

IT4490 - SOFTWARE DESIGN AND CONSTRUCTION

6. IDENTIFY DESIGN ELEMENTS

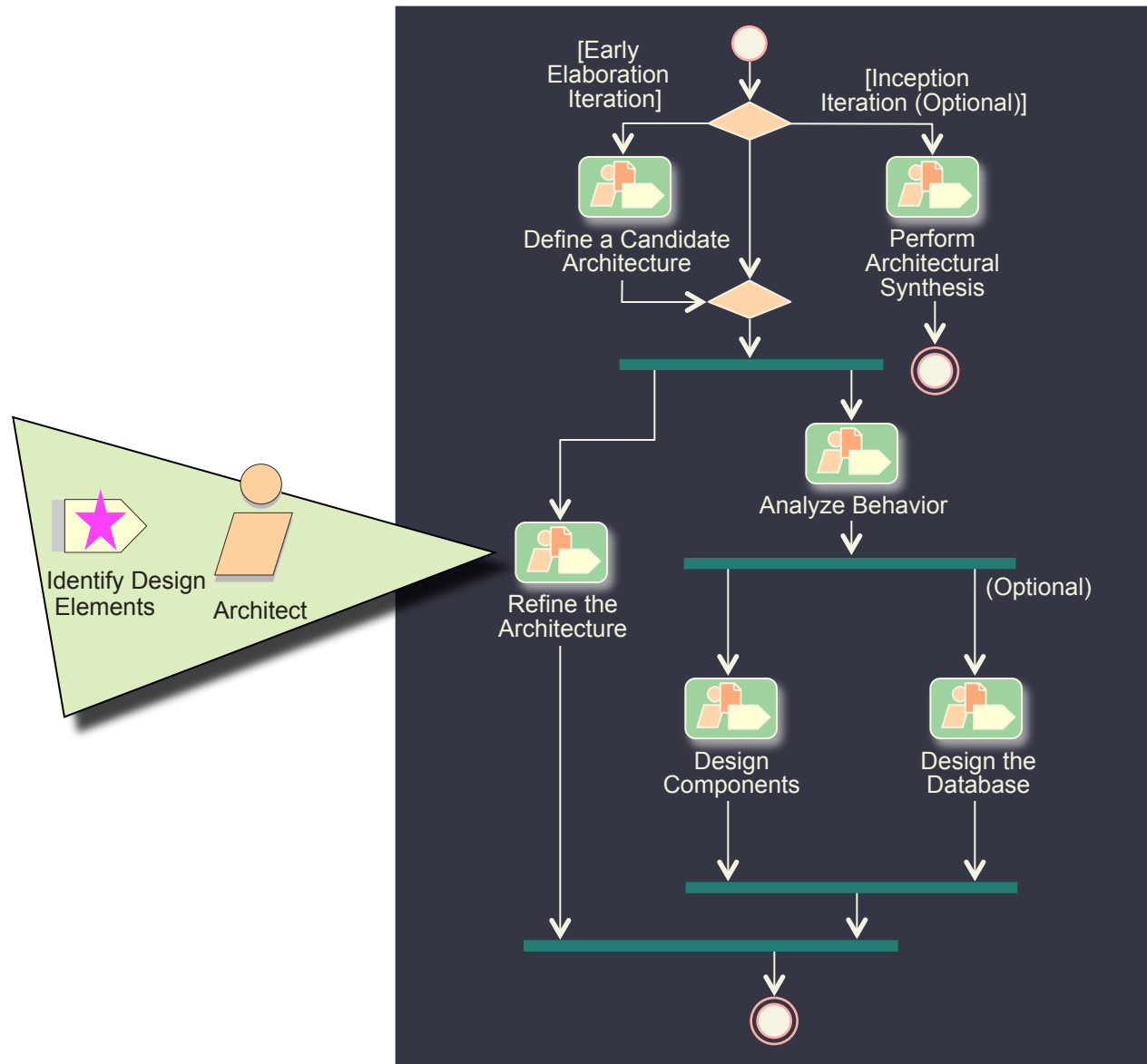
GV: Nguyen Nhat Hai
hainn@soict.hust.edu.vn



