Software Engineering

Lesson #05 - Practice

Lesson #05 - Practice

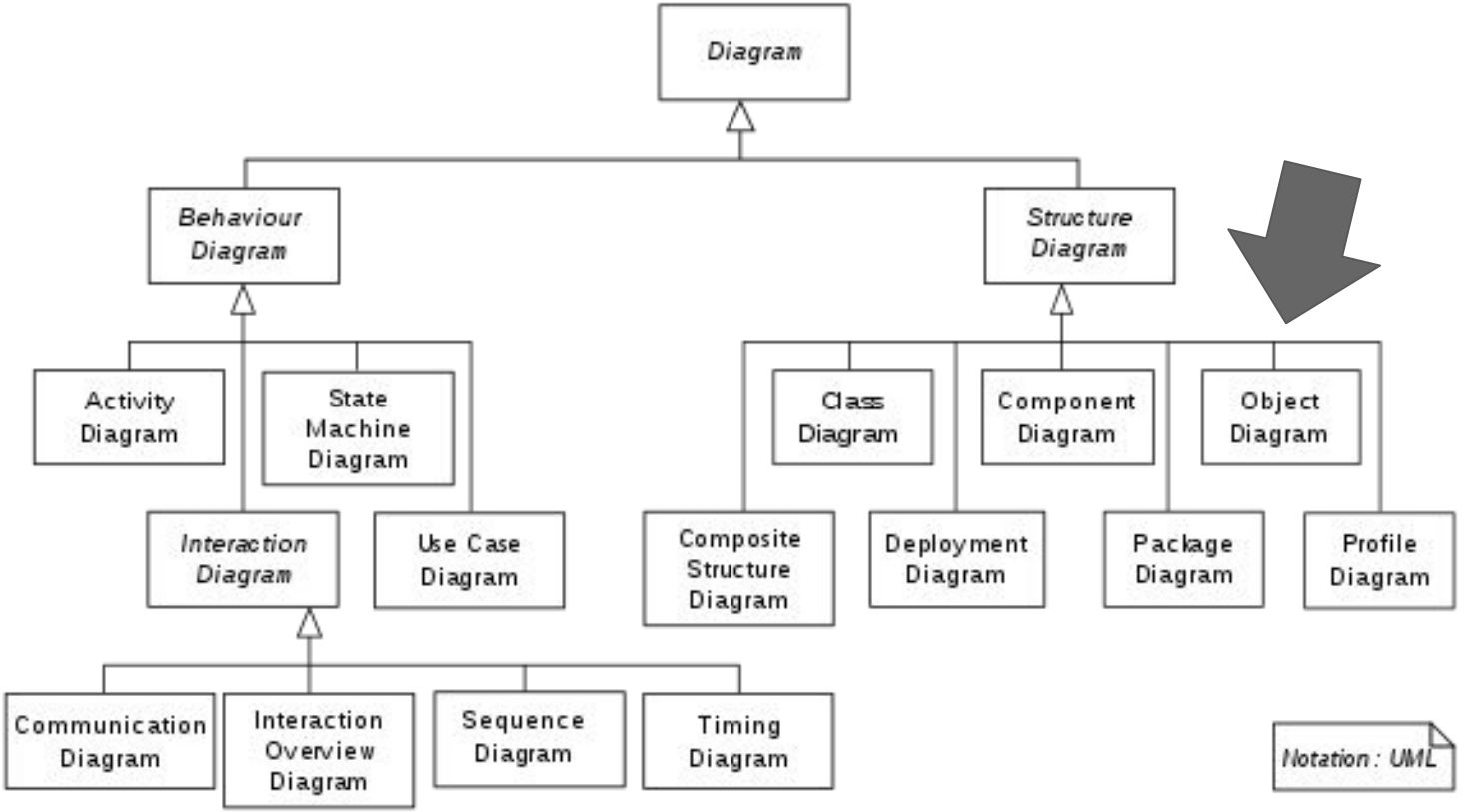
Agenda: Lesson #05 - Software Engineering - Practice

