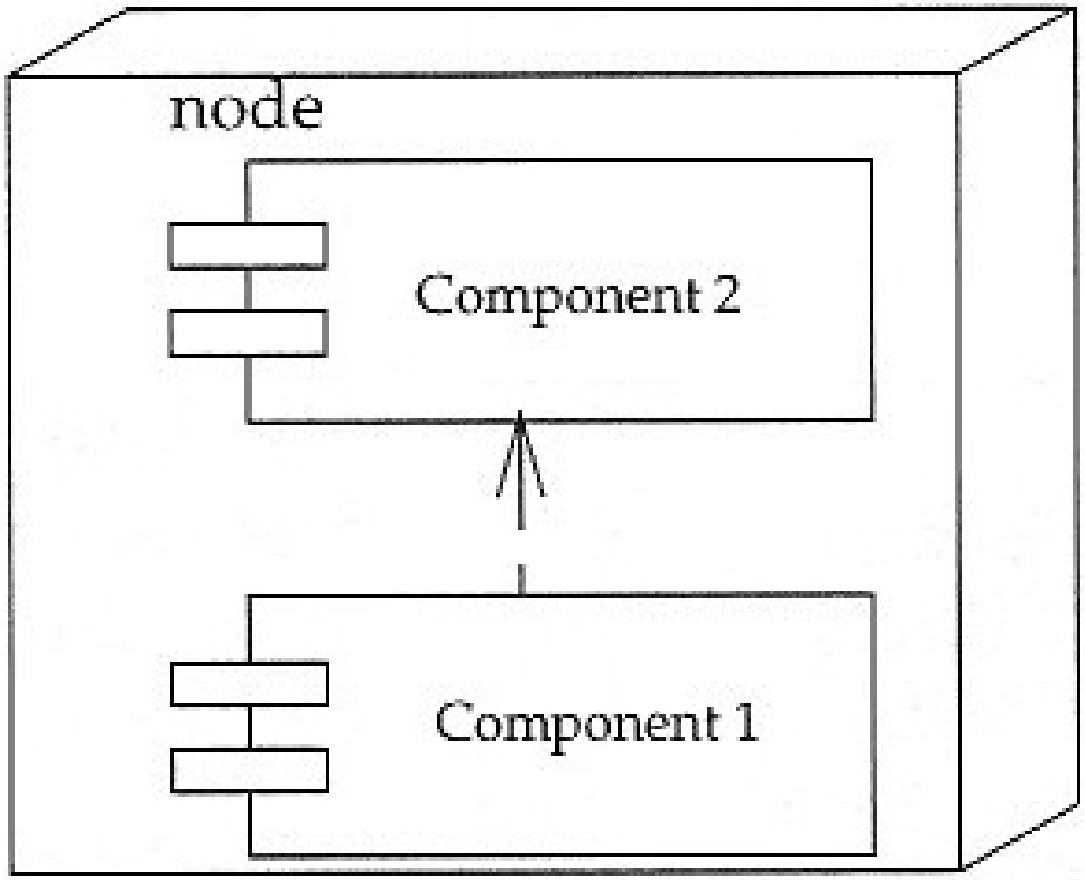
What is a Deployment Diagram?

- Deployment Diagram – a diagram that shows the physical relationships among software and hardware components in a system
- Shows the configuration of:
  - run time processing nodes and
  - the components that live on them

