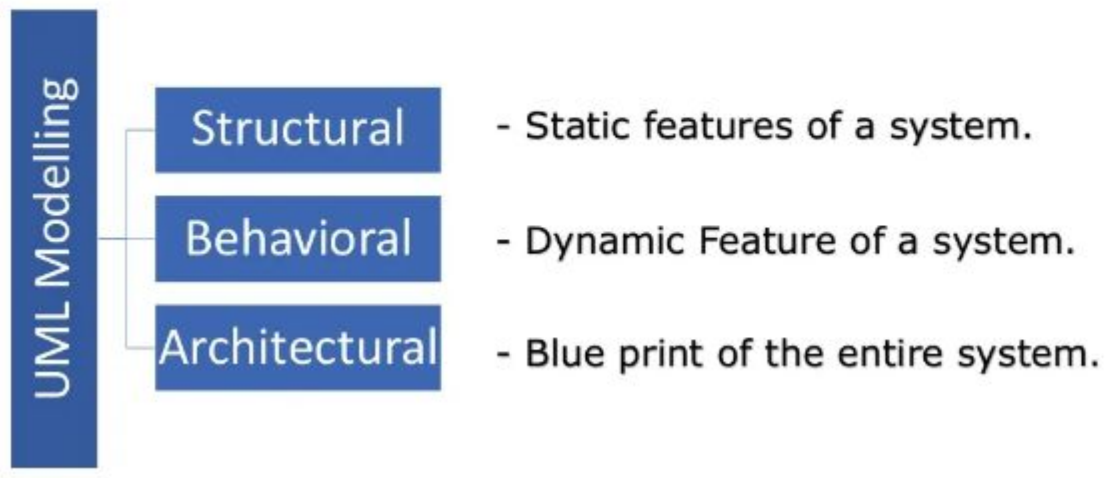
UML: Unified Modeling Language

UML: Unified Modeling Language

- UML stands for Unified Modeling Language
- It is a Visual Language
- It is Industry Standard Graphical Language for specifying,visualizing, constructing and documenting the artifact of the system.
- UML mostly uses graphical notations to express Object Oriented Analysis and Design of the software
- It simplifies the complex process of the software design

UML: Unified Modeling Language


UML Modelling
Structural
- Static features of a system.
Behavioral
- Dynamic Feature of a system.
Architectural
- Blue print of the entire system.

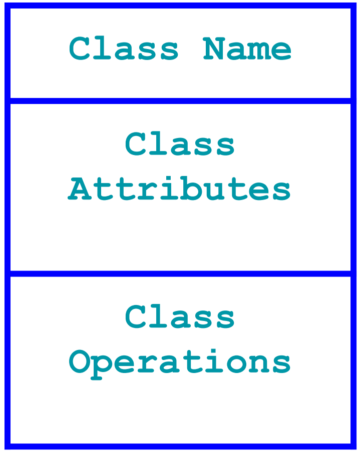
Structural Diagram

- Structural Diagram
  - Class Diagram
  - Object Diagram
  - Component Diagram
  - Deployment Diagram

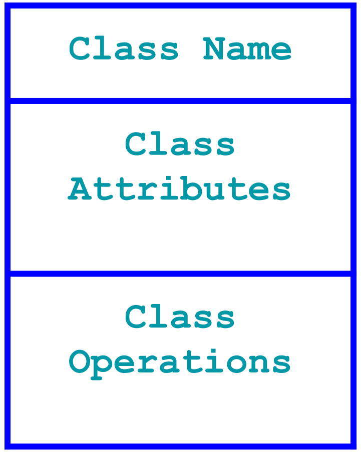
Class Diagram

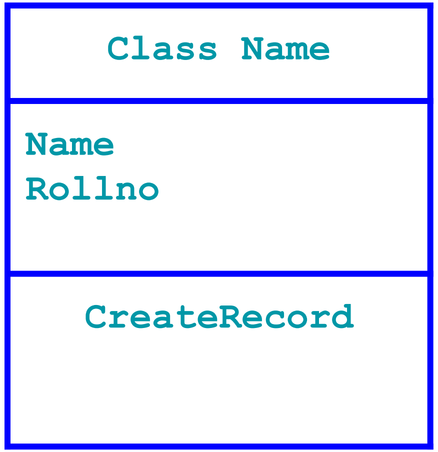
- It shows the classes of the system, their inter-relationships, the operations and the attributes of the classes.
- Explore domain concepts in the form of domain model.
- Analyze requirements in the form of conceptual/analysis model.
- Depict the detailed design of object oriented or object based software.