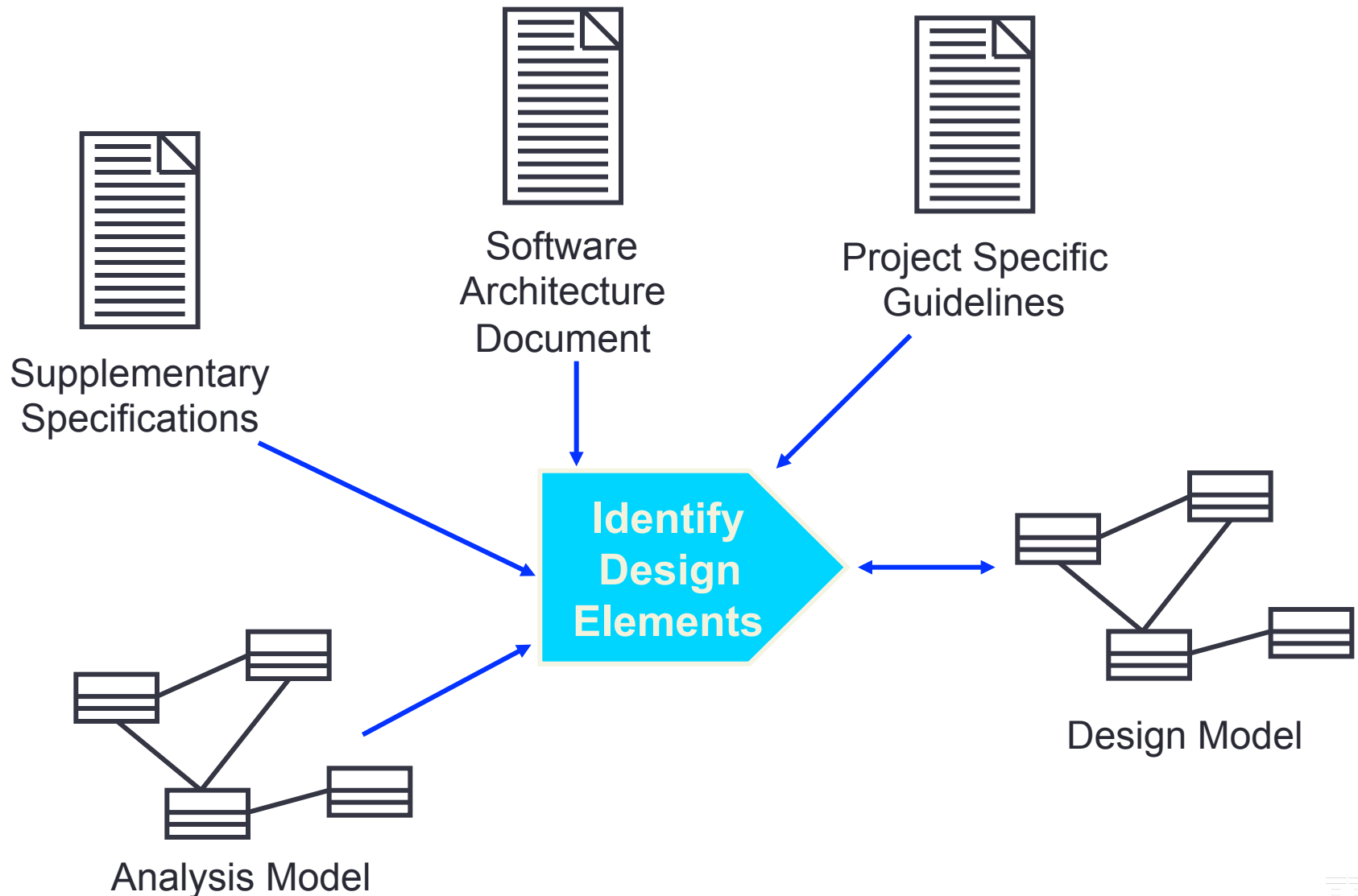
Objectives: Identify Design Elements

- Define the purpose of Identify Design Elements and demonstrate where in the lifecycle it is performed
- Analyze interactions of analysis classes and identify Design Model elements => Design classes