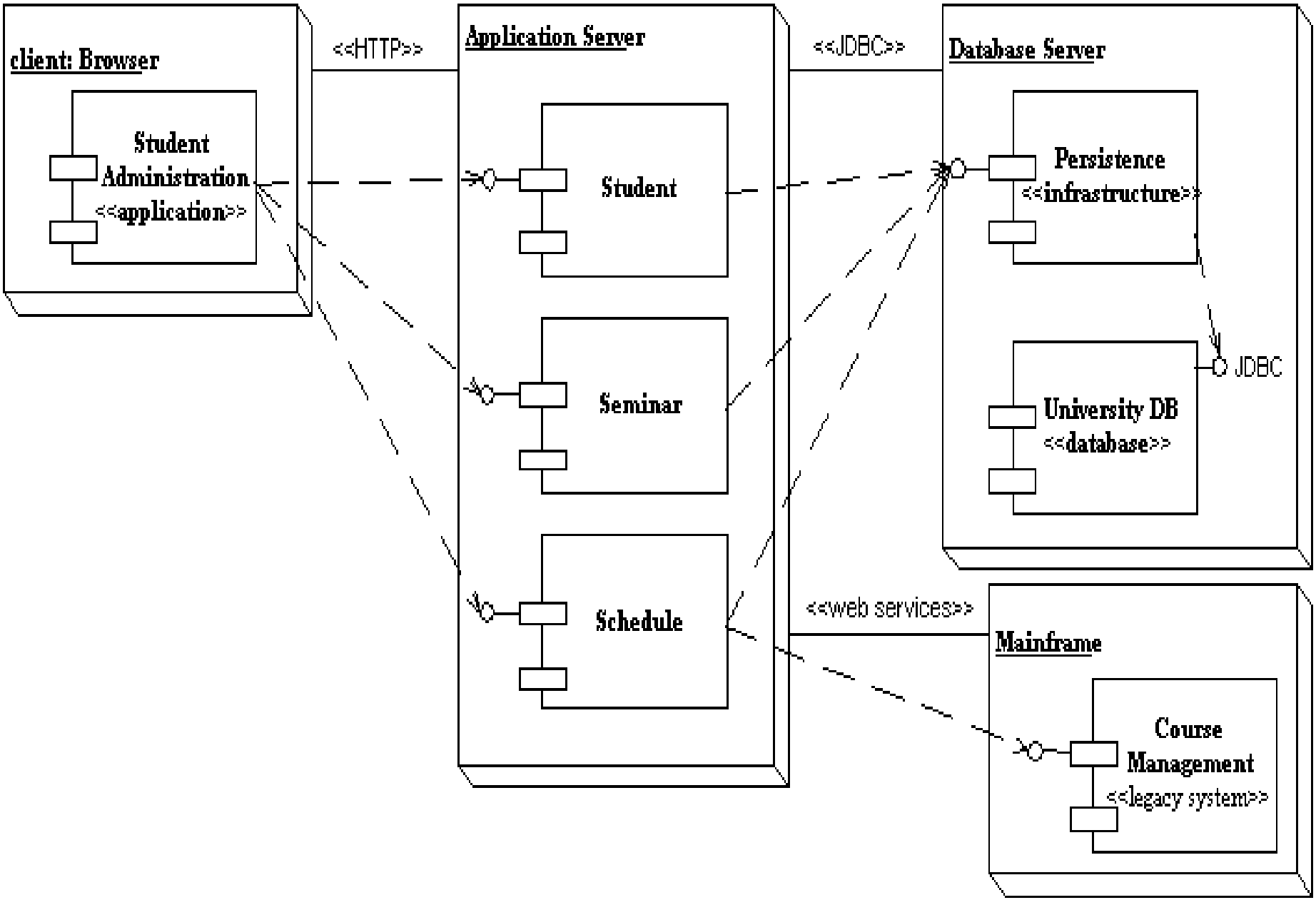
  Components – physical modules of code
  - Connections – show communication paths
  - Dependencies – show how components communicate with other components
  - Nodes – computational units, usually a pieces of hardware

Deployment Diagram



A node, depicted as a three-dimensional box, represents a computational unit, typically a single piece of hardware, such as a computer, network router, mainframe, sensor, or personal digital assistant (PDA).

Components, depicted as rectangles with two smaller rectangles jutting out from the left-hand side (the same notation used on UML Component diagrams), represent software artifacts such as file, framework, or domain component.


client: Browser
Student Administration
<<application>>
<<HTTP>>
Application Server
Student
Seminar
Schedule
<<JDBC>>
Database Server
Persistence
<<infrastructure>>
JDBC
University DB
<<database>>
<<web services>>
Mainframe
Course Management
<<legacy system>>

- Communication associations support one or more communication protocols, each of which should be indicated by a UML stereotype. In Figure 1 you see that the HTTP, JDBC, and web services protocols are indicated using this approach. Table 1 provides a representative list of stereotypes for communication associations – your organization will want to develop its own specific standards.
- **Table 1. Common stereotypes for communication associations.**
- **Stereotype Implication**
- Asynchronous An asynchronous connection, perhaps via a message bus or message queue.
- HTTP HyperText Transport Protocol, an Internet protocol.
- JDBC Java Database Connectivity, a Java