Class Diagram

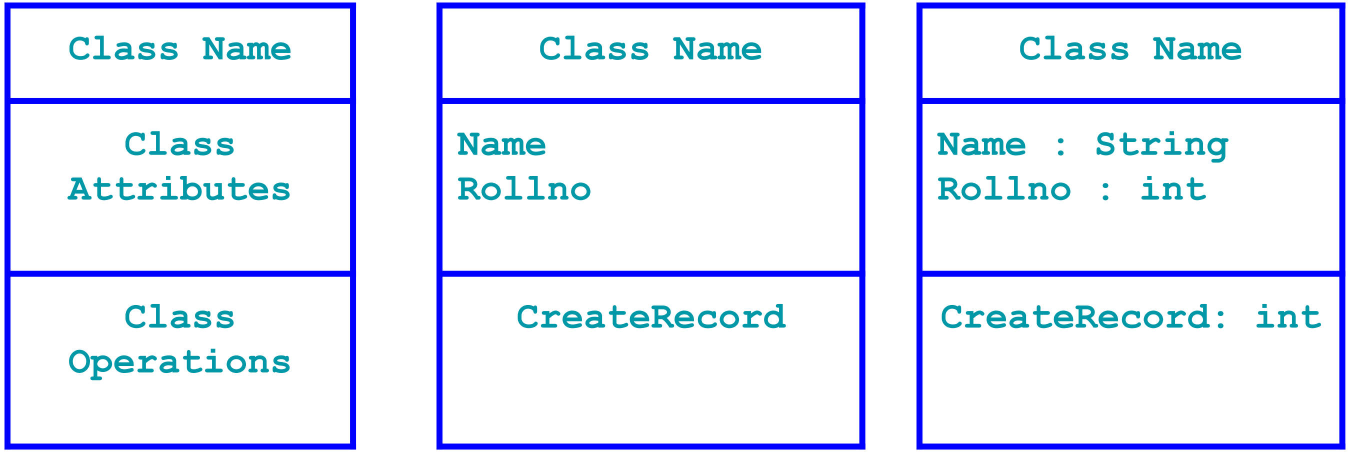


Class Diagram

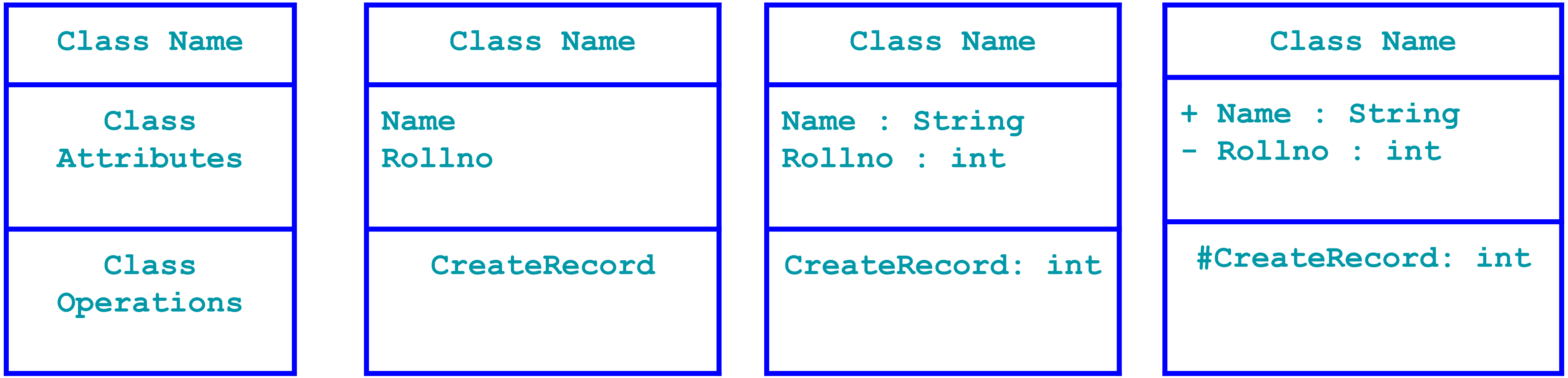




Class Diagram



Class Diagram



Visibility Notations:

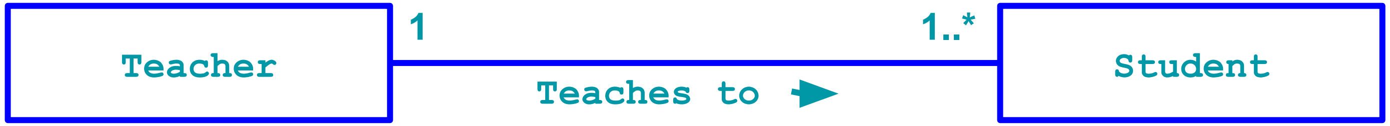
+	Public
-	Private
#	Protected

Class Diagram

- It has relationships between the classes
  - Association
  - Dependency
  - Aggregation
  - Composition
  - Generalization

Class Diagram

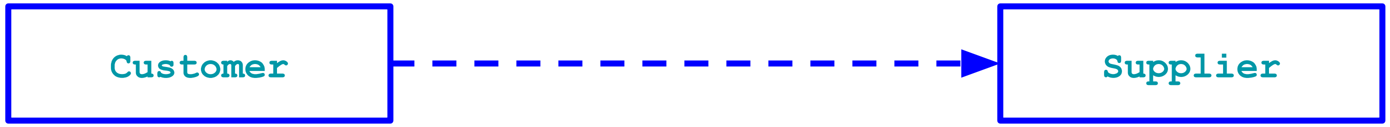
- **Association**



1
0..1
0..*
1..*
Or Exact numbers 3,4,5,..

