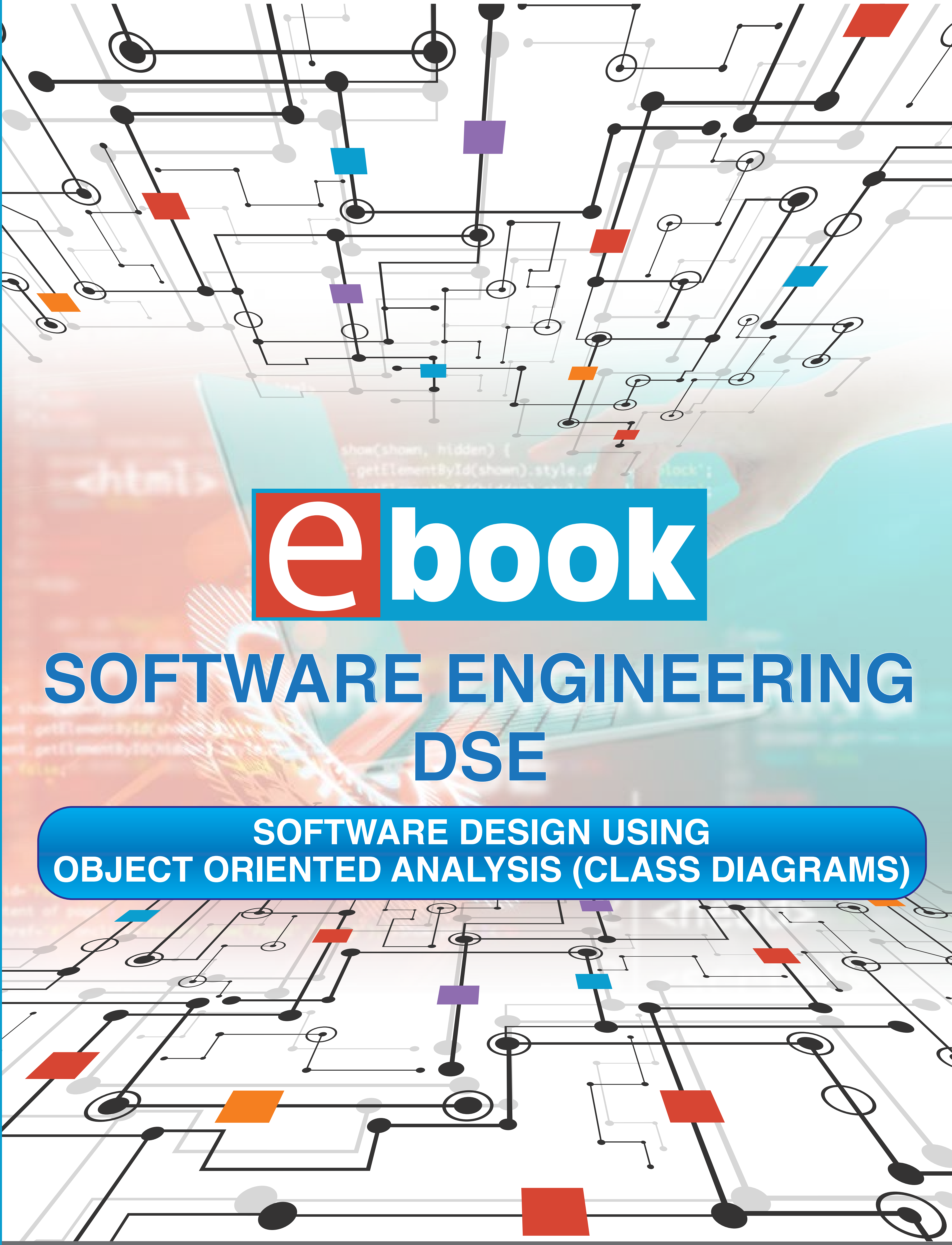
NIBM WORLD WIDE



ebook

SOFTWARE ENGINEERING DSE

SOFTWARE DESIGN USING OBJECT ORIENTED ANALYSIS (CLASS DIAGRAMS)

Lesson 07 – Software Design Using Object Oriented Analysis (Class Diagrams)

Class Diagram

Class Diagram describes the structure of a system by showing the system's

- Classes
- Attributes of the Classes
- Operations (or methods) of the Classes
- Relationships among Objects

A Class is a **Blueprint for an Object**. We can't talk about Class without talking about the Object. Objects have states and behaviors. Each Object was built from the same set of blueprints and therefore contains the same components (properties and methods).