1	Object Diagrams
2	Package Diagrams
3	Class Work
4	Q & A

Agenda: Lesson #05 - Software Engineering - Practice

1. **Object Diagrams**
2. Package Diagrams
3. Class Work
4. Q & A

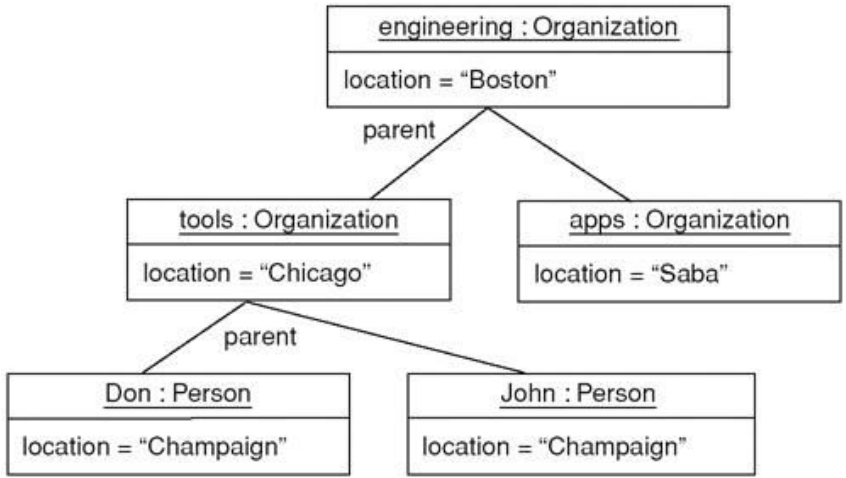
Object Diagrams



Object Diagrams

An Object diagram is a snapshot of the objects in a system at a point in time

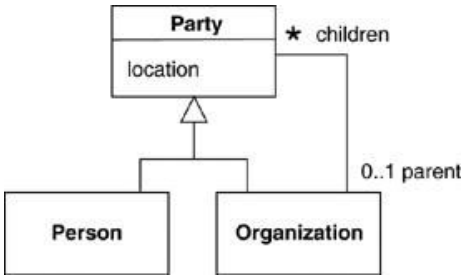
Because it shows instances rather than classes, an object diagram is often called an instance diagram

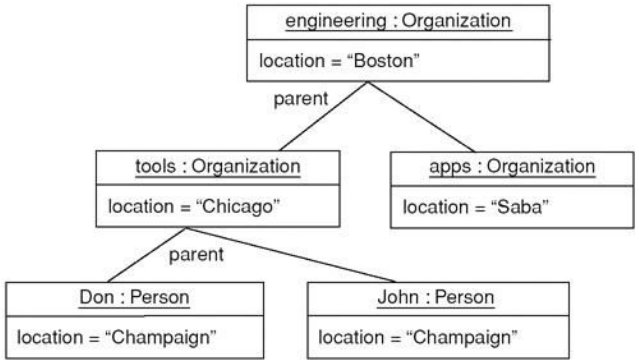


Object Diagrams

You can use an object diagram to show an example configuration of objects

This latter use is very useful when the possible connections between objects are complicated





Object Diagrams

You can tell that the elements are instances because the names are underlined

Each name takes the form instance name : class name

Both parts of the name are optional, so John, :Person, and aPerson are legal names

If you use only the class name, you must include the colon
You can show values for attributes and links

Object Diagrams

Strictly, the elements of an object diagram are instance specifications rather than true instances

The reason is that it's legal to leave mandatory attributes empty or to show instance specifications of abstract classes