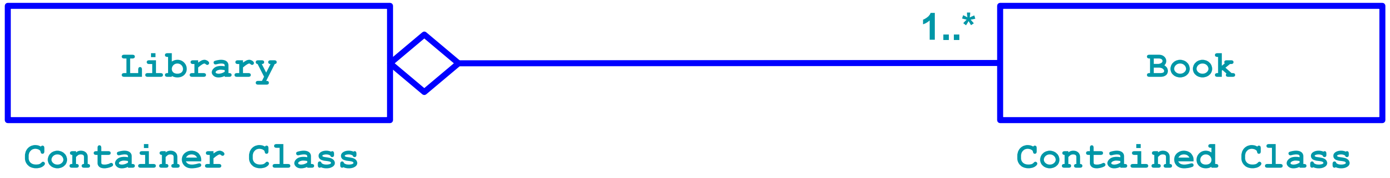
Class Diagram

- **Dependency**



Class Diagram

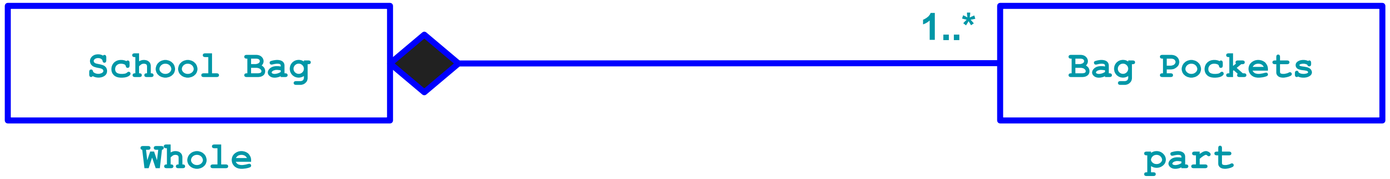
- **Aggregation**



Here, book class is not strongly depend on library class i.e. contained class is not strongly dependent on container class.

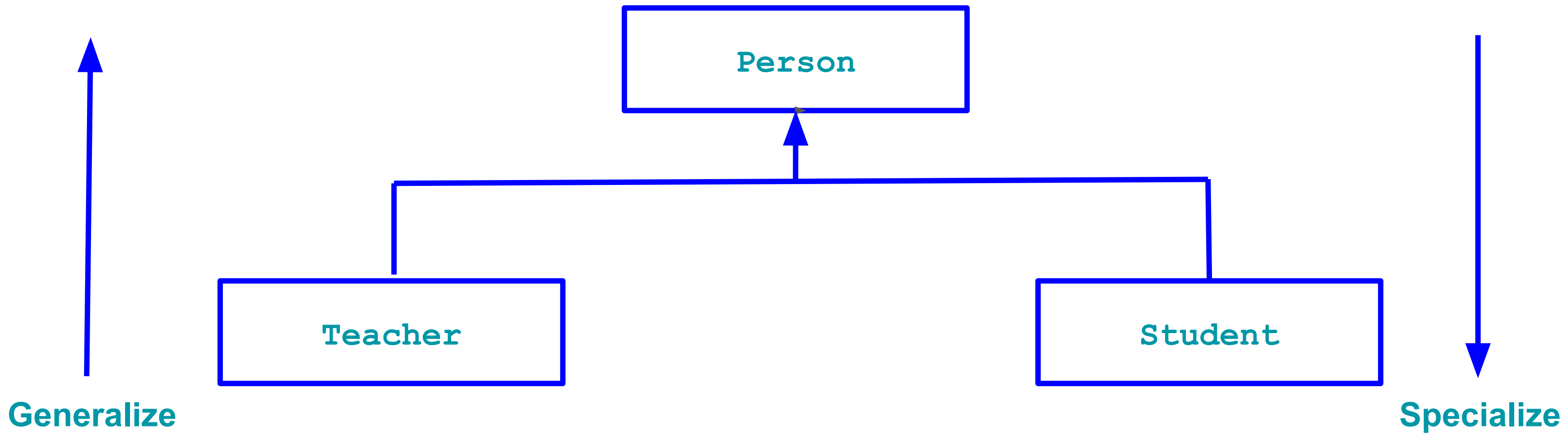
Class Diagram

- **Composition**
  - It is like aggregation.
  - Example: A school bag and its pockets.
  - If school bag is destroyed, then pockets automatically destroyed.
  - I.e. Contained class has strongly dependency on contained class.