The standard meaning is that an object is an instance of a class and object.

Example - A dog has states - color, name, breed as well as behaviors -wagging, barking, eating. An object is an instance of a class.

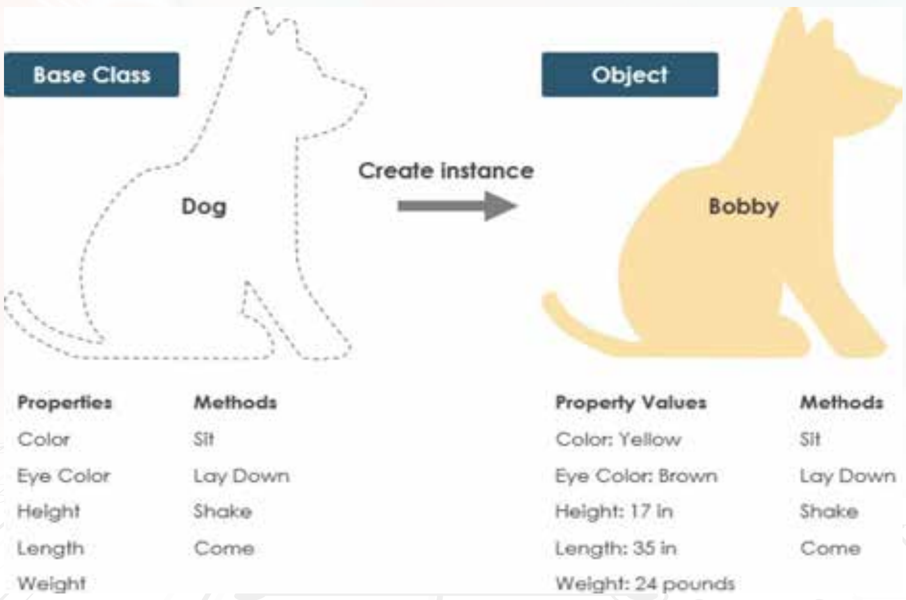


Figure 7.0.1 What is a Class?

Class Diagram Notation and Symbol

- A Class represent a concept which Encapsulates state (attributes) and behavior (methods).
- A Class Notation consists of three parts:

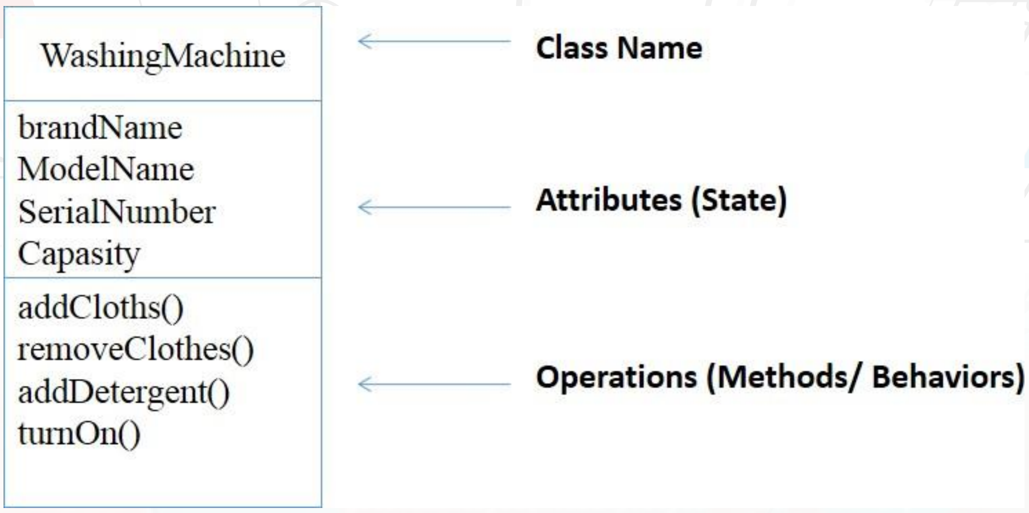


Figure 7.0.2 Class Notation

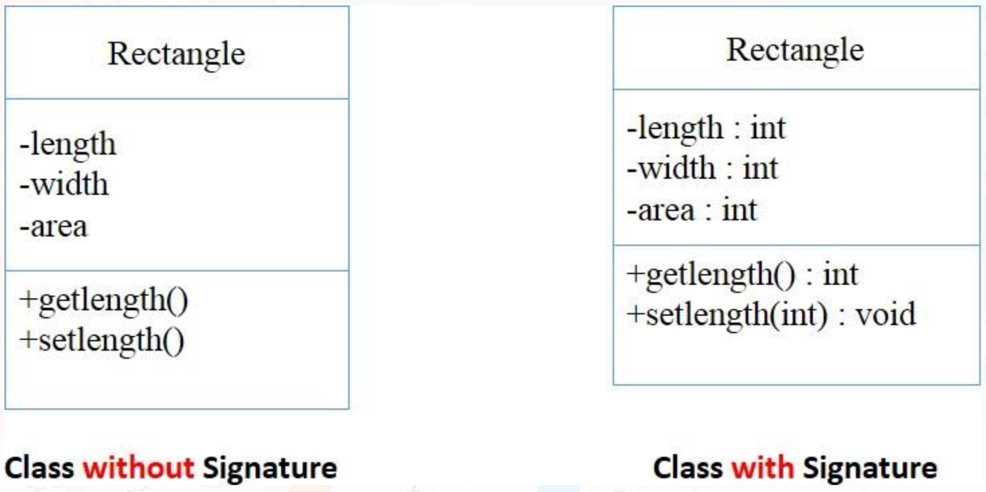


Figure 7.0.3 Class with & without signature

Class Name:

- The name of the class appears in the first partition.

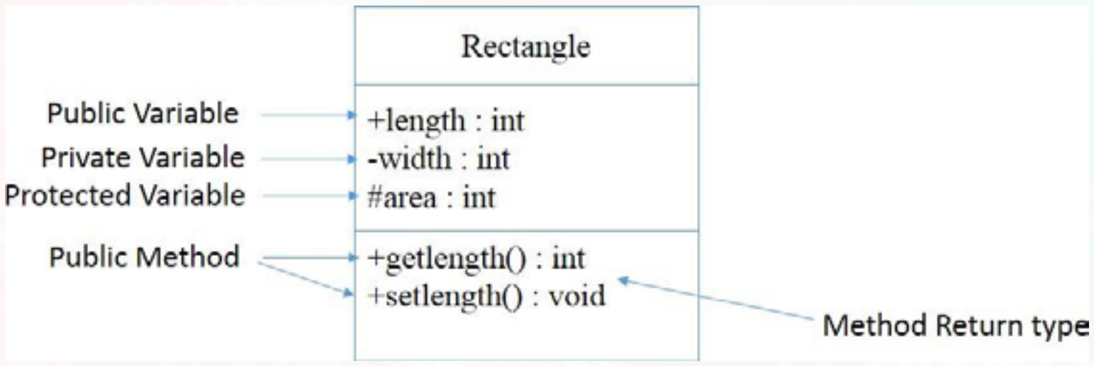
Class Attributes:

- Attributes are shown in the second partition.
- The attribute type is shown after the colon.
- Attributes map onto member variables (data members) in code.

Class Operations (Methods):

- Operations are shown in the third partition. They are services the class provides.
- The return type of a method is shown after the colon at the end of the method signature.
- The return type of method parameters is shown after the colon following the parameter name. Operations map onto class methods in code.

Class Diagram Visibility



The +, - and # symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.