You can think of an instance specification as a partly defined instance

When to Use Object Diagrams

Object diagrams are useful for showing examples of objects connected together

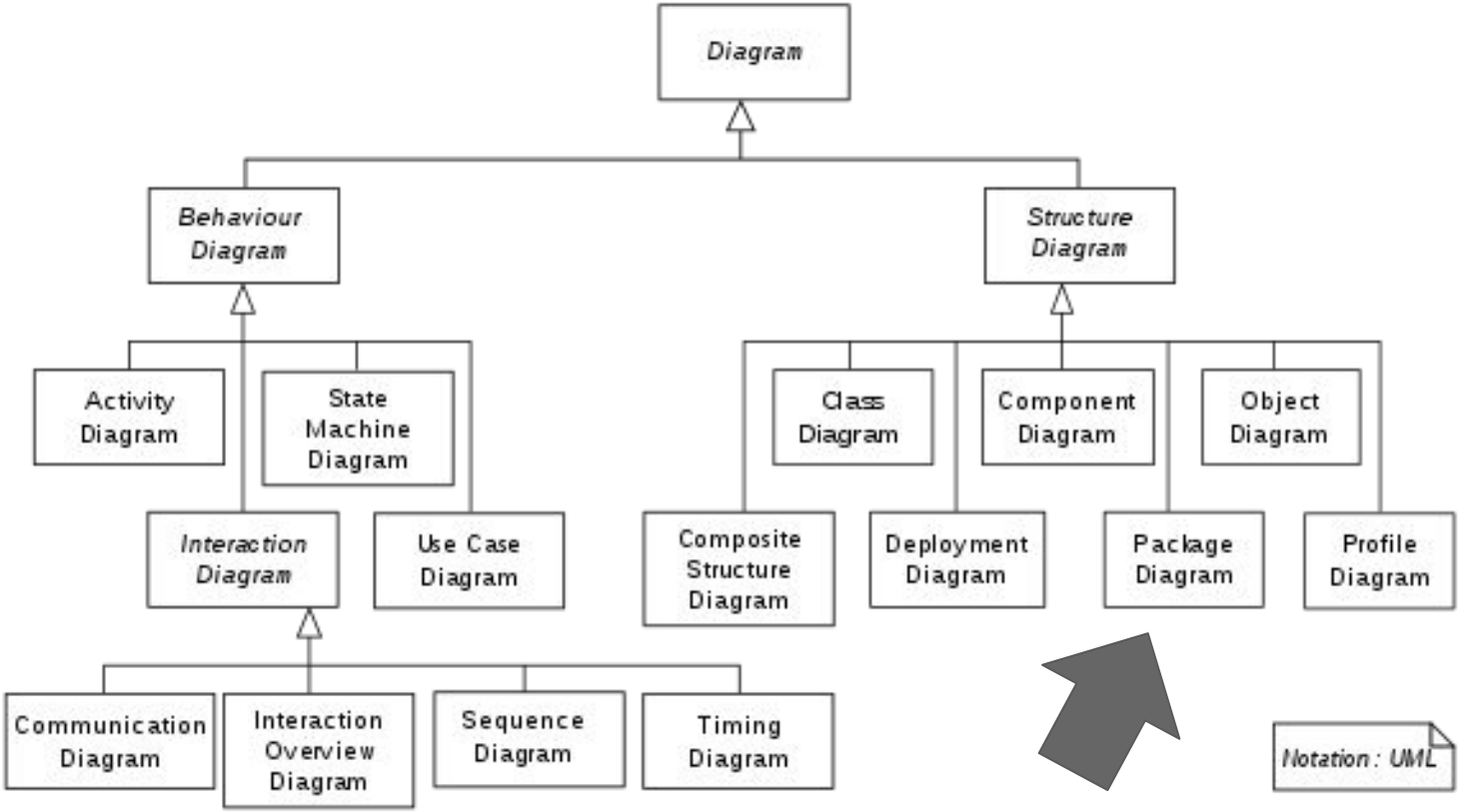
In many situations, you can define a structure precisely with a class diagram, but the structure is still difficult to understand

In these situations, a couple of object diagram examples can make all the difference