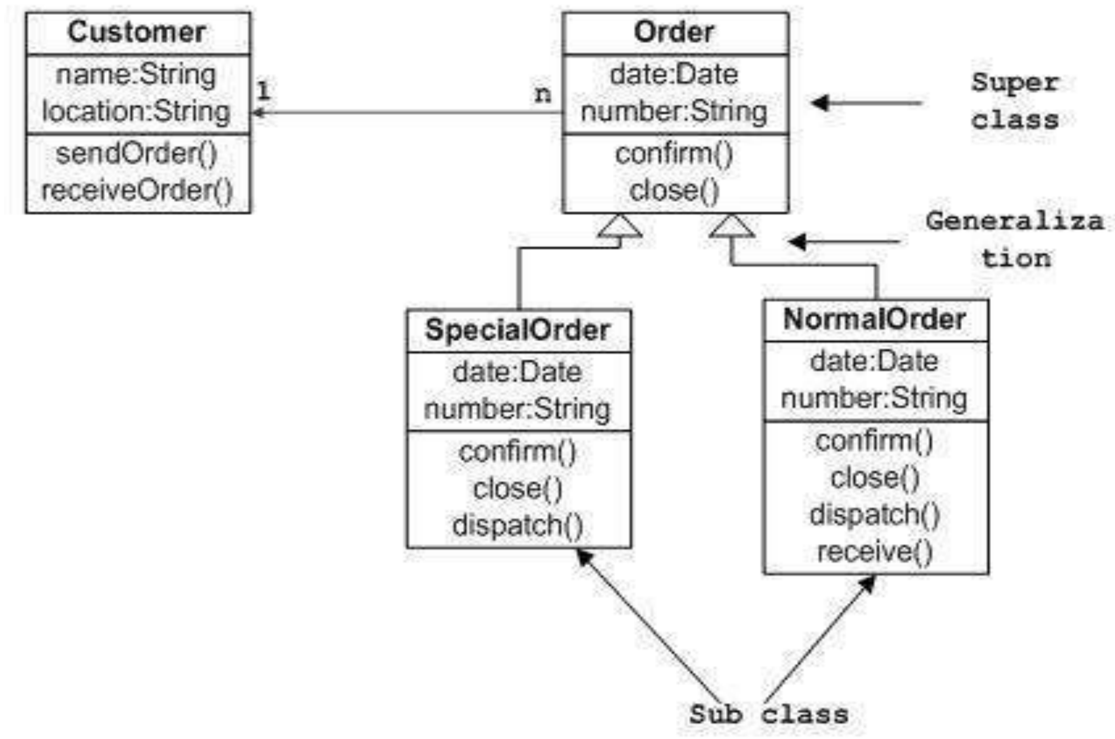
Class Diagram

- **Generalization**



Class Diagram

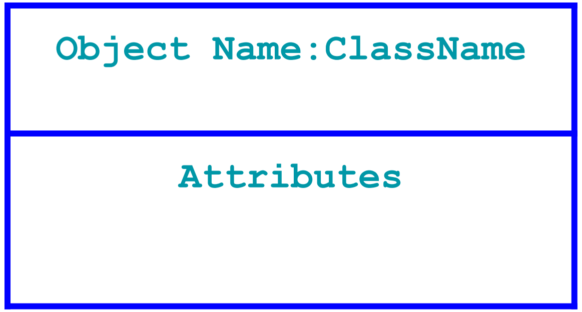
Sample Class Diagram



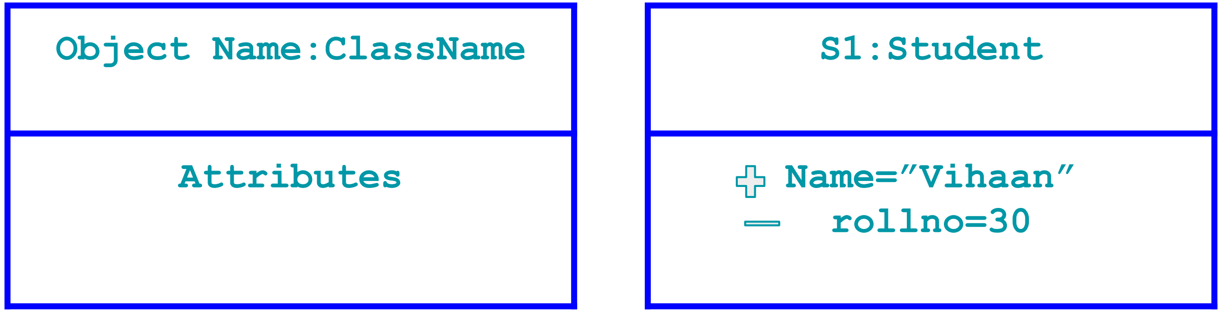
Object Diagram

- Object Diagrams or Instance Diagrams are useful for exploring real world examples of objects and the relationship between them.
- It shows instances instead of classes.
- They are useful for exploring small pieces with complicated relationships, especially recursive relationships.

