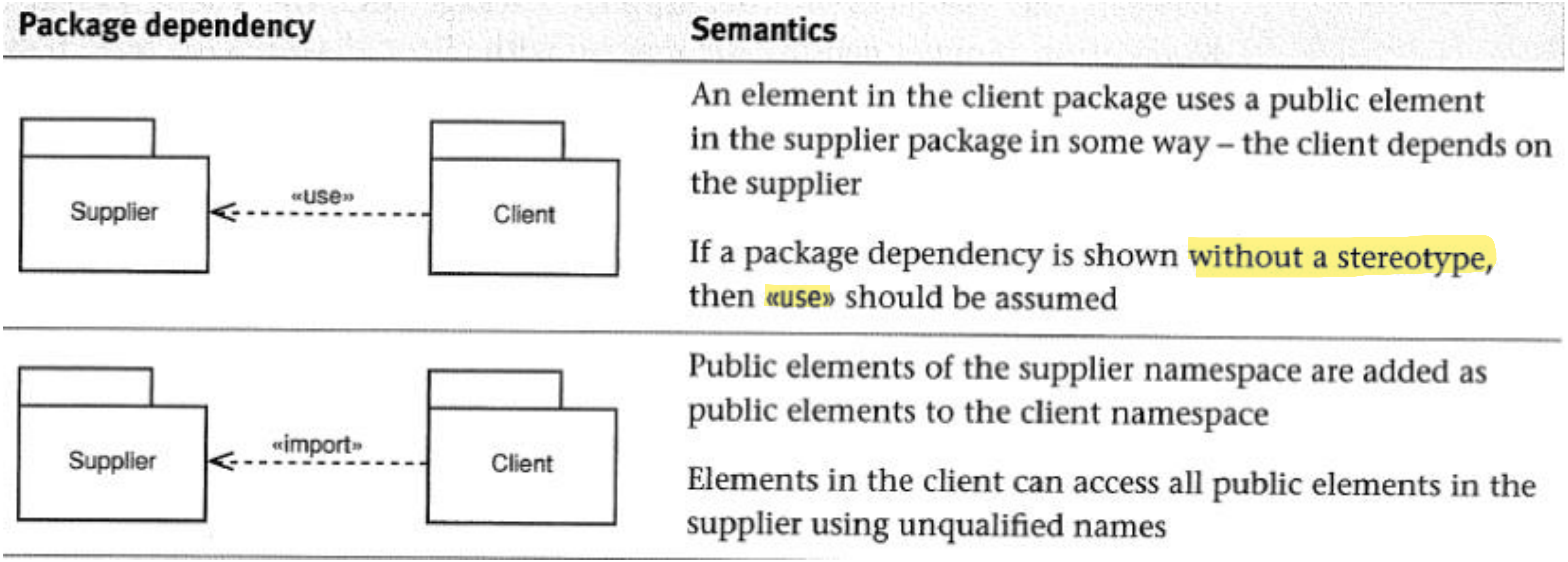


Package dependency



- A package A depends on package B if at least one element of A depends on one element of B
- The change in a dependent package can lead to changes in the using package

Package dependency



Package dependency	Semantics
Supplier ←«use»- - - Client	An element in the client package uses a public element in the supplier package in some way – the client depends on the supplier If a package dependency is shown without a stereotype, then «use» should be assumed
Supplier ←«import»- - - Client	Public elements of the supplier namespace are added as public elements to the client namespace Elements in the client can access all public elements in the supplier using unqualified names