Agenda: Lesson #05 - Software Engineering - Practice

1. Object Diagrams
2. **Package Diagrams**
3. Class Work
4. Q & A

Package Diagrams

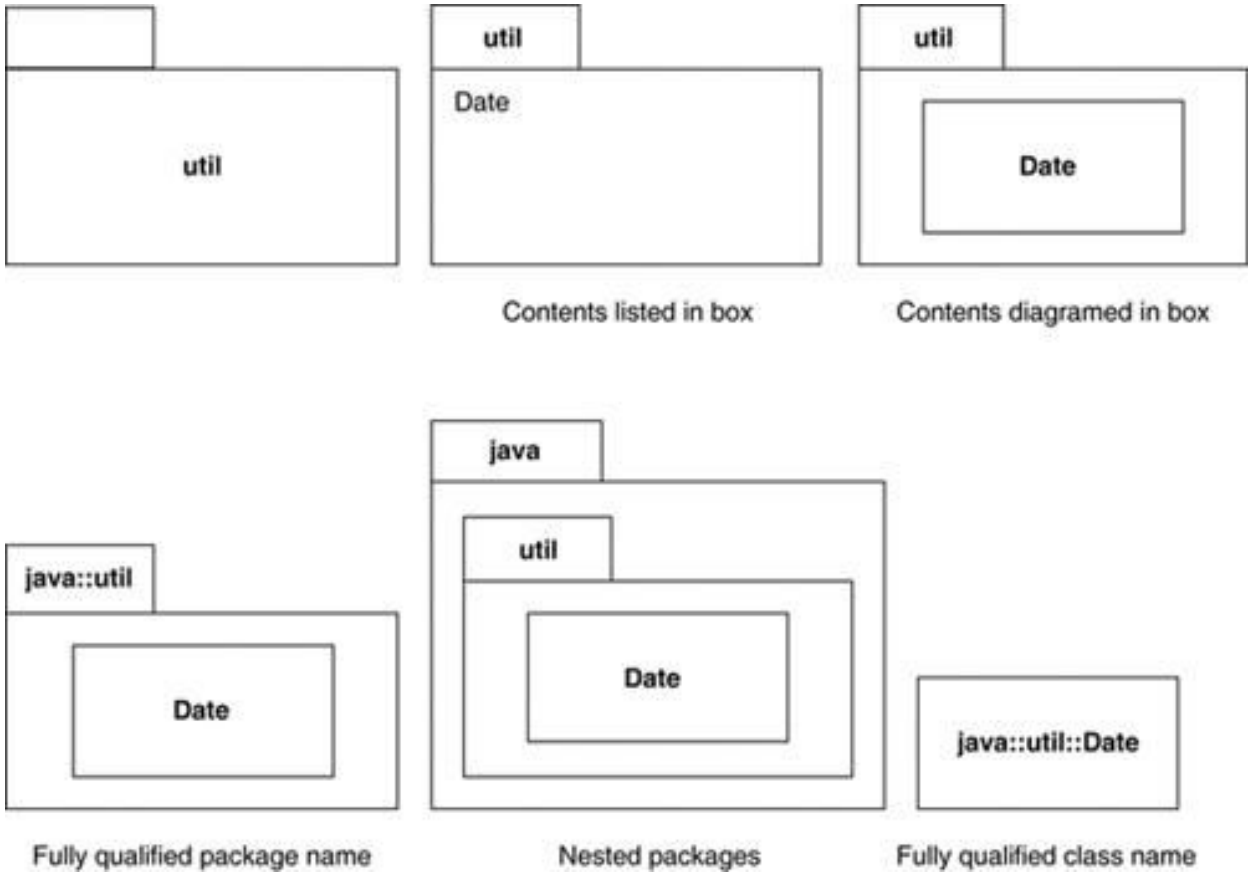


Package Diagrams

Classes represent the basic form of structuring an object-oriented system

Although they are wonderfully useful, you need something more to structure large systems, which may have hundreds of classes

Package Diagrams



Package Diagrams

A package is a grouping construct that allows you to take any construct in the UML and group its elements together into higher-level units

Its most common use is to group classes, and that's the way I'm describing it here, but remember that you can use packages for every other bit of the UML as well

Package Diagrams

In a UML model, each class is a member of a single package

Packages can also be members of other packages, so you are left with a hierarchic structure in which top-level packages get broken down into subpackages with their own subpackages and so on until the hierarchy bottoms out in classes

A package can contain both subpackages and classes

Package Diagrams

In programming terms, packages correspond to such grouping constructs as packages (in Java) and namespaces (in C++ and .NET)

Package Diagrams

Each package represents a namespace, which means that every class must have a unique name within its owning package

If I want to create a class called Date, and a Date class is already in the System package, I can have my Date class as long as I put it in a separate package

To make it clear which is which, I can use a fully qualified name, that is, a name that shows the owning package structure

Packages and Dependencies

A package diagram shows packages and their dependencies

If you have packages for presentation and domain, you have a dependency
from the presentation package to the domain package if any class in the presentation package has a dependency to any class in the domain package

In this way, inter-package dependencies summarize the dependencies between their contents