+ denotes public attributes or operations

- denotes private attributes or operations

# denotes protected attributes or operations

Class Diagram Visibility with all the options

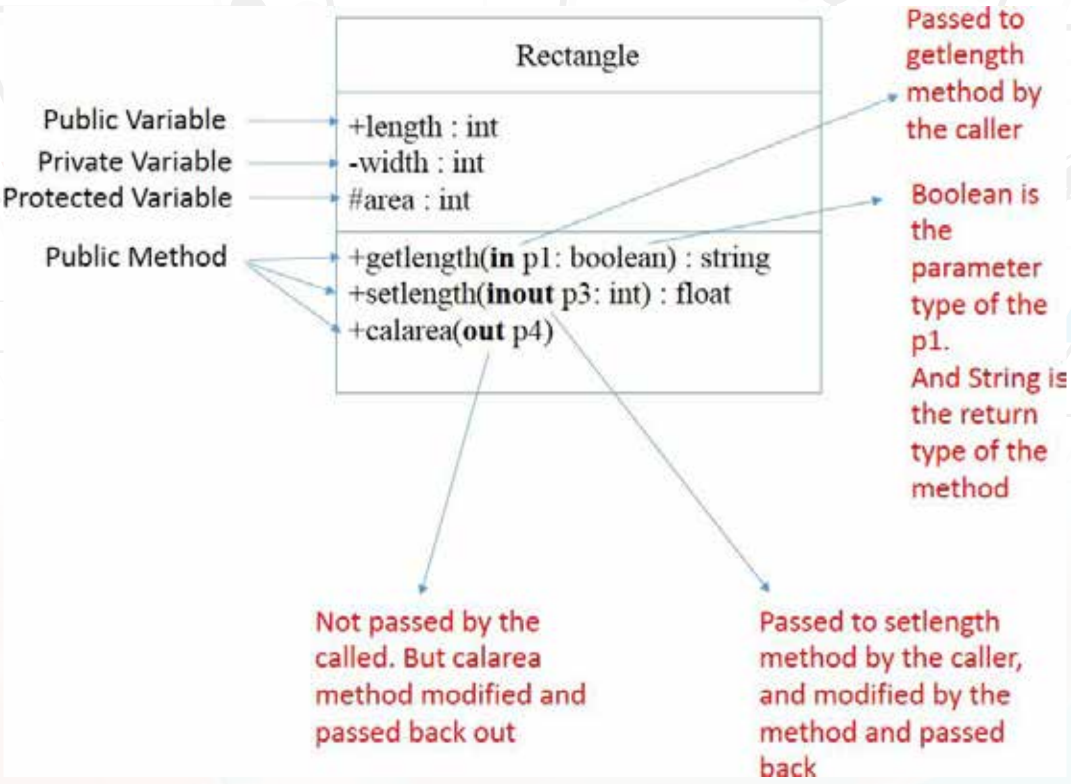

Rectangle
Public Variable
Private Variable
Protected Variable
+length : int
-width : int
#area : int
Public Method
+getlength(**in** p1: boolean) : string
+setlength(**inout** p3: int) : float
+calarea(**out** p4)
Passed to getlength method by the caller
Boolean is the parameter type of the p1. And String is the return type of the method
Not passed by the called. But calarea method modified and passed back out
Passed to setlength method by the caller, and modified by the method and passed back

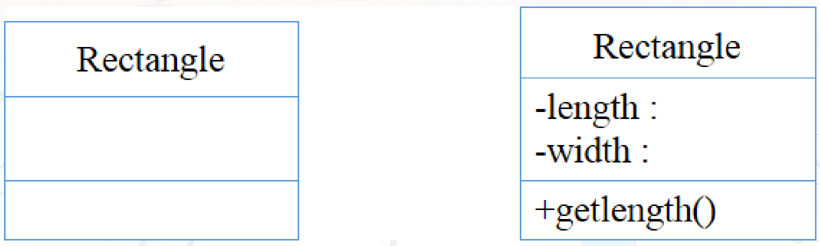
Figure 7.0.4 Class Diagram Visibility with all the options

Perspectives of Class Diagram

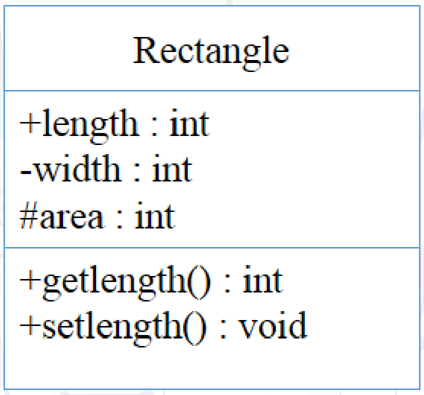
A diagram can be interpreted from various perspectives:

- **Conceptual**: Represents the concepts in the domain

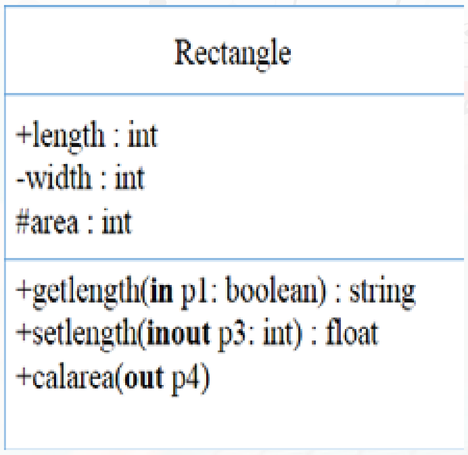
The class name is the only mandatory information.