<<import>> : public import
<<access>> : private import

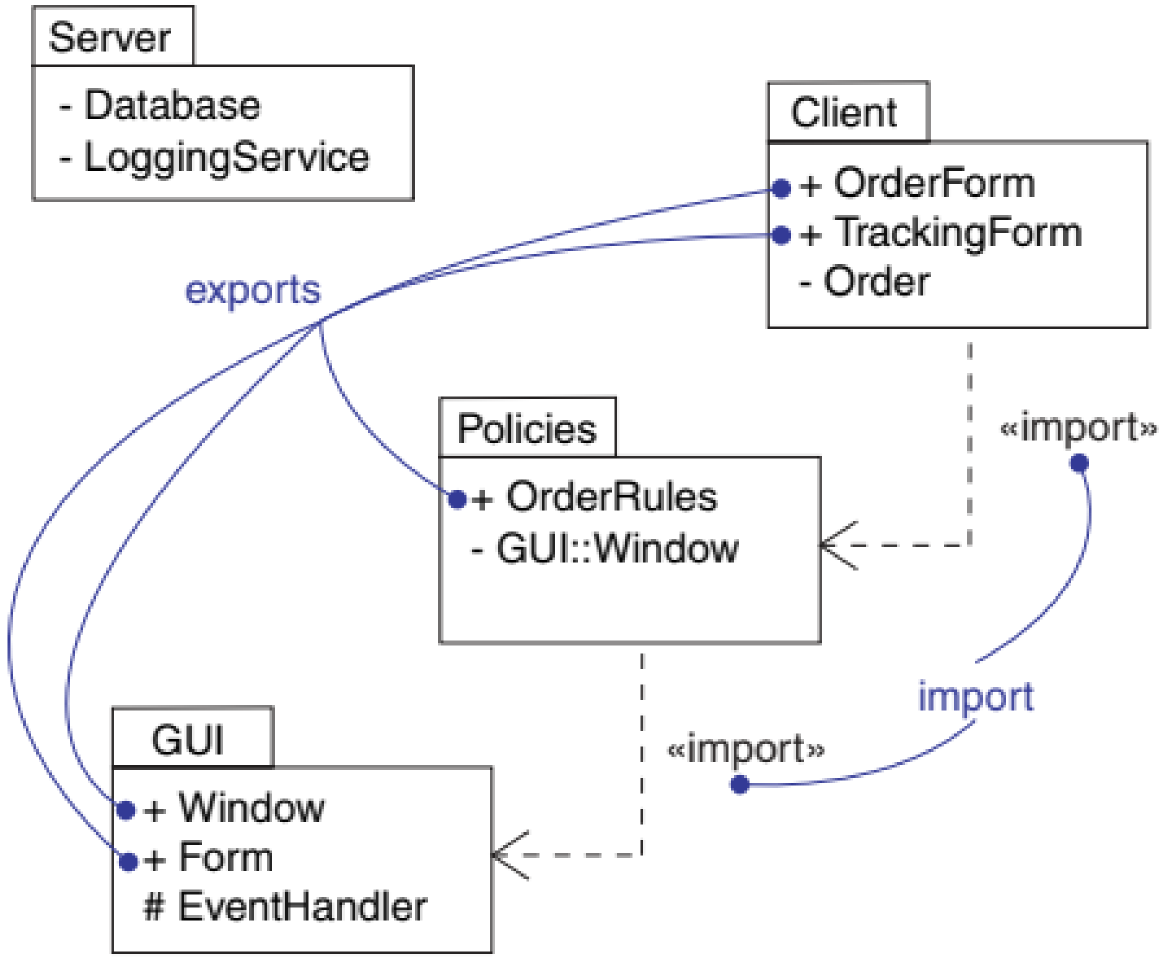
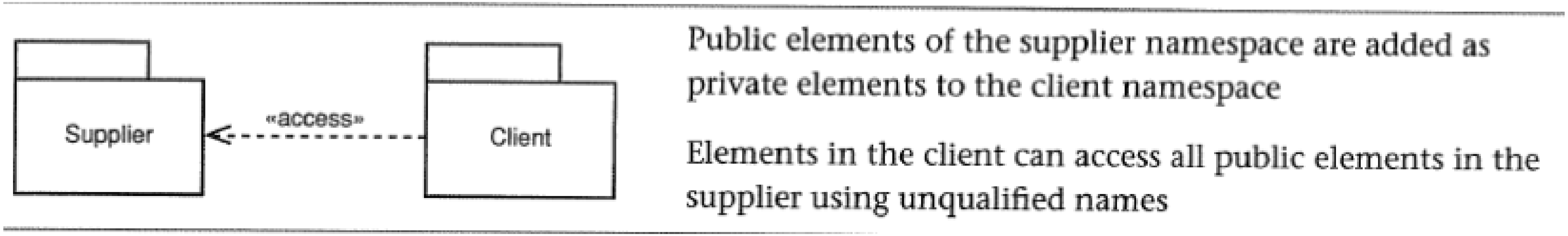