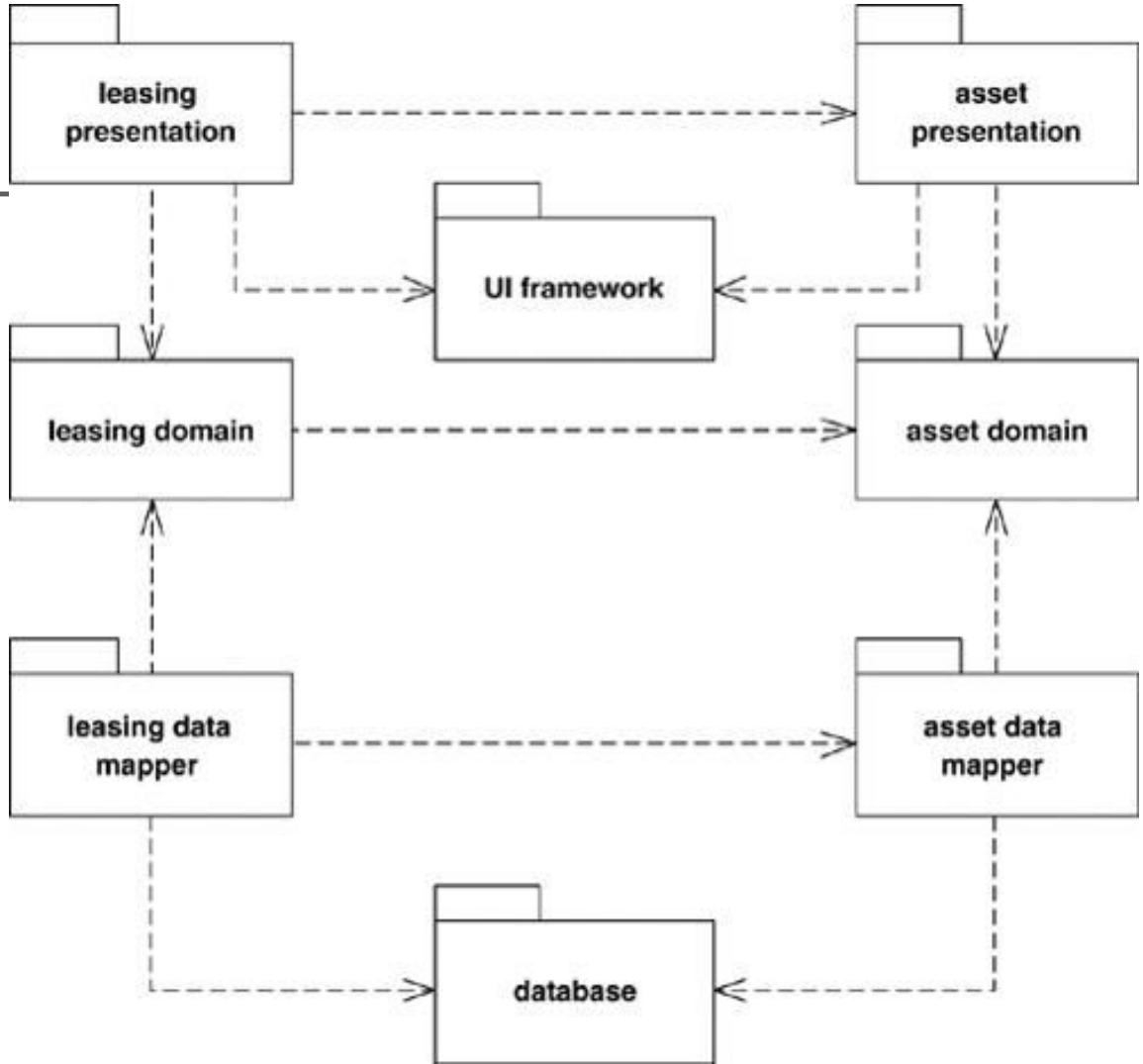
Packages and Dependencies

The UML has many varieties of dependency, each with particular semantics and stereotype

I find it easier to begin with the unstereotyped dependency and use the more particular dependencies only if I need to, which I hardly ever do