- **Specification**: Focus is on the interfaces of Abstract Data Type (ADTs) in the software.



- **Implementation**: Describes how classes will implement their interfaces.


Rectangle
+length : int
-width : int
#area : int
+getlength(**in** p1: boolean) : string
+setlength(**inout** p3: int) : float
+calarea(**out** p4)

Relationships between Classes

A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:

Inheritance (or Generalization)

- Represents an "is-a" relationship.
- An abstract class name is shown in italics.
- SubClass1 and SubClass2 are specializations of Super Class.
- A solid line with a hollow arrowhead that point from the child to the parent class.

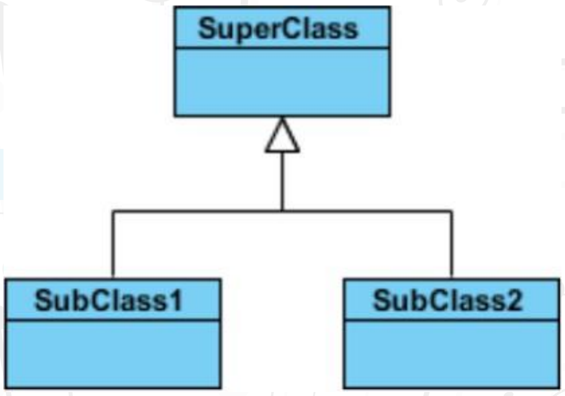
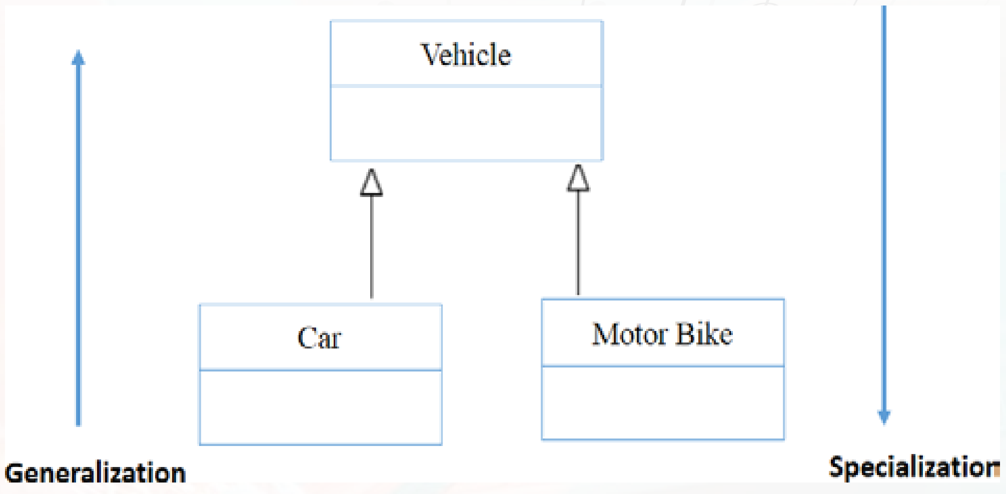


Figure 7.0.5 Relationship between classes - Inheritance

Example for Inheritance (or Generalization):

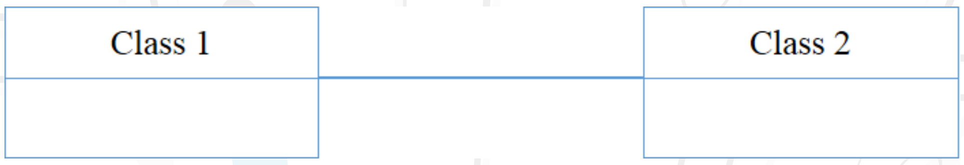


Association

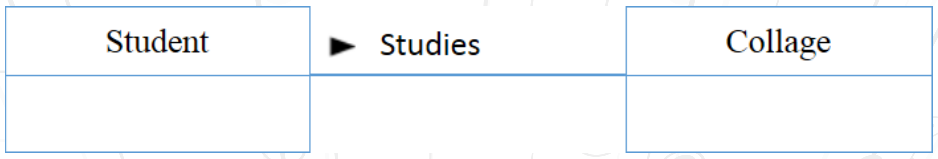
- Associations are relationships between classes in a UML Class Diagram. They are represented by a solid line between classes.
- Associations are typically named using a verb or verb phrase which reflects the real world problem domain.

Simple Association

- A structural link between two peer classes.
- There is an association between Class1 and Class2.