Identify Design Elements in Context

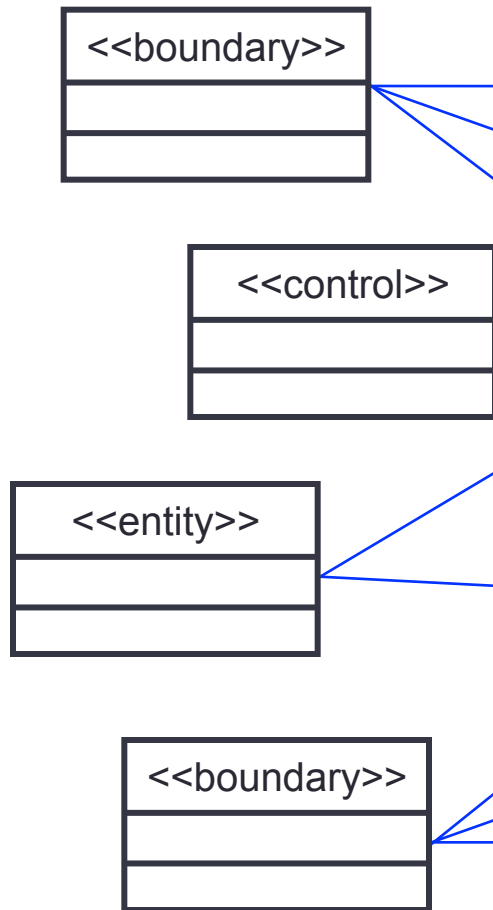


Identify Design Elements Overview

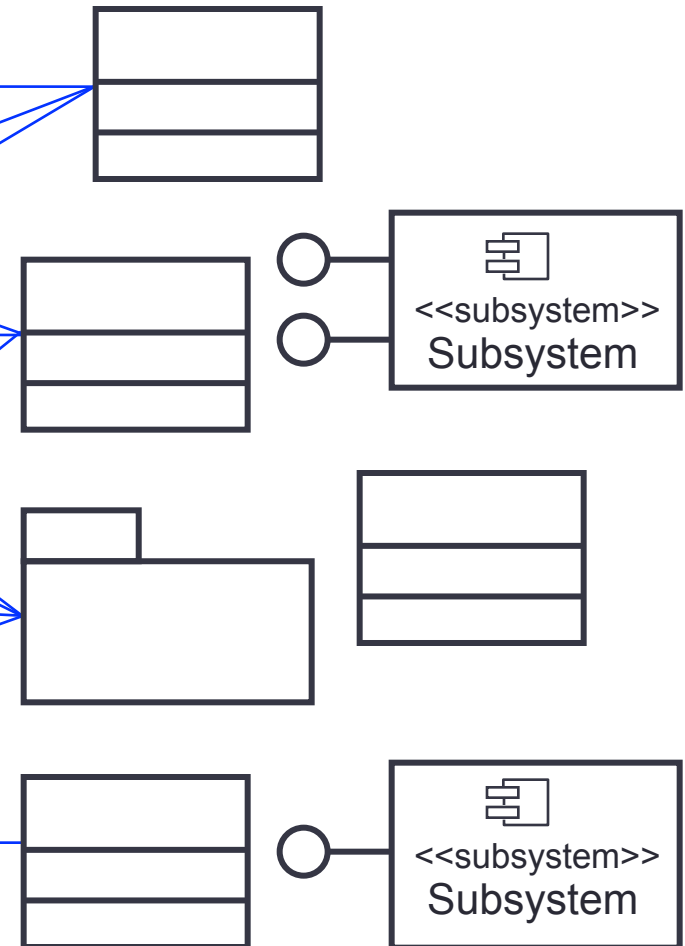


From Analysis Classes to Design Elements

Analysis Classes



Design Elements



Many-to-Many Mapping

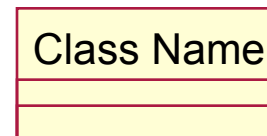
Identifying Design Classes

- An analysis class maps directly to a design class if:
 - It is a simple class
 - It represents a single logical abstraction
- More complex analysis classes may
 - Split into multiple classes
 - Become a package
 - Become a subsystem (discussed later)
 - Any combination ...