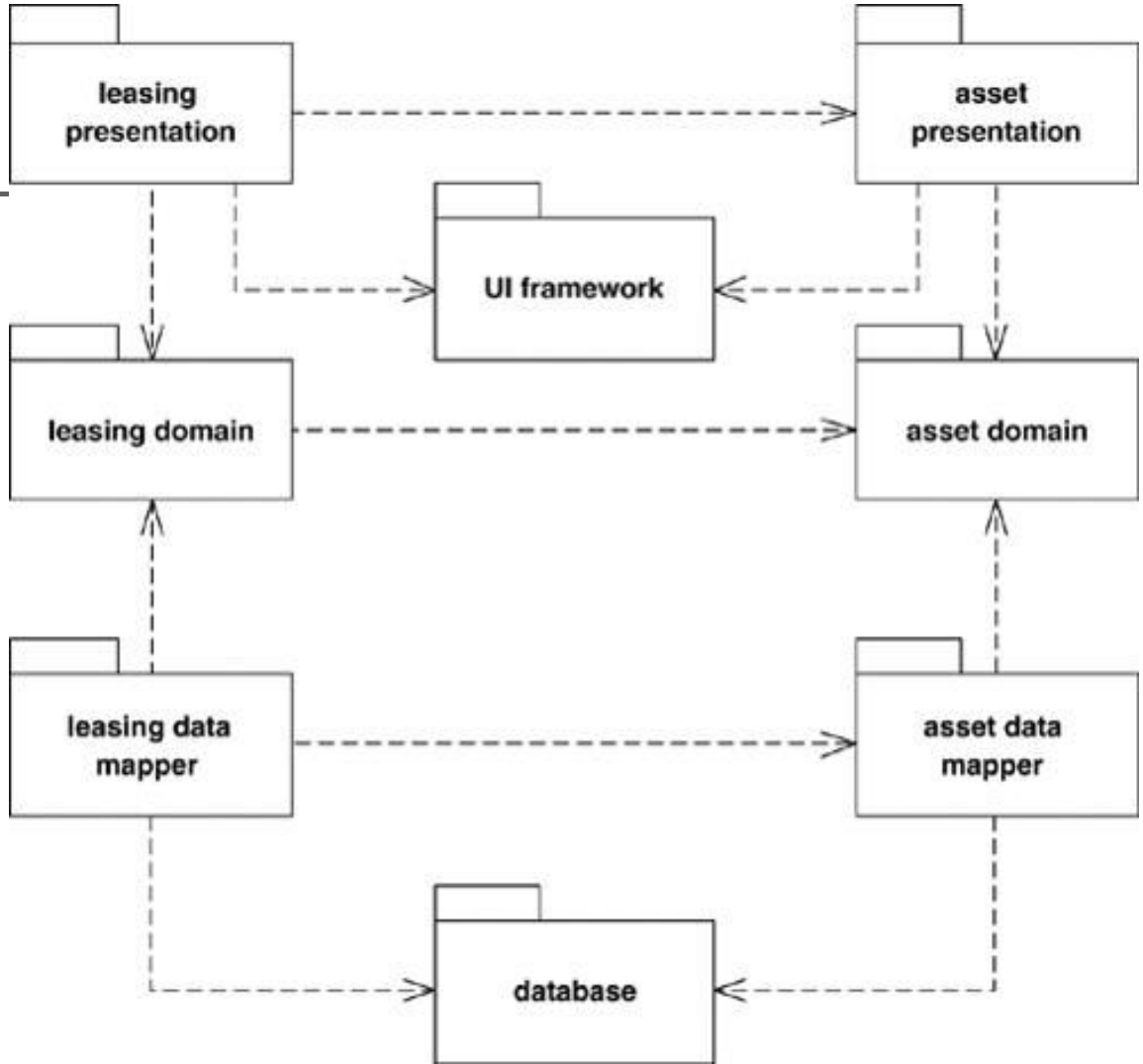
Packages and Dependencies

Often, you can identify a clear flow because all the dependencies run in a single direction



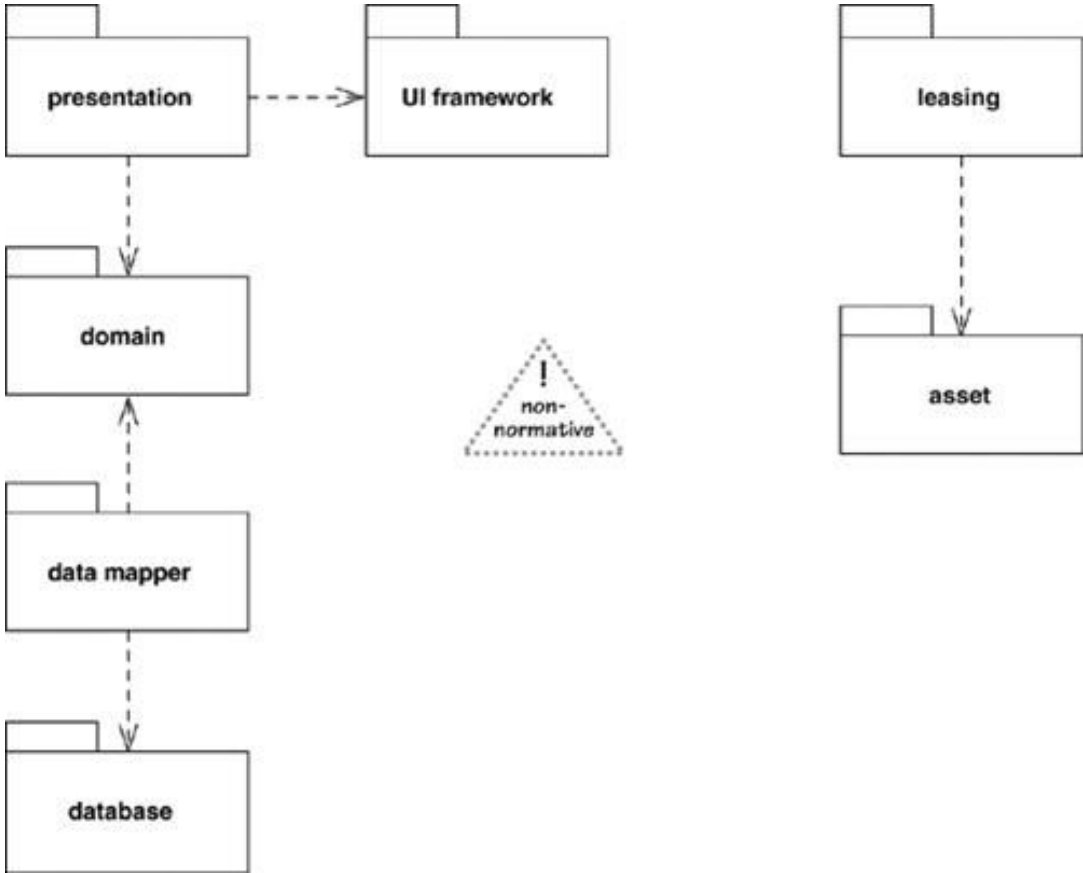
Packages and Dependencies

Although that is a good indicator of a well-structured system, the data mapper packages show an exception to that rule of thumb