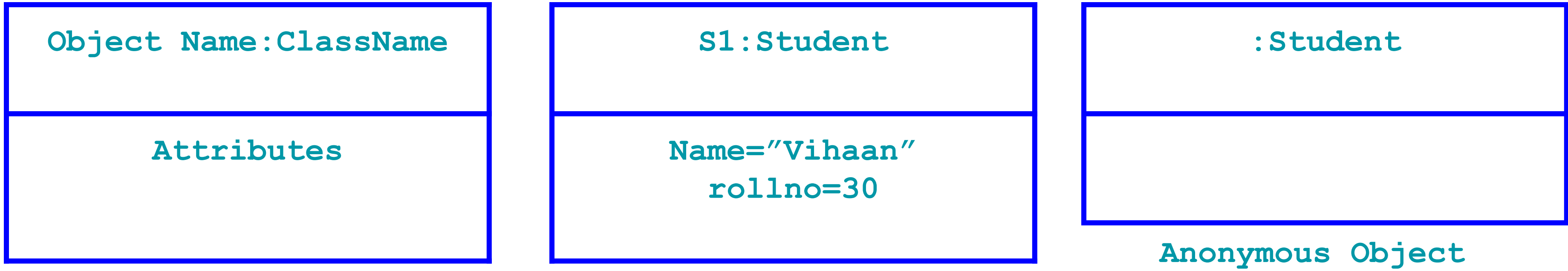
Object Diagram



Object Diagram

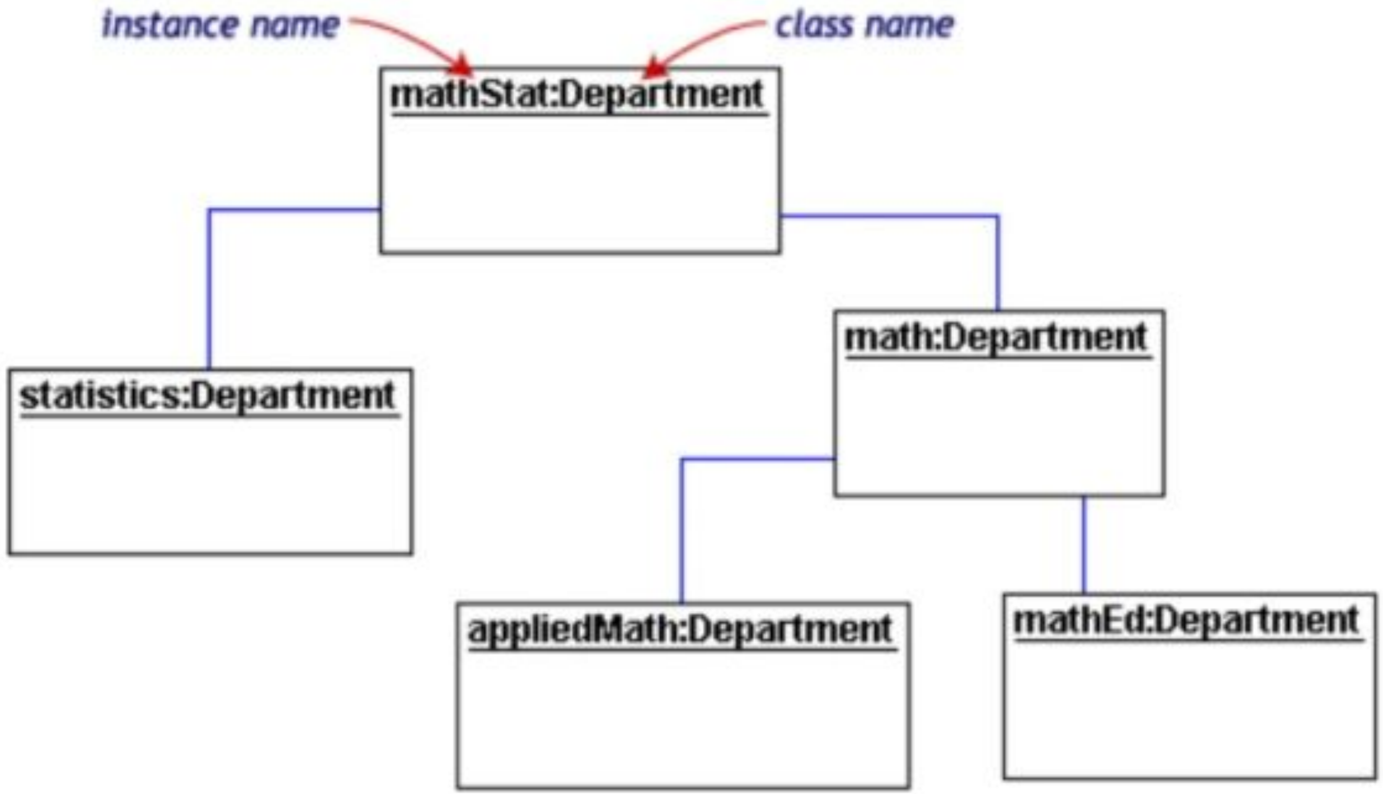


Object Diagram



Anonymous Object

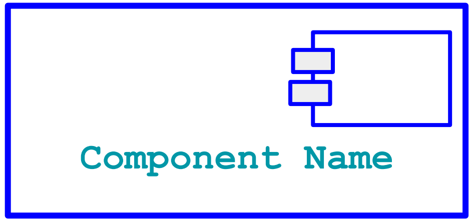
Object Diagram



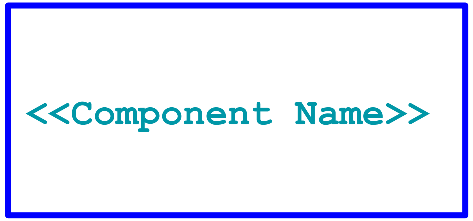
Component Diagrams

- To represent physical aspect of our system, we use this diagram.
- It shows dependency among software components including the software classifiers that specify them (e.g. Implementation Classes)and the artifacts that implement them; such as source code files, binary code files, executable files, script and tables