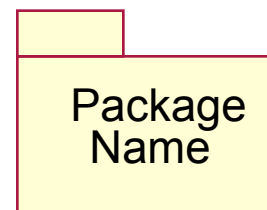


Review: Class and Package

- What is a class?
 - A description of a set of objects that share the same responsibilities, relationships, operations, attributes, and semantics

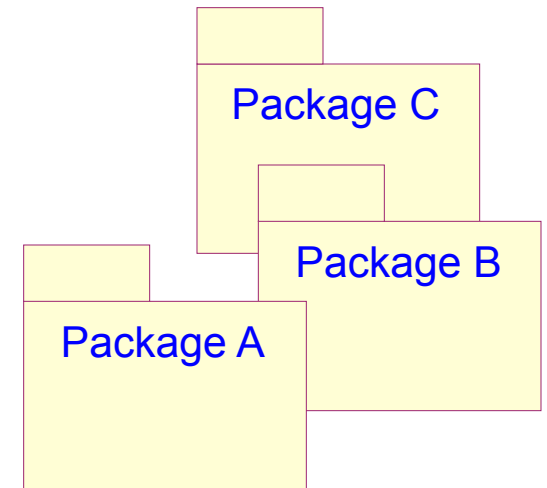


- What is a package?
 - A general purpose mechanism for organizing elements into groups
 - A model element which can contain other model elements



Group Design Classes in Packages

- You can base your packaging criteria on a number of different factors, including:
 - Configuration units
 - Allocation of resources among development teams
 - Reflect the user types
 - Represent the existing products and services the system uses



Packaging Tips:

Functionally Related Classes

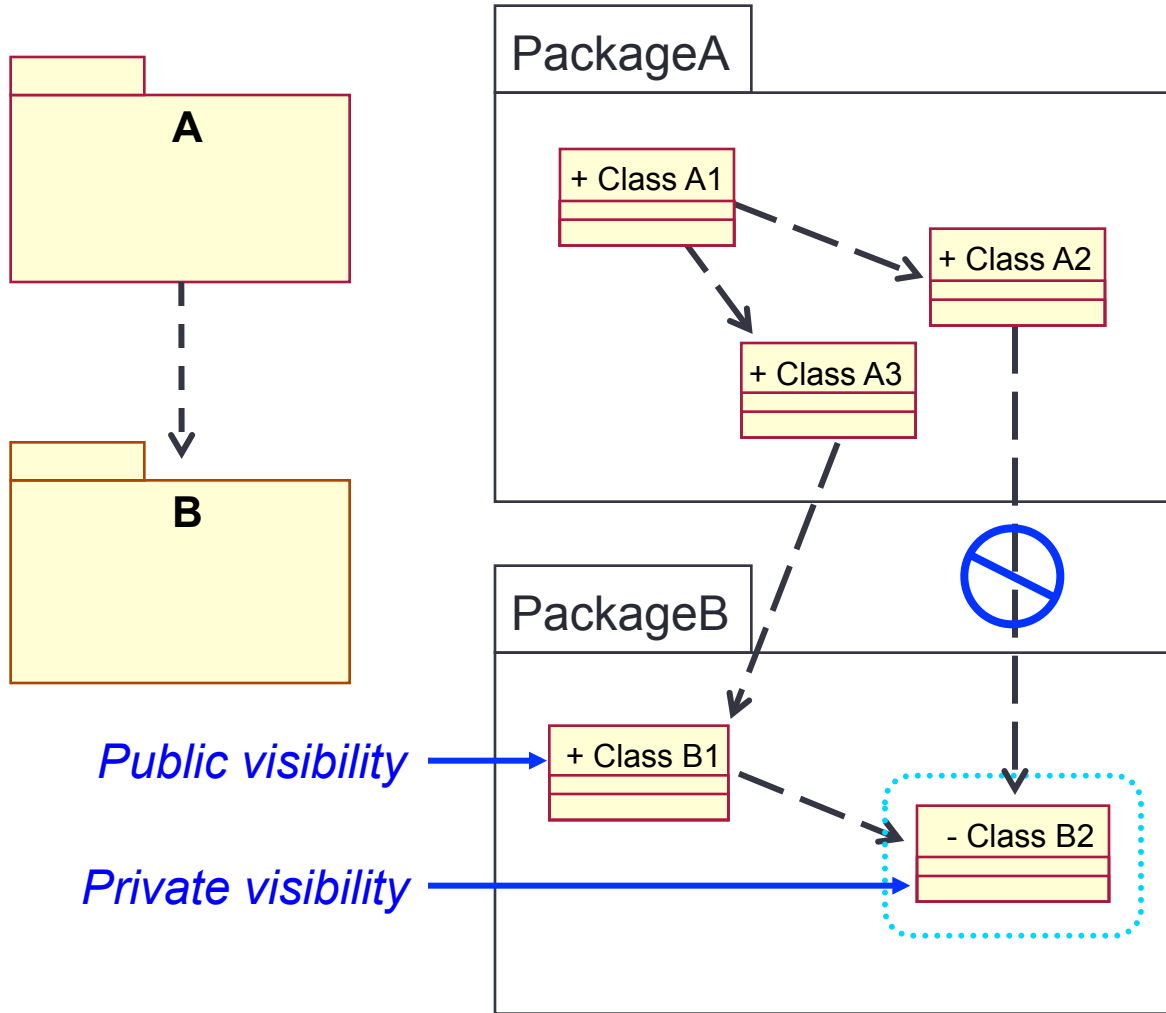
- Criteria for determining if classes are functionally related:
 - Changes in one class' behavior and/or structure necessitate changes in another class
 - Removal of one class impacts the other class
 - Two objects interact with a large number of messages or have a complex intercommunication
 - A boundary class can be functionally related to a particular entity class if the function of the boundary class is to present the entity class
 - Two classes interact with, or are affected by changes in the same actor