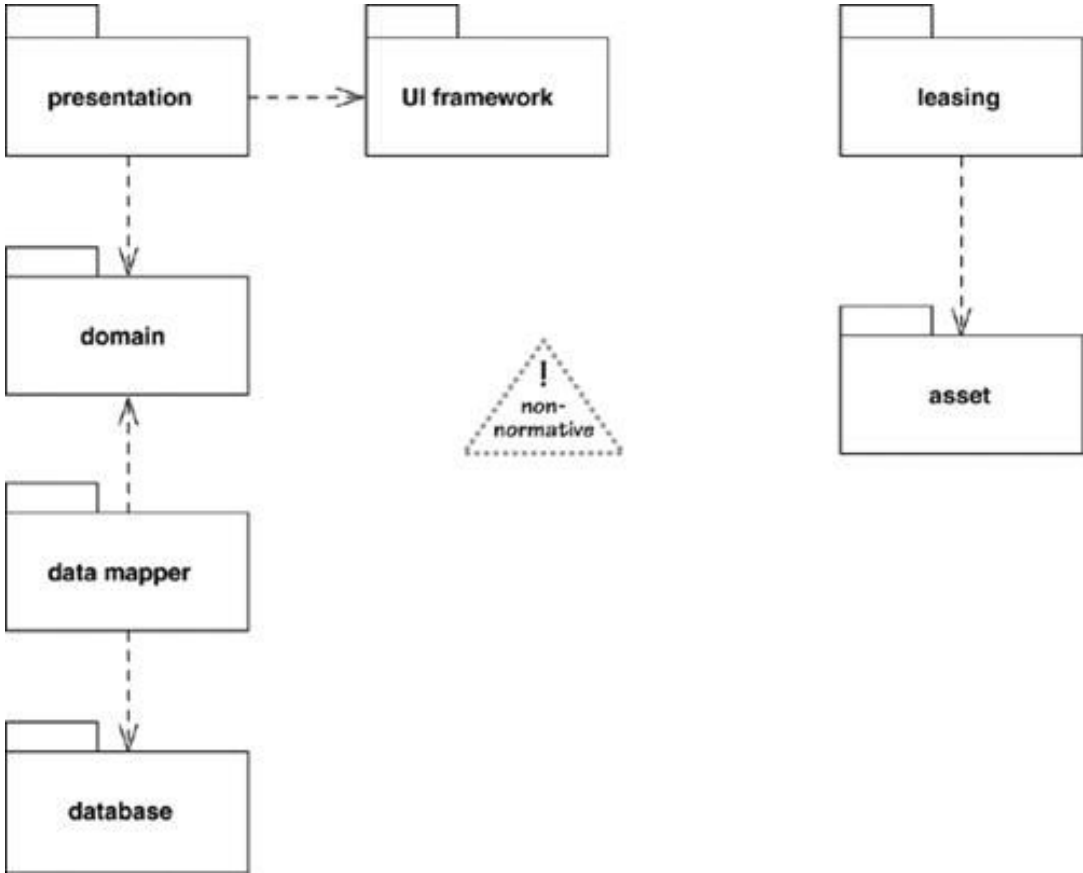
Package Aspects

One is a structure of layers in the application: presentation, domain, data mapper, and database



Package Aspects

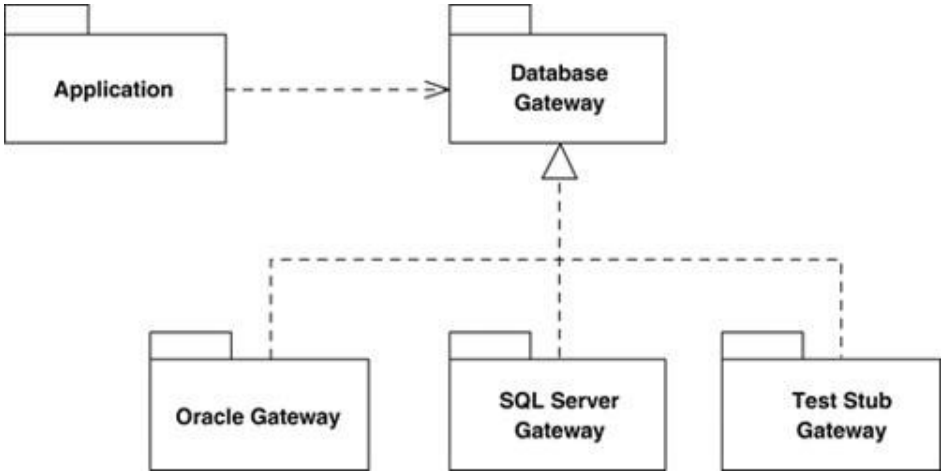
The other is a structure of subject areas: leasing and assets



Implementing Packages

Often, you'll see a case in which one package defines an interface that can be implemented by a number of other packages.

