Object Oriented Software Engineering (SE 313)
Lab Session # 02

To understand UML class diagram with their notation and relationships

Class Diagram

- Class diagrams are visual representations of the static structure and composition of a particular system using the conventions set by the Unified Modeling Language (UML).
- System designers use class diagrams as a way of simplifying how objects in a system interact with each other.
- Using class diagrams, it is easier to describe all the classes, packages, and interfaces that constitute a system and how these components are interrelated.
- Since class diagrams are used for many different purposes, such as making stakeholders aware of requirements to highlighting your detailed design, you need to apply a different style in each circumstance

Example

- Simple class diagram may be used to show how an organization such as a convenient store chain is set up.
- Precisely detailed class diagrams can readily be used as the primary reference for translating the designed system into a programming code.

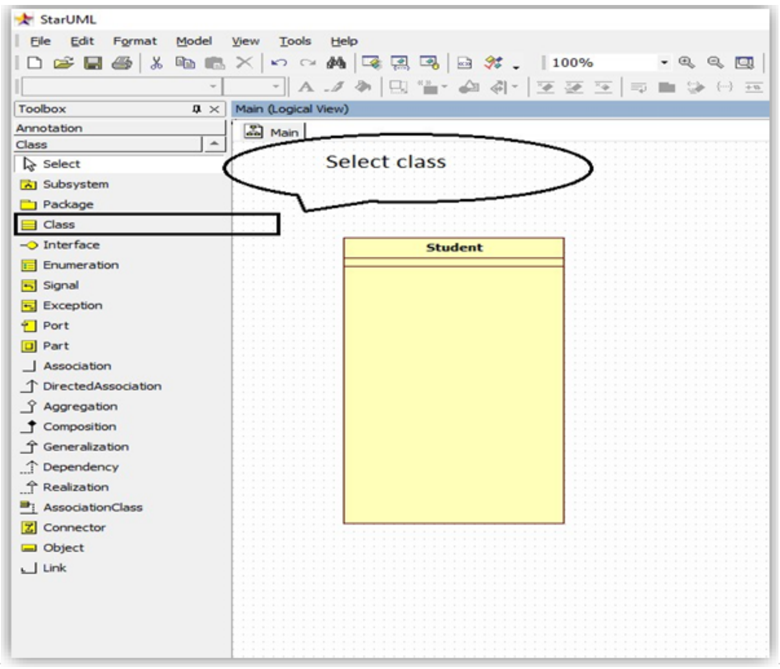
Notation of Class Diagram

1. Class

- An object is any person, place, thing, concept, event, screen, or report applicable to your system. Objects both know things (they have attributes) and they do things (they have methods).
- A class is a representation of an object and, in many ways, it is simply a template from which objects are created.
- Classes form the main building blocks of an object-oriented application.