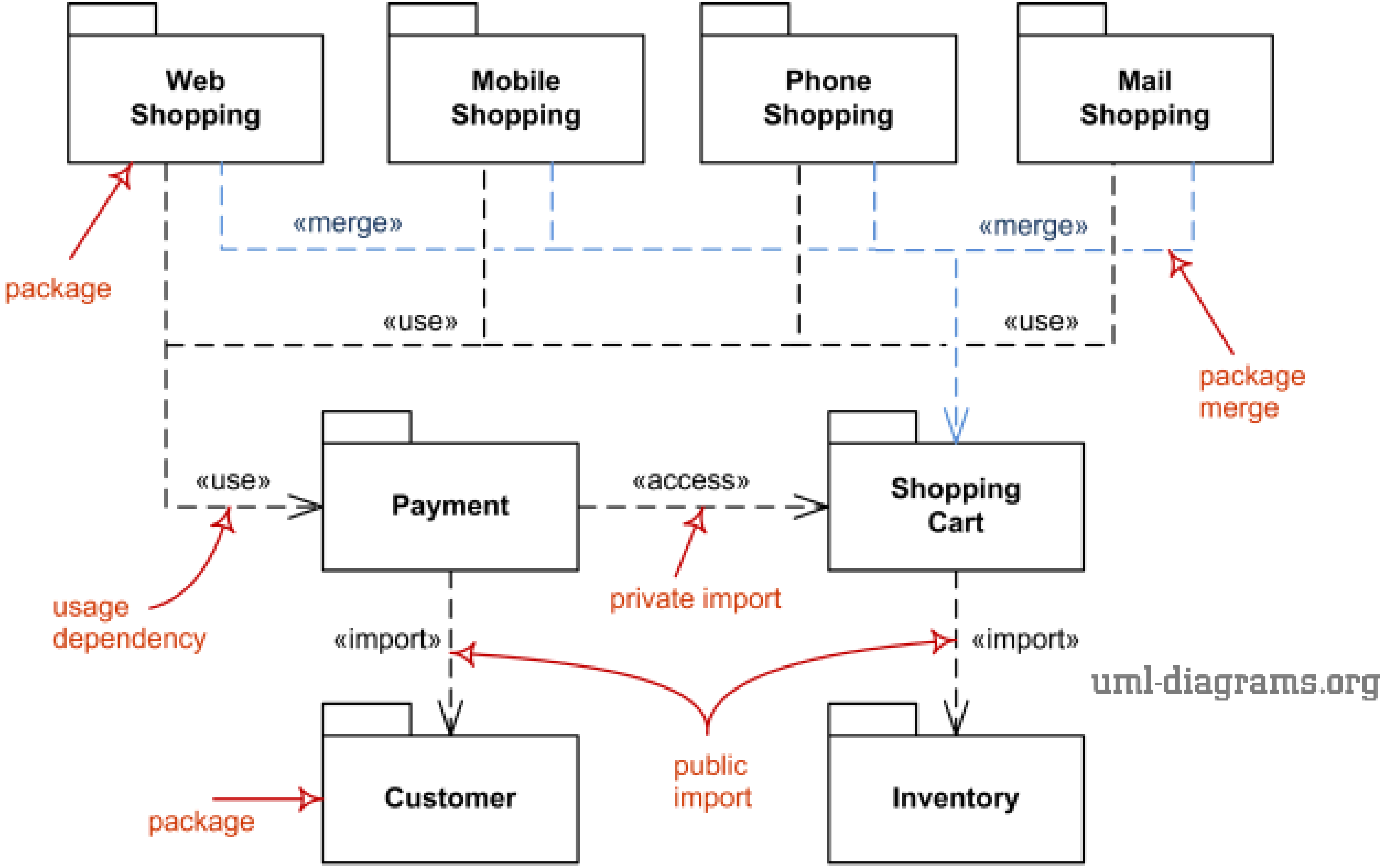
Figure 12-4: Importing and Exporting

Package dependency


Supplier
«access»
Client

Public elements of the supplier namespace are added as private elements to the client namespace

Elements in the client can access all public elements in the supplier using unqualified names


Web Shopping
Mobile Shopping
Phone Shopping
Mail Shopping
«merge»
«merge»
package
«use»
«use»
package merge
«use»
Payment
«access»
Shopping Cart
usage dependency
private import
«import»
«import»
uml-diagrams.org
Customer
public import
Inventory
package