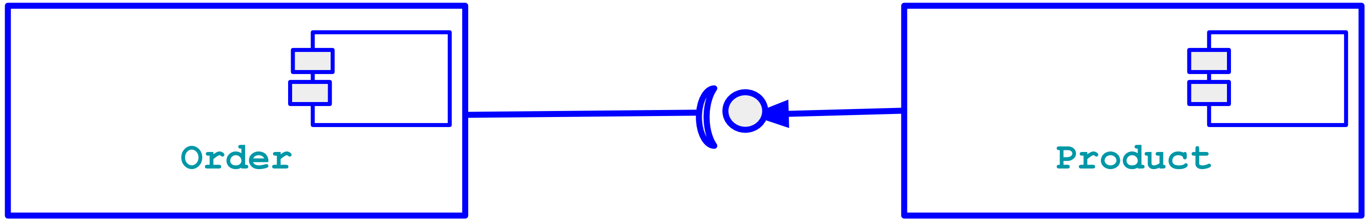
Component Diagrams



One way of Representation

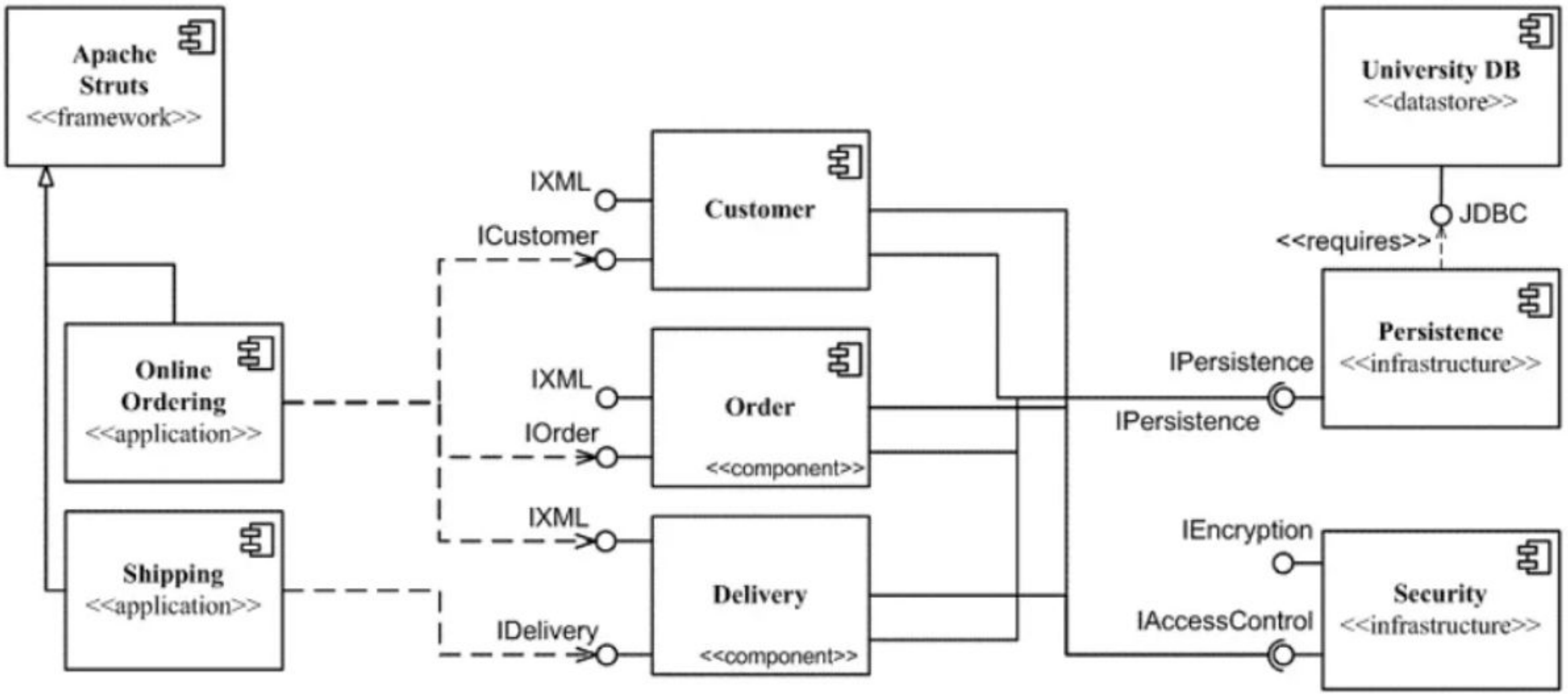


Second way of Representation



One way of Representation

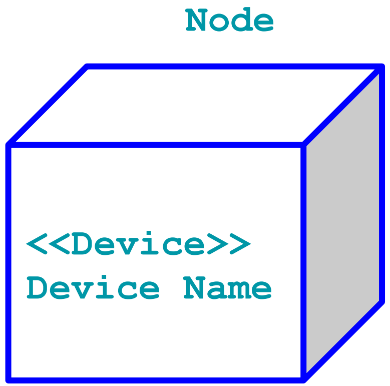
Component Diagrams

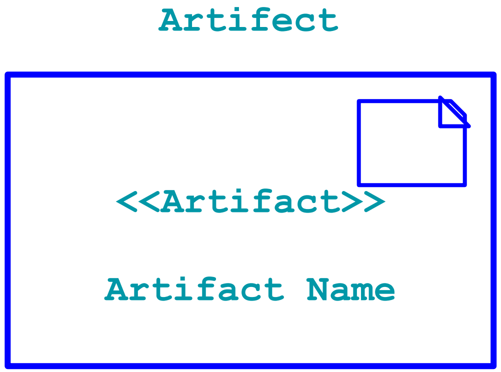


Deployment Diagrams

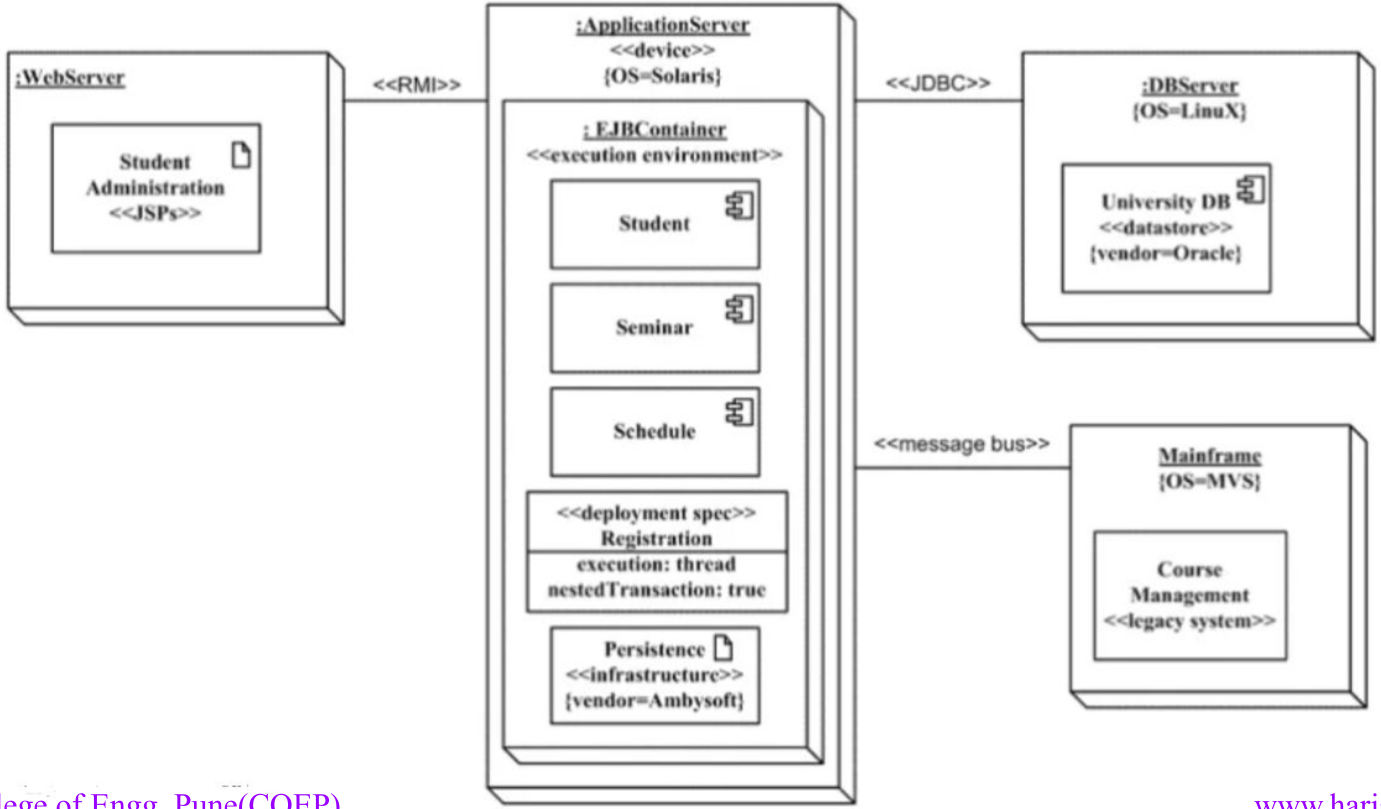
- It has two basic building blocks
  - Node
  - Artifacts (Ex Source file, executable file etc
- It depicts the static view of the runtime configuration of hardware nodes and the software components
- Deployment diagram shows the hardware for your system, the software that is installed on that hardware and the middleware used to connect the disparate machines to one another