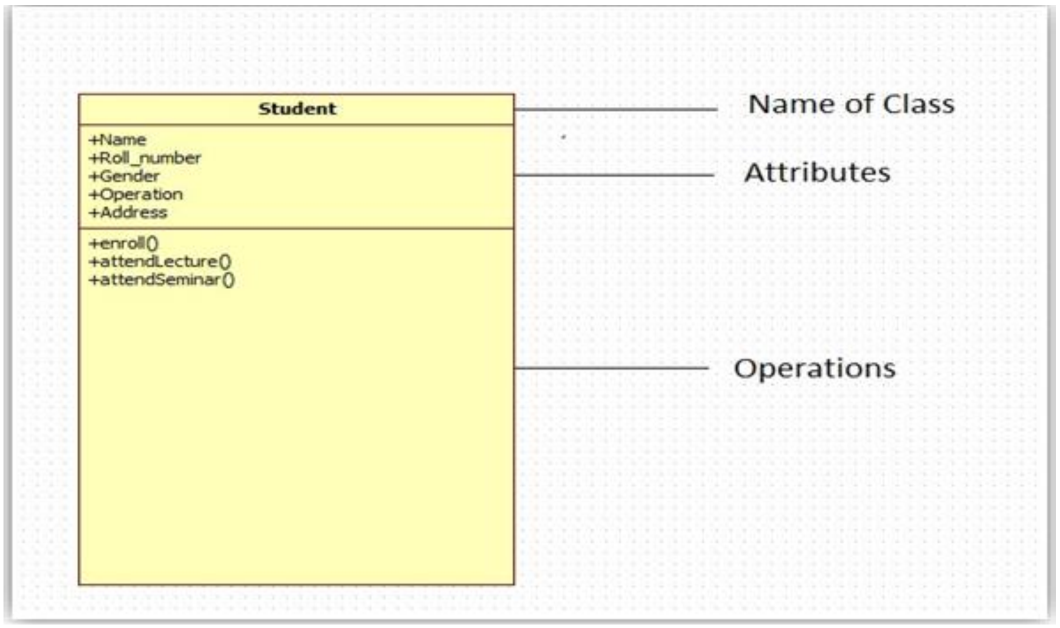
Example

Although thousands of students attend the university, you would only model one class, called Student, which would represent the entire collection of students.



The UML representation of a class is a rectangle containing three compartments stacked vertically.

- The top compartment shows the class's name.
- The middle compartment lists the class's attributes.
- The bottom compartment lists the class's operations.



2. **Attributes**

An attribute of a class represents a characteristic of a class that is of interest for the user.

The full format of the attribute text notation is:
Visibility name: type multiplicity = default [property-string]

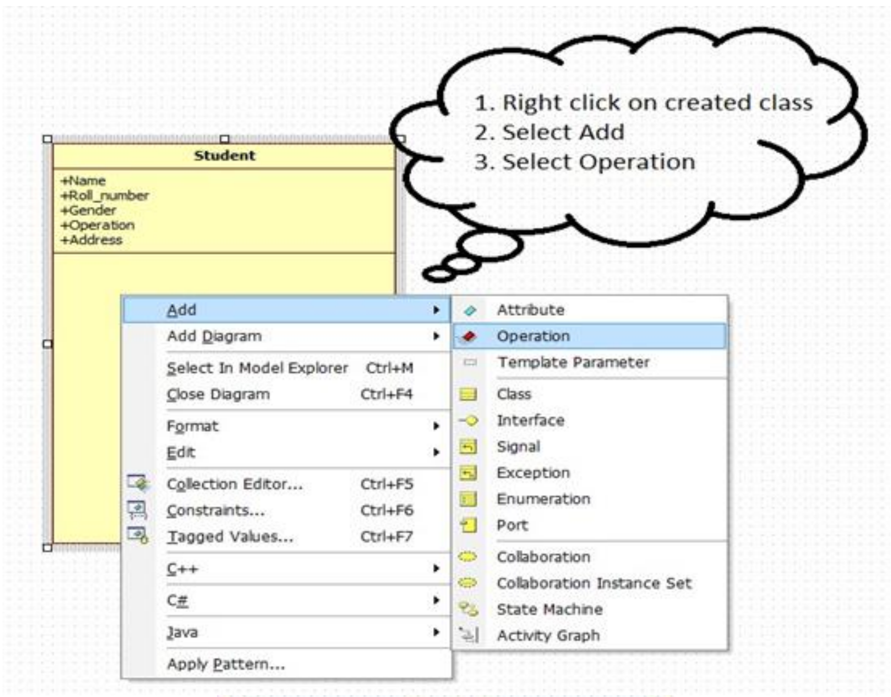
3. **Operations**

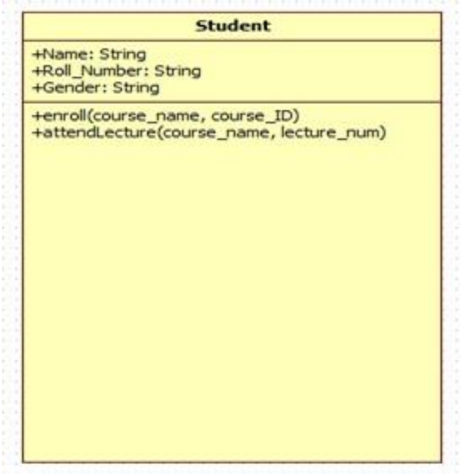
A UML operation is a declaration, with a name, parameters, return type, exceptions list, and possibly a set of constraints of pre and post conditions. But, it isn't an implementation – rather, methods are implementation.