Transitivity

if a imports b and b imports c then a imports c

but the same is not true for <<access>>

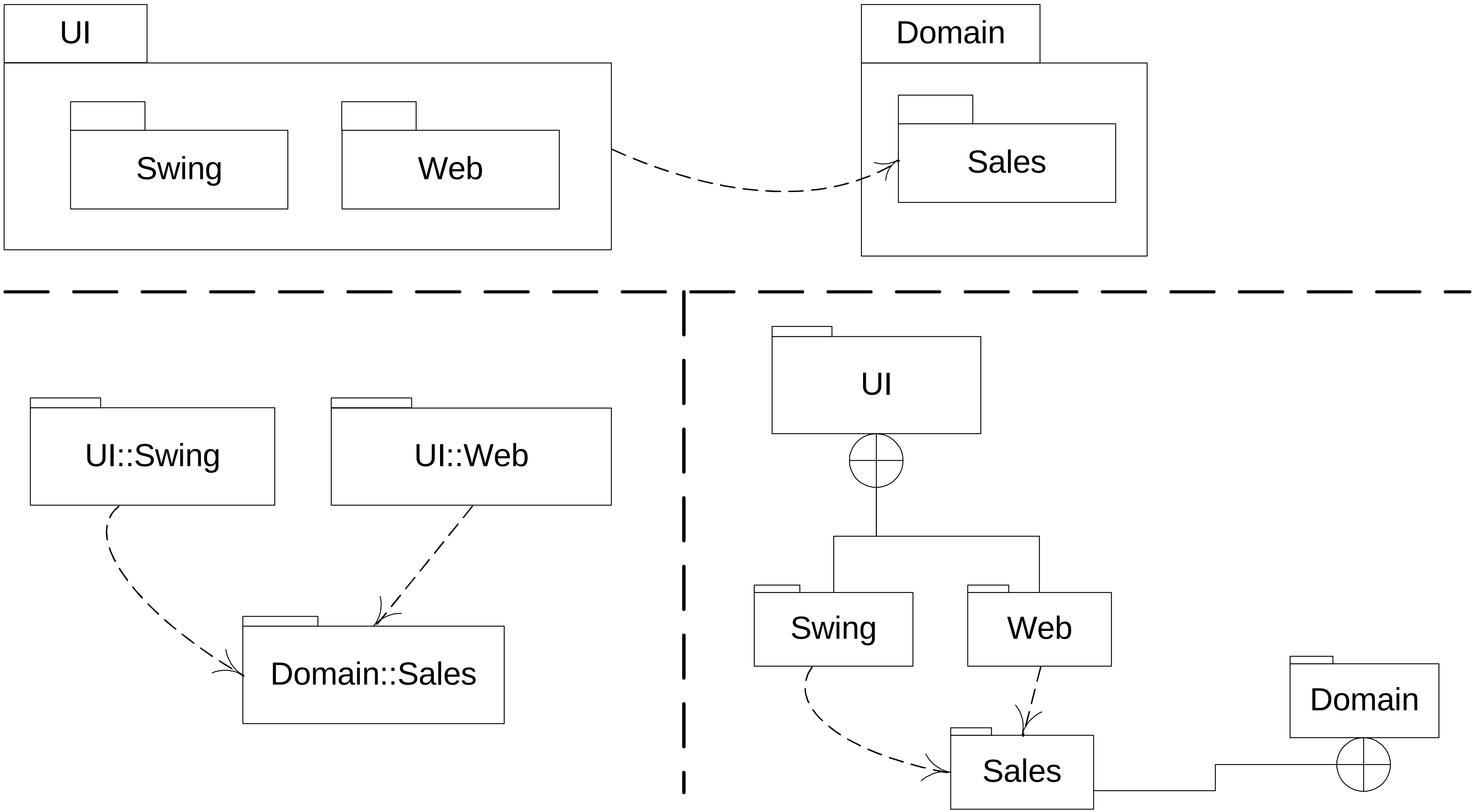
- The transitivity rule states that if A is related to B, and B is related to C, then it follows that A is related to C.
- The relationship is usually depicted as follows:
    - If A => B, and B => C, then by transitivity A => C.
- In package dependency the following applies:
- <<import>> dependency is transitive; that is to say
    - If package A <<import>> B, and B <<import >> C, then A <<import>> C
- However, the <<access>> dependency is not transitive.

Transitivity

- If package A accesses package B, and package B accesses package C.