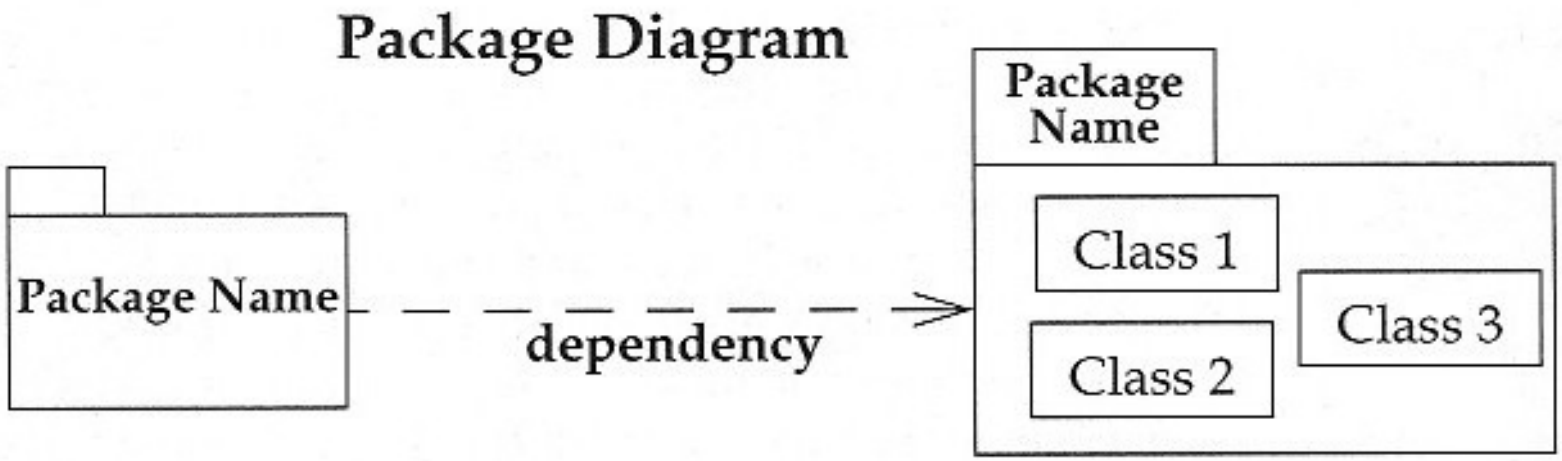
What is a Package?

- Package – a grouping of classes (a conventional Package - a unit above a class in the abstraction hierarchy) and other packages (a Domain Package)
- Package Diagram – a UML diagram that shows packages of classes and the dependencies among them
  - A dependency exists between two elements if changes to the definition of one element may cause changes to the other
- Classes have dependencies for several reasons, including:
  - One class sends a message to another
  - One class has another as part of its data
  - One class mentions another as a parameter to an operation

- **Packages**
  - groups of “basic elements”, e.g., classes or use cases
  - namespaces, i.e., all members should have unique names
  - represented as file folders
  - can contain other packages, creating hierarchy