Packaging Tips:

Functionally Related Classes (continued)

- Criteria for determining if classes are functionally related (continued):
 - Two classes have relationships between each other
 - One class creates instances of another class
- Criteria for determining when two classes should **NOT** be placed in the same package:
 - Two classes that are related to different actors should not be placed in the same package
 - An optional and a mandatory class should not be placed in the same package

Package Dependencies: Package Element Visibility

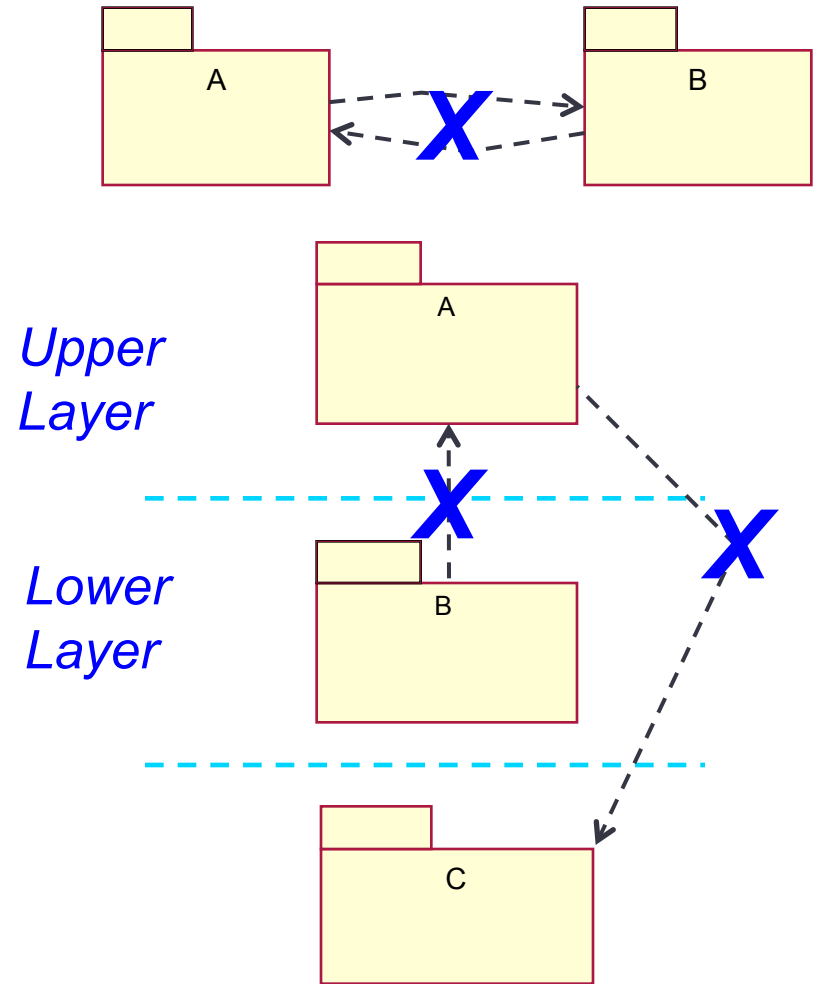


*Only public classes
can be referenced
outside of the
owning package*

OO Principle: Encapsulation

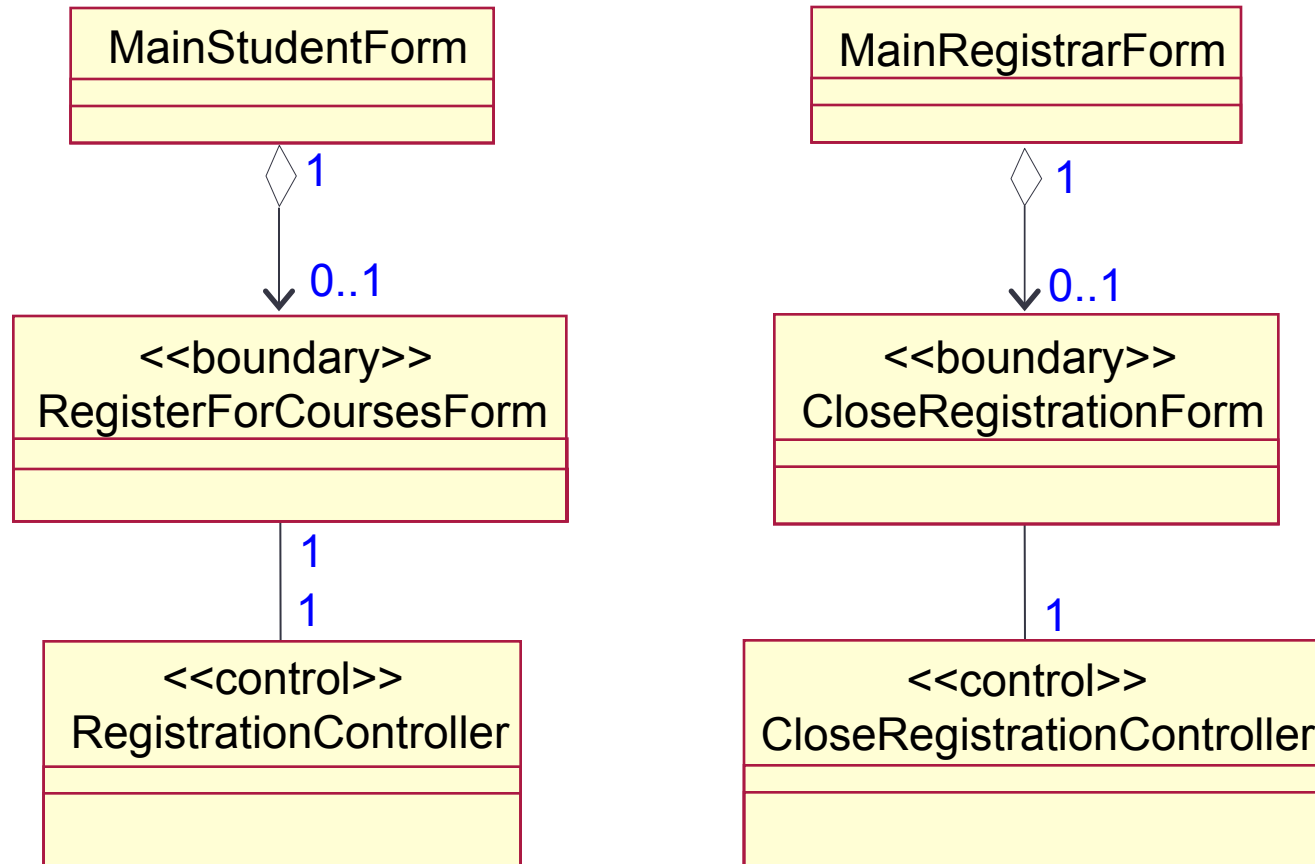
Package Coupling: Tips

- Packages should not be cross-coupled
- Packages in lower layers should not be dependent upon packages in upper layers
- In general, dependencies should not skip layers