- The following applies by the <<access>> dependency definition:
- Public elements in C become private elements in B;
- Public elements in B become private elements in A;
- This implies elements in A can’t access elements in C.

Package Nesting (Subpackages)



Class Package Diagrams

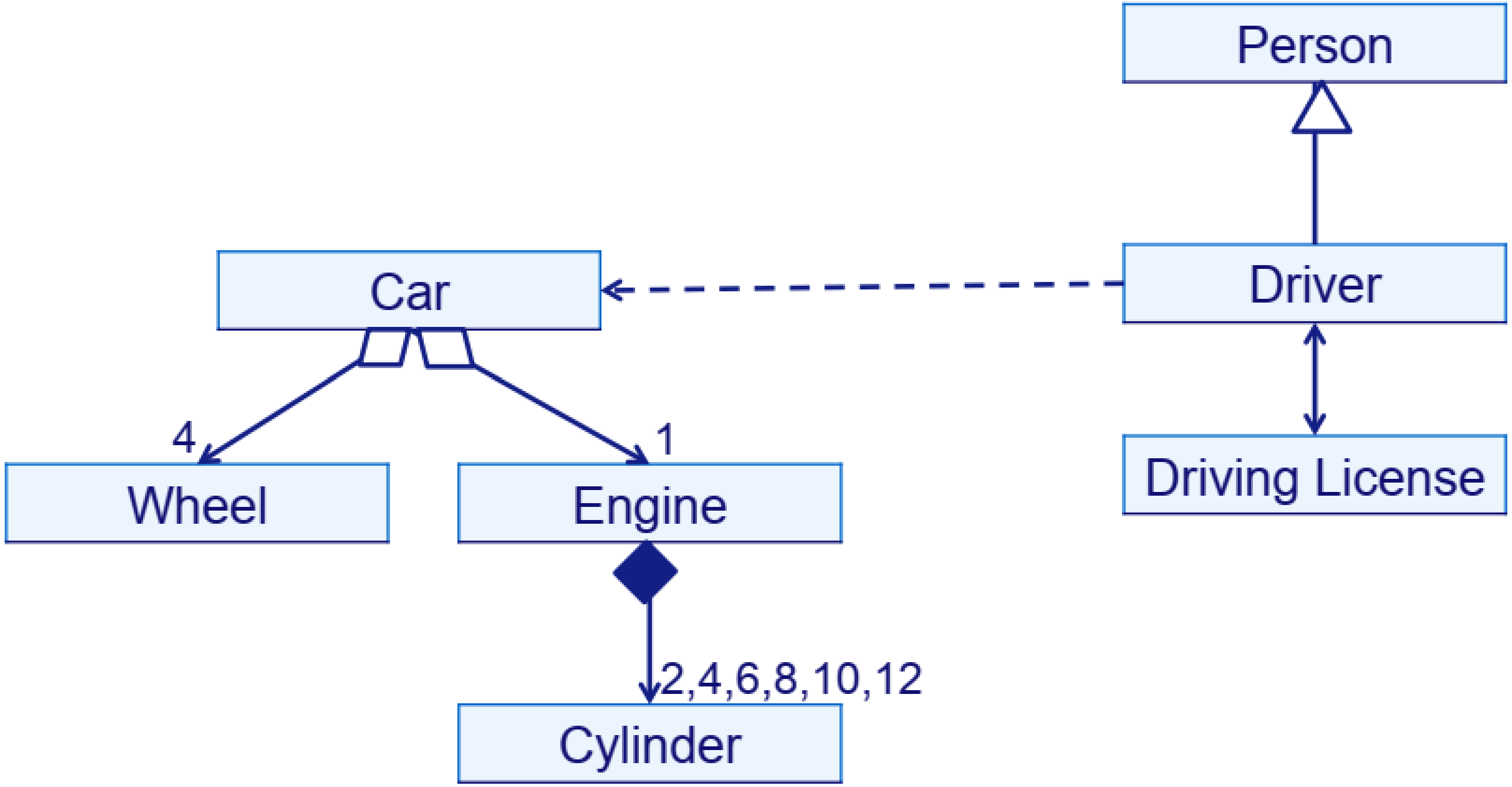
Heuristics to organize classes into packages:

- Classes of a framework belong in the same package.
- Classes in the same inheritance hierarchy typically belong in the same package.
- Classes related to one another via aggregation or composition often belong in the same package.
- Classes that collaborate with each other a lot often belong in the same package.