4. **Visibility**:

Use visibility markers to signify who can access the information contained within a class.

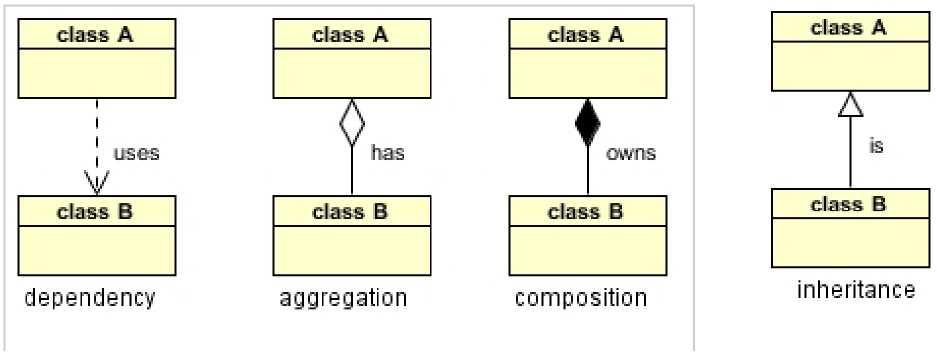
- Public +
- Private -
- Protected #
- Package ~


1. Right click on created class
2. Select Add
3. Select Operation
Student
+Name
+Roll_number
+Gender
+Operation
+Address
Add
Add Diagram
Select In Model Explorer Ctrl+M
Close Diagram Ctrl+F4
Format
Edit
Collection Editor... Ctrl+F5
Constraints... Ctrl+F6
Tagged Values... Ctrl+F7
C++
C#
Java
Apply Pattern...
Attribute
Operation
Template Parameter
Class
Interface
Signal
Exception
Enumeration
Port
Collaboration
Collaboration Instance Set
State Machine
Activity Graph


Student
+Name: String
+Roll_Number: String
+Gender: String
+enroll(course_name, course_ID)
+attendLecture(course_name, lecture_num)

Relationship

Dependency: class A uses class B
Aggregation: class A has a class B
Composition: class A owns a class B
Inheritance: class B is a Class A (or class A is extended by class B)



1. Association

An association is a "using" relationship between two or more objects in which the objects have their own life time and there is no owner.

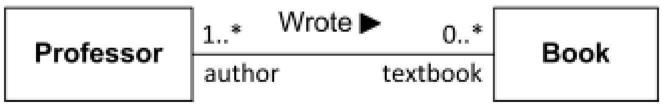
For **Example**: A patient may visit one or many doctors and same way, a doctor can be associated with multiple patients. If a patient dies, existence of doctor will not be vanished and similarly if doctor dies patient will remain patient.

Association is represented as thin line connecting two classes. Association can be unidirectional (shown by arrow at one end) or bidirectional (shown by arrow at both end) or without arrow.

Multiplicity defines how many instances can be associated at any given moment**.**

0..1	No instances or one instance	A flight seat can have no or one passenger only
1	Exactly one instance	An order can have only one customer
0..* or *	Zero or more instances	A class can have zero or more students.
1..*	One or more instances (at least one)	A flight can have one or more passenger

Example:



*Association **Wrote** between **Professor** and **Book** with association ends **author** and **textbook**.*

2. Aggregation

Aggregation is a special form of association. It is also a relationship between two classes like association, however, it's a **directional** association, which means it is strictly a **one way association, means unidirectional association.** It represents a **Has-A** relationship.