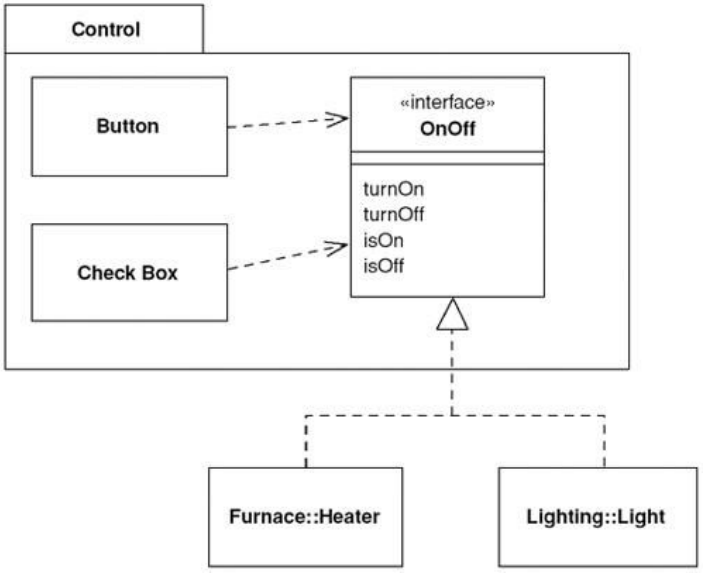
In this case, the realization relationship indicates that the database gateway defines an interface and that the other gateway classes provide an implementation



Implementing Packages

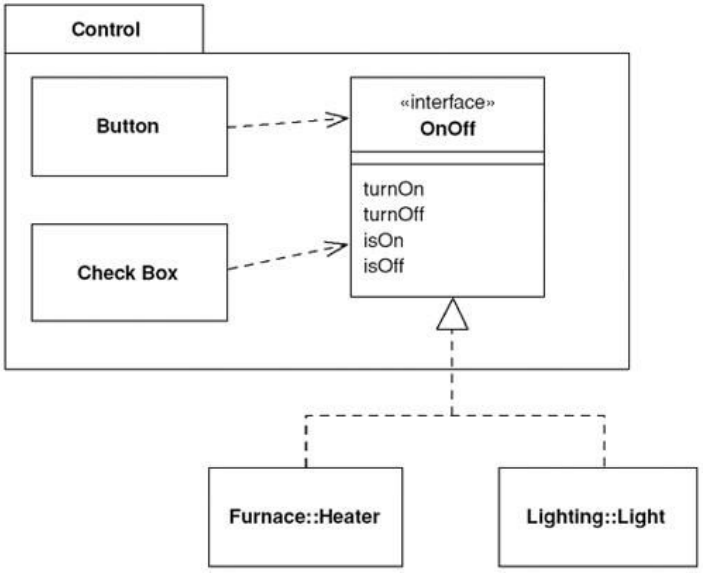
We want this to work with a lot of different things, such as heaters and lights

The UI controls need to invoke methods on the heater, but we don't want the controls to have a dependency to the heater



Implementing Packages

We can avoid this dependency by defining in the controls package an interface that is then implemented by any class that wants to work with these controls



When to Use Package Diagrams

I find package diagrams extremely useful on larger-scale systems to get a picture of the dependencies between major elements of a system

These diagrams correspond well to common programming structures

Plotting diagrams of packages and dependencies helps you keep an application's dependencies under control

When to Use Package Diagrams

Package diagrams represent a compile-time grouping mechanism

For showing how objects are composed at runtime, use a composite structure diagram

Agenda: Lesson #05 - Software Engineering - Practice

1. Object Diagrams
2. Package Diagrams
3. **Class Work**
4. Q & A

Software Engineering, 10. Global Edition

At the end of class, please, send your work to z.aldamuratov@kbtu.kz, indicating Software Engineering Practice #05 & your surname and name

Page: 223 Exercise: 7.3 Tool: StarUML

Class work

Software Engineering, 10. Global Edition

Page: 223 Exercise: 7.3 Tool: StarUML

Using the UML graphical notation for object classes, design the following object classes, identifying attributes and operations. Use your own experience to decide on the attributes and operations that should be associated with these objects.

a messaging system on a mobile (cell) phone or tablet
a printer for a personal computer
a personal music system
a bank account a library catalogue

Agenda: Lesson #05 - Software Engineering - Practice

1. Object Diagrams
2. Package Diagrams
3. Class Work
4. **Q & A**

Agenda: Lesson #05 - Software Engineering - Practice

Q & A