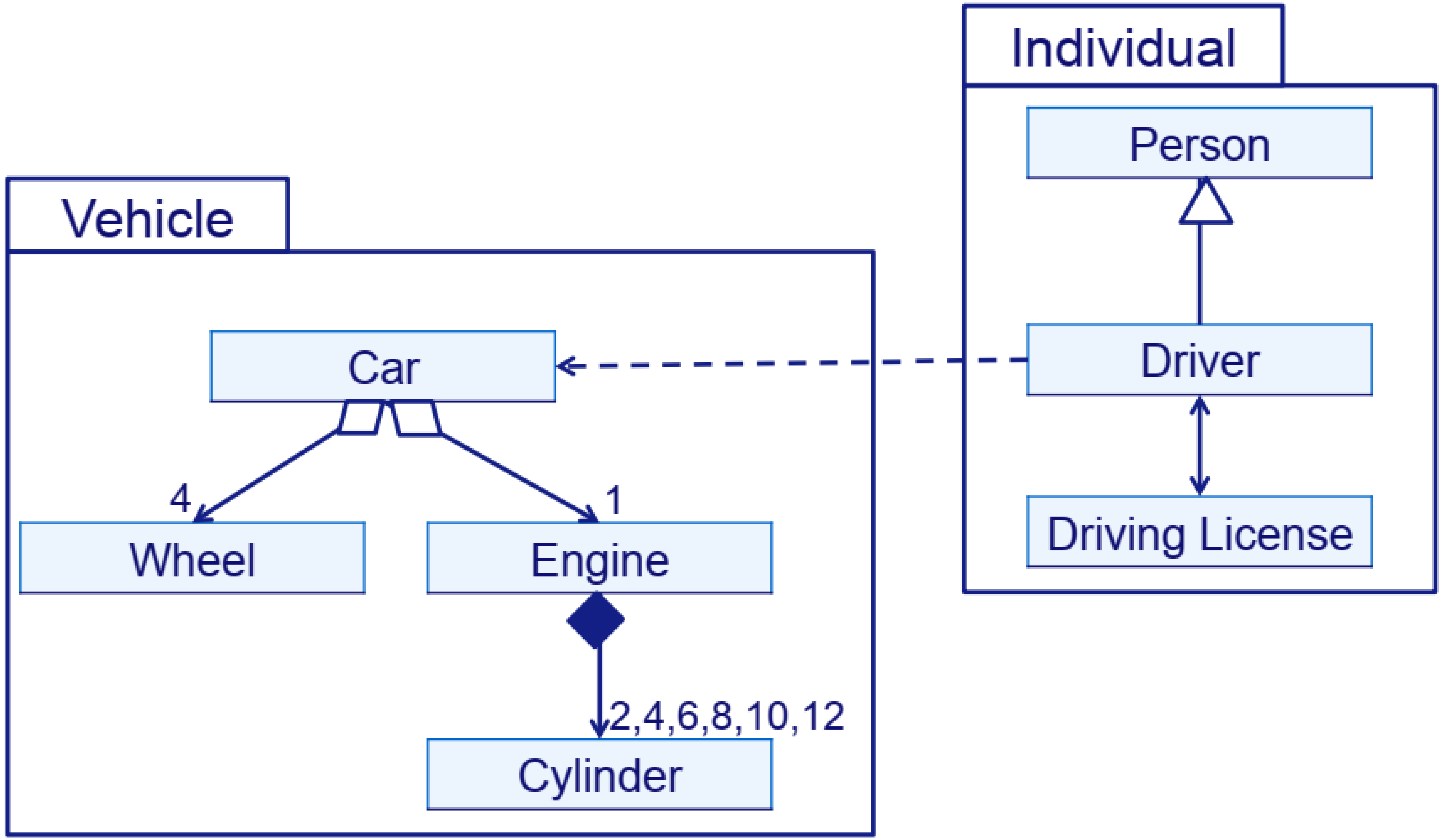
How would you organize into 2 packages?

- Car, Cylinder, Driver, Driving License, Engine, Person, Wheel
- Start by drawing a class diagram

How would you organize into 2 packages?