- **Relations**
  - dependencies, implementations, …
  - *imports* and *merges*

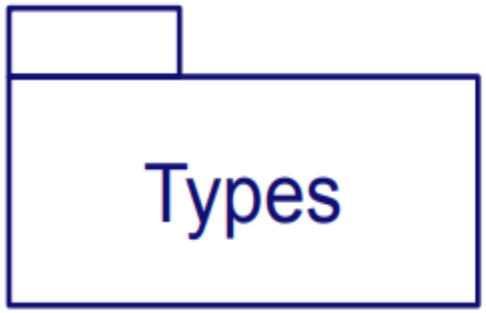
Package Diagrams - Notation



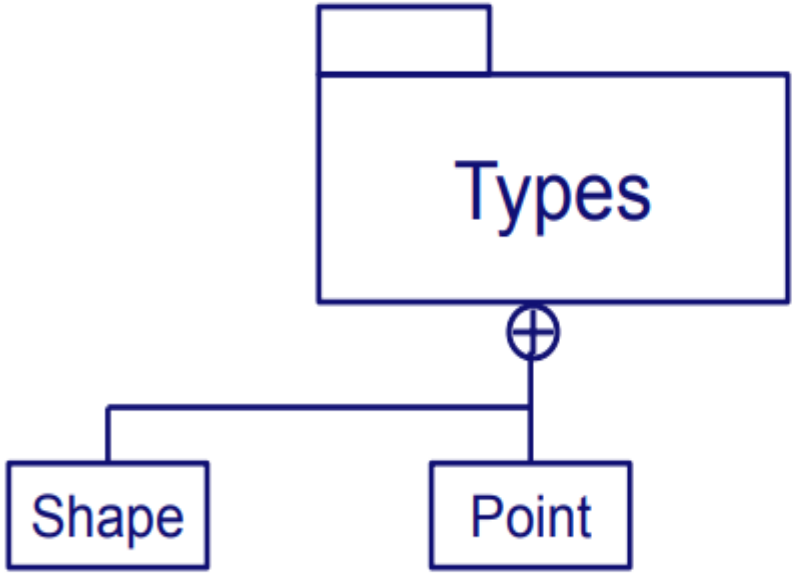
- Package – contains classes
- Dependency – changes to the definition (interface) of one package may cause changes in the other package

Reference: *UML Distilled*, Inside Cover

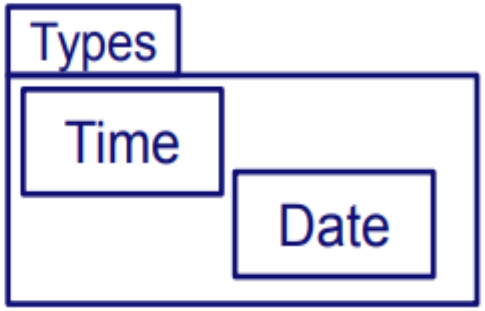
Package representations



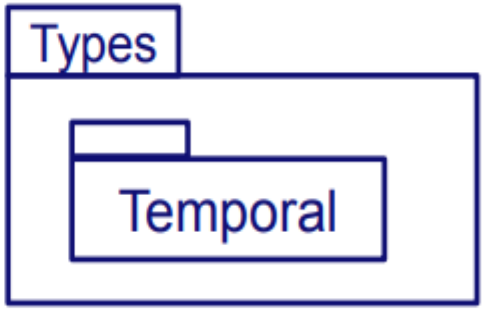
Package Types,
members not shown



Package Types, ***some*** members
shown using ⊕-notation



Package Types, ***some*** members
within the borders of the package



Nested packages

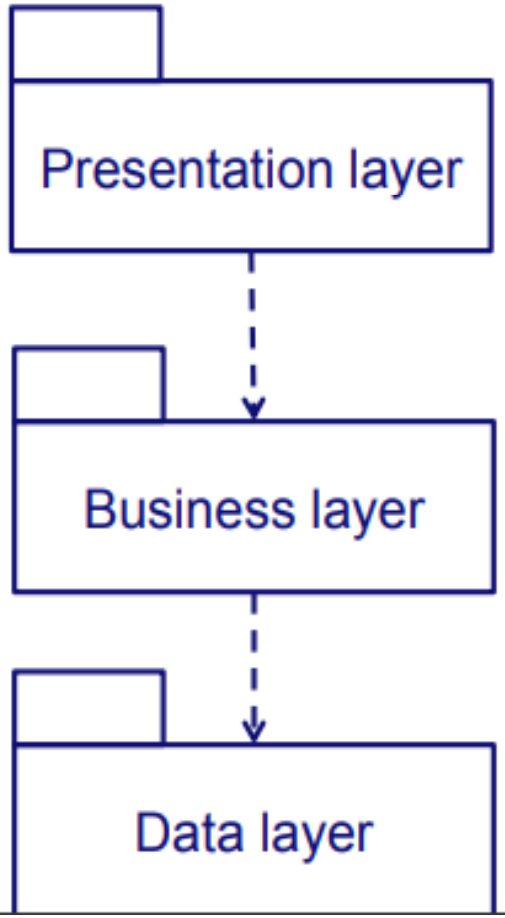
Relations: Dependencies

- Package A **depends** on package B if A contains a class which depends on a class in B
  - Summarise dependencies between classes

- Graphic representation:

Relations: Dependencies

- Package A **depends** on package B if A contains a class which depends on a class in B
  - Summarise dependencies between classes
- Typical 3-tier application (*sketch*):



UI, web-interface, services to other systems

Core calculations, operations, etc

Data storage (DB)