Example for Simple Association:



Names of relationships are written in the middle of the association line. They often have a **small arrowhead to show the direction** in which direction to read the relationship.

Multiplicity

- How many objects of each class take part in the relationships?

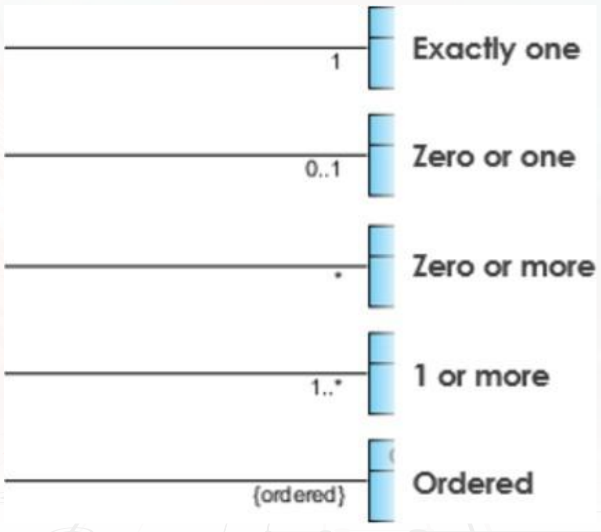
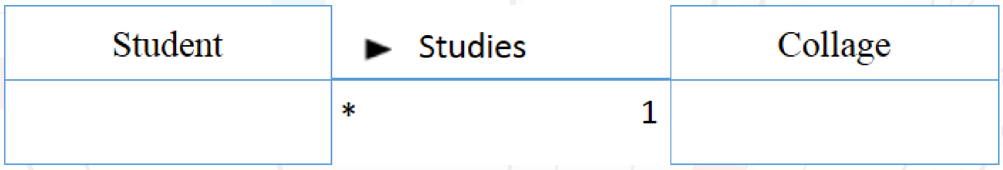


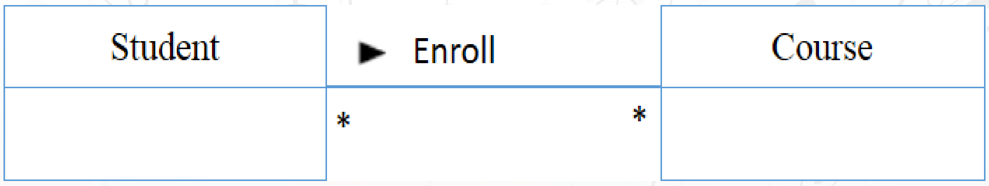
Figure 7.0.6 Multiplicity

Examples for Multiplicity:

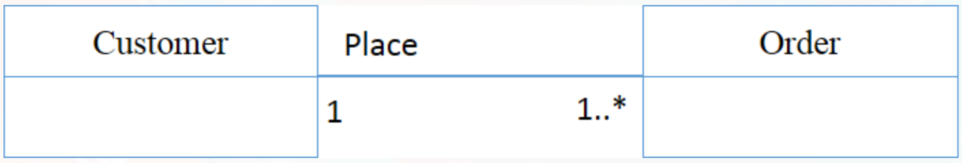
Ex: The college can have multiple students.



Ex: A Student can take many Courses and many Students can be enrolled in one Course.



Ex: One Customer can place one or more orders.



Aggregation

- **A special type of Association.**
- It represents a "part of" relationship. ("Consists-of" hierarchy)
- Class2 is part of Class1.
- Many instances (denoted by the *) of Class2 can be associated with Class1.
- Objects of Class1 and Class2 have separate lifetimes.

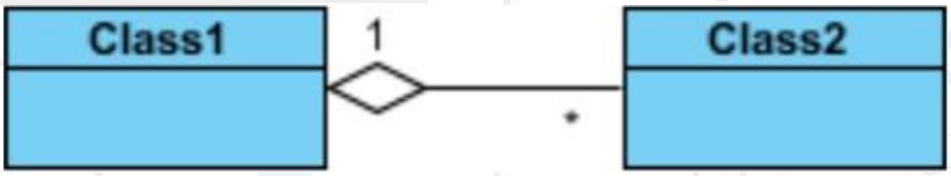
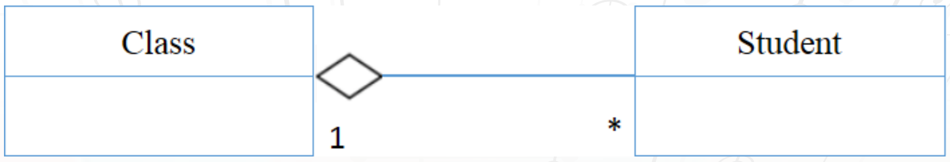