Deployment Diagrams

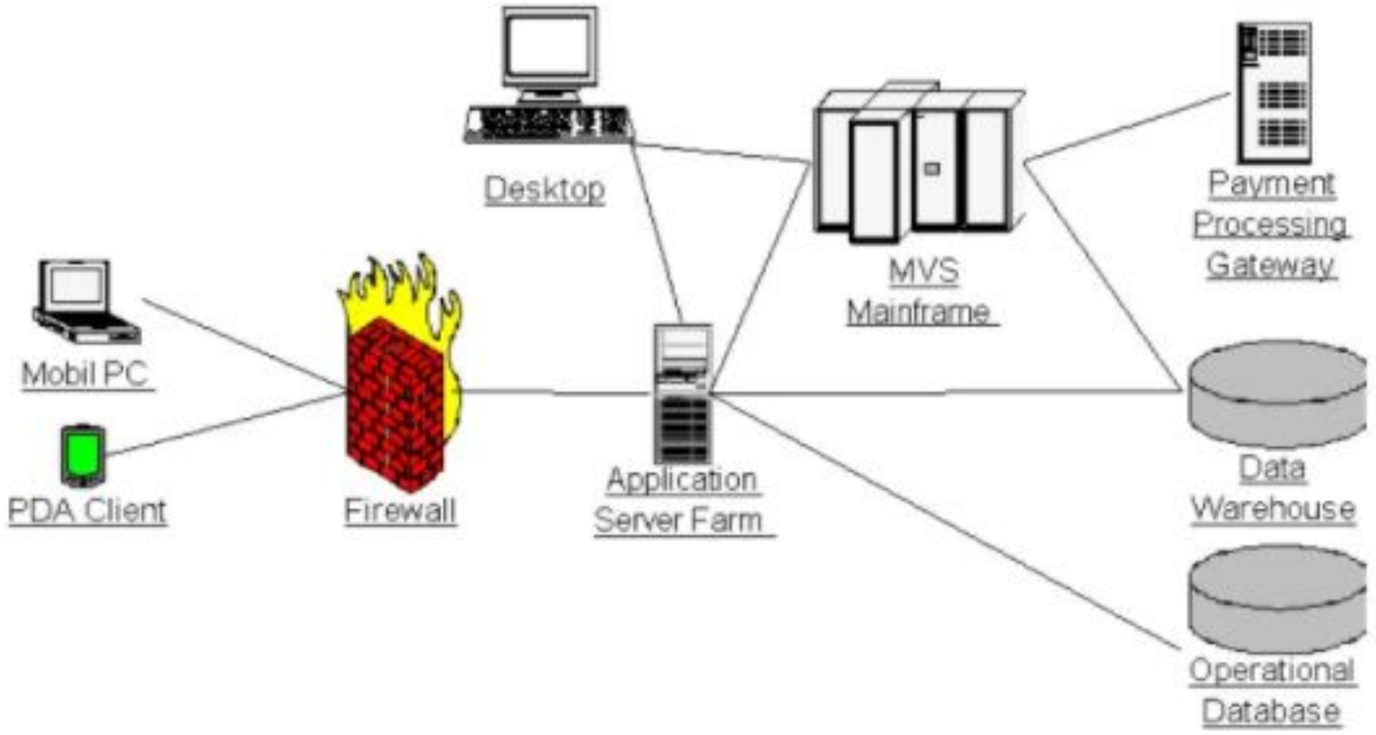




Deployment Diagrams



Deployment Diagrams


Desktop
MVS Mainframe
Payment Processing Gateway
Mobil PC
PDA Client
Firewall
Application Server Farm
Data Warehouse
Operational Database