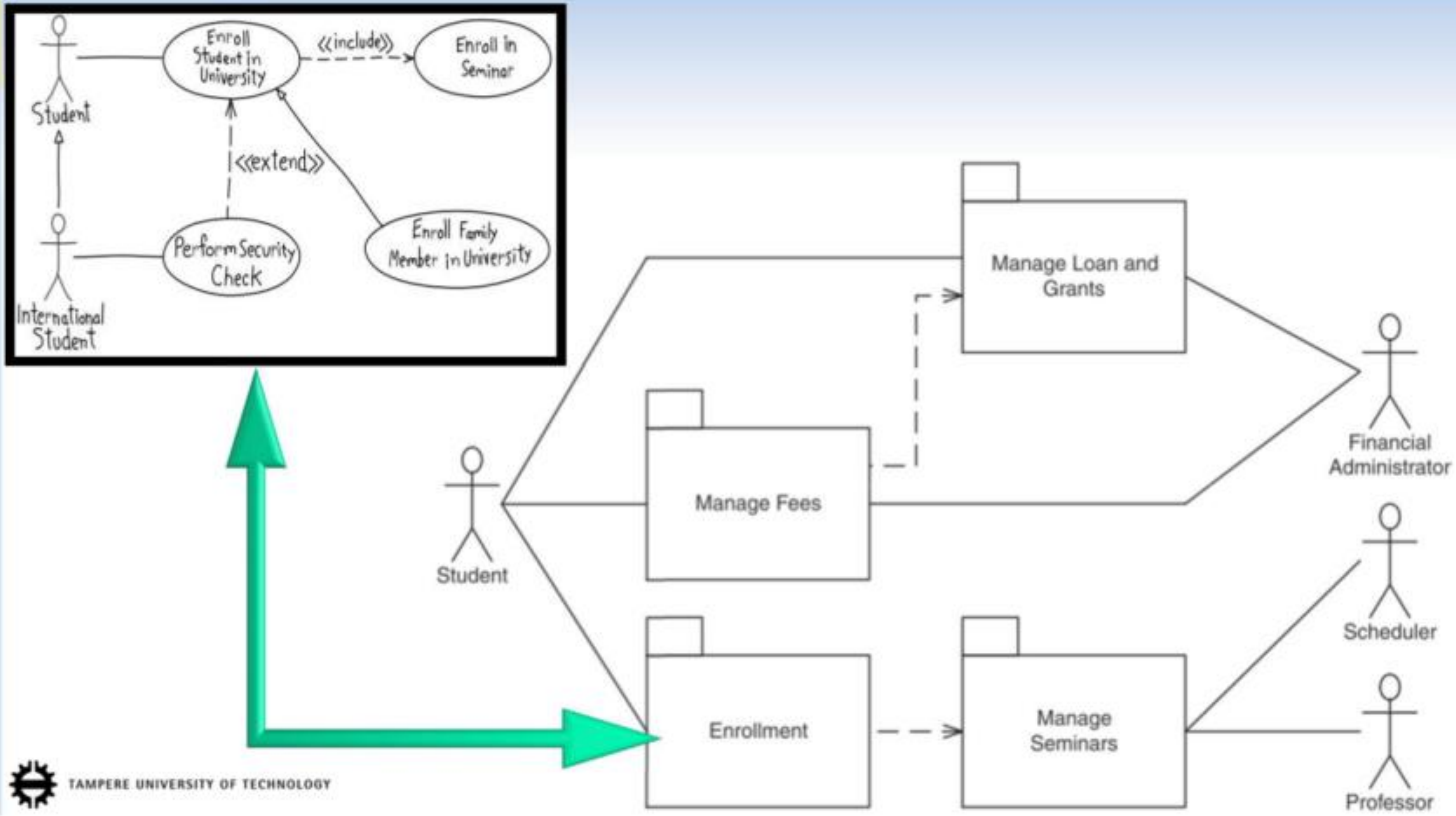


How would you organize into 2 packages?



Use-Case Package Diagrams

- Heuristics to organize use cases into packages:
- Keep associated use cases together: included, extending and inheriting use cases belong in the same package.
- Group use cases on the basis of the needs of the main actors.


Student
Enroll Student in University
<<include>>
Enroll In Seminar
<<extend>>
International Student
Perform Security Check
Enroll Family Member in University
Student
Manage Loan and Grants
Manage Fees
Enrollment
Manage Seminars
Financial Administrator
Scheduler
Professor