Figure 7.0.7 Aggregation

Example for Aggregation:



For example, the Class is made up of one or more student. In aggregation, the contained classes are never totally dependent on the lifecycle of the container. Here, the **Class will remain even if the student is not available.**

Composition:

- **A special type of Aggregation** where parts are destroyed when the whole is destroyed.
- Objects of Class2 live and die with Class1.
- Class2 cannot stand by itself.
- A solid line with a filled diamond at the association connected to the class of composite.

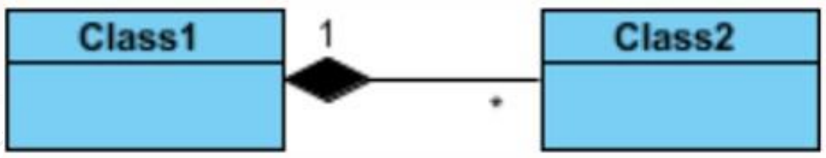
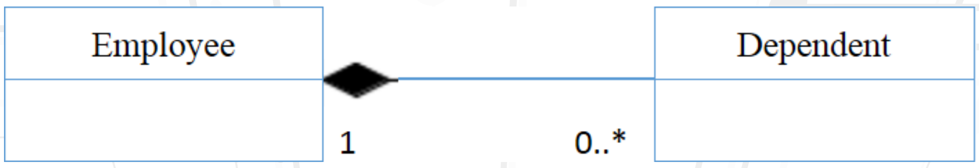


Figure 7.0.8 Composition

Example for Composition:



For example, the Employee have zero or many dependents. This relationship is composition relationship **because without the Employee class Dependent class cannot be exit.**

Dependency:

- Exists between two classes if the changes to the definition of one may cause changes to the other (but not the other way around).
- Class1 depends on Class2.
- A dashed line with an open arrow.

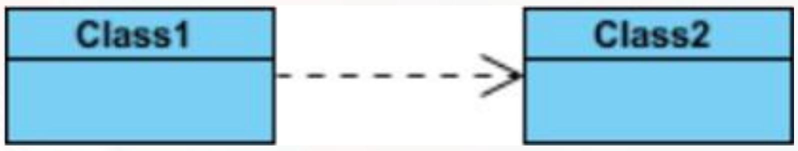
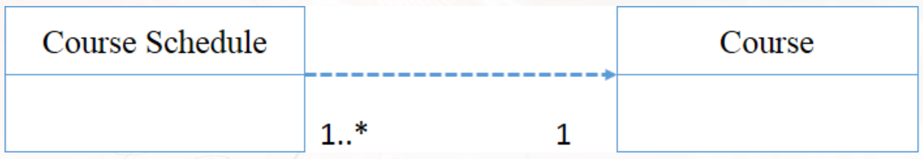


Figure 7.0.9 Dependency

Example for Dependency:



Here Course Schedule class is depending on the Course. If something changed in the Course Automatically Course Schedule can be changed.

The figure below shows the three types of association connectors: **Association, Aggregation, and Composition.**

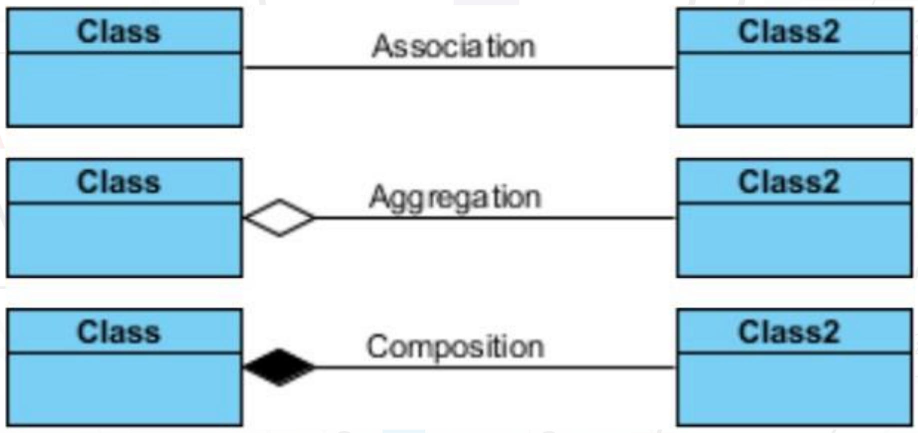


Figure 7.0.10 Association, Aggregation, and Composition

The figure below shows Generalization (Inheritance).