For Example: Consider two classes Student class and Address class. Each student must have an address so the relationship between student and address is a Has-A relationship. But if you consider its vice versa then it would not make sense as an Address doesn't need to have a Student necessarily.

NOTE: Unarguably, Address is an attribute of a student, but here in this example I am breaking address into several fields i.e city, province and country. This is the reason for making address a class**.**

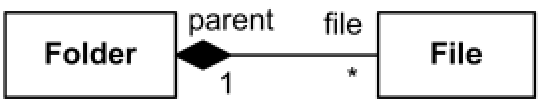
3. Composition

Composition is a special case of aggregation. In a more specific manner, a restricted aggregation is called composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition.

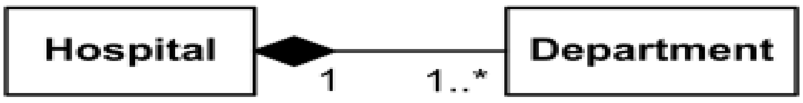
For **Example**: Consider the same scenario with some modifications. In this scenario, a student has address and each student has different address (Please keep sibling relationship argument apart). So, when a student record is added his house number and street number will be entered. And if I delete the record of a particular student, then his/her record will be of no use.

filled black diamond at the aggregate (whole) end.

Example:



Folder could contain many files, while each File has exactly one Folder parent.
If Folder is deleted, all contained Files are deleted as well.



Hospital has 1 or more Departments, and
each Department belongs to exactly one Hospital.
If Hospital is closed, so are all of its Departments.

4. Generalization

In object oriented programming, the concept of IS-A is a totally based on Inheritance, which can be of two types Class Inheritance or Interface Inheritance. It is just like saying "A is a B type of thing". For example, Apple is a Fruit, Car is a Vehicle etc. Inheritance is uni-directional. For example House is a Building. But Building is not a House.
It is key point to note that you can easily identify the IS-A relationship. Wherever you see an extends keyword or implements keyword in a class declaration, then this class is said to have IS-A relationship.

Example: Refer your theory lectures.

Student Tasks

Task 1: Create a UML Class diagram of the below scenario also include all possible relations

1. An **Order** is ordered by a **Customer**.
2. An **Order** is fulfilled by an **Employee**.
3. An **Order** is paid via a **Payment Method**.
4. An **Order** is shipped via an **Address** belonging to the **Customer** who is the buyer.
5. An **Order** is composed of **Order Items**.

Note: Add appropriate attributes and methods according to the order management system

Task 2: Create a UML Class diagram of the below scenario

Project Management System

Organizations, projects, team, employee and employee roles in teams. An organization relates to zero or more projects, zero or more teams and zero or more people who are employee of the organization. A project related to a single organization and to a single team. A team relates to a single organization to a single project. A person relates to a single organization that is an employer. A team relates to a zero or more people as member of the team in which a person plays a role. A person relates to a single team in which a person plays a role.

A project has name that is a string, a start and end date that are strings, a budget that is a real number and an operation to ensure that the start date and end date of a project are valid. Each team and organization has a name that is a string. A person has an identification number that is an integer, a name that is string, hours that they are available to work, and an operation to determine whether the number of hours that they are available to work, and an operation to determine whether the number of hours that they are available to work is within the range of minimum and maximum number of hours. The relationship between a person and a team defines the title as a string of the role that the person plays on the team. All the attributes and operations are public, but a project’s start date, end date and the hours they are available to work are private

Task 3 Create a UML Class diagram of the given School Management System.

- A school has one or more departments
- A department can only belong to one school
- A school has one or more students.
- A student can attend one or more courses
- A courses can have one or more students
- A course is taught by one teacher only
- A teacher can teach one or more courses
- A teacher is assigned one or more departments
- A department can have one or more teachers.
- A teacher can be the chairman of zero or one department.
- A department can have zero or one chairman (teacher)
- A course belongs to one or more departments.
- A department can have one or more courses etc.