X = Coupling violation

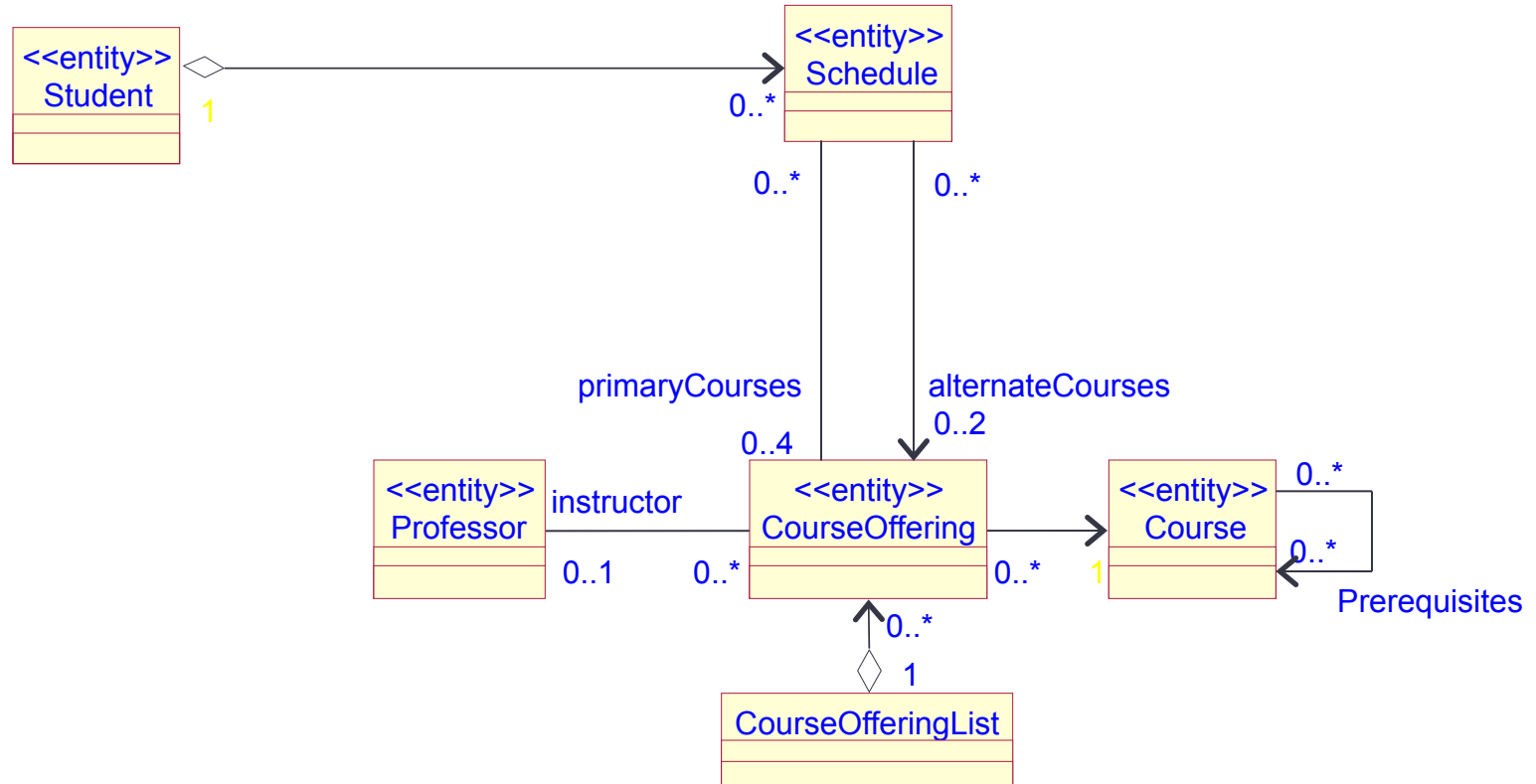
Example: Registration Package



Example: University Artifacts Package: Generalization



Example: University Artifacts Package: Associations



Example: External System Interfaces Package