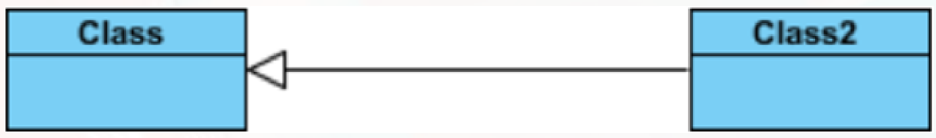


Figure 7.0.11 Generalization (Inheritance)

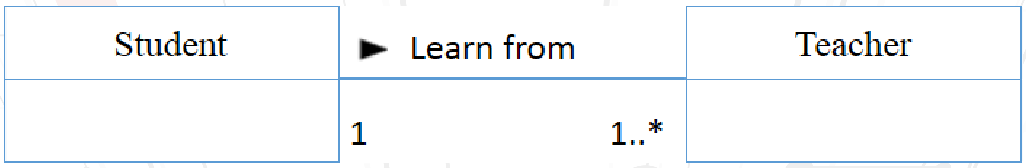
Exercise

What kind of a relationship can be exits in following Classes?

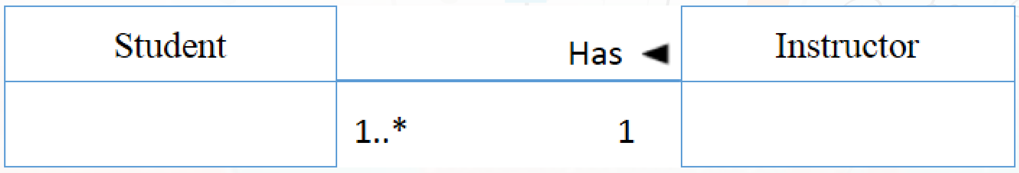
1. A Single Student can associate with Multiple teachers:
2. Every Instructor has one or more Students:
3. Customer has one or more Accounts:

Answers

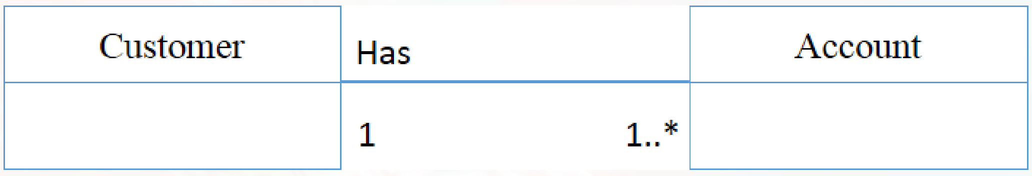
1. A single student can associate with Multiple teachers:



2. Every Instructor has one or more Students:



3. Customer has one or more Accounts:



Aggregation vs Composition

- **Aggregation** and **Composition** are subsets of association meaning they are **specific cases of association**.
- In both aggregation and composition object of one class "owns" object of another class. But there is a subtle difference:

- **Aggregation** implies a relationship where the **child can exist independently of the parent.** Example: Class (parent) and Student (Child). Delete the Class and the Students still exist.

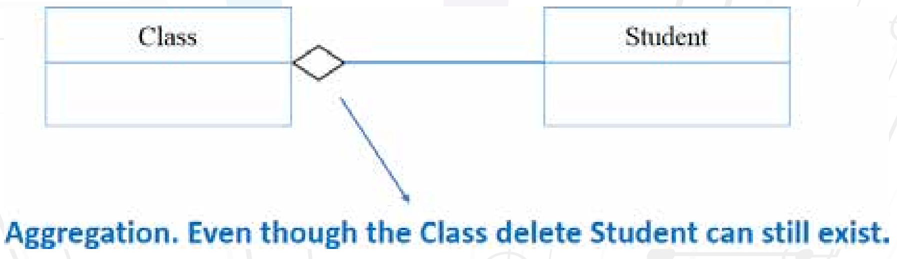

Class
Student
Aggregation. Even though the Class delete Student can still exist.

Figure 7.0.12 Aggregation

- **Composition** implies a relationship where the child cannot exist independent of the parent. Example: House (Parent) and Room (Child). Rooms don't exist separate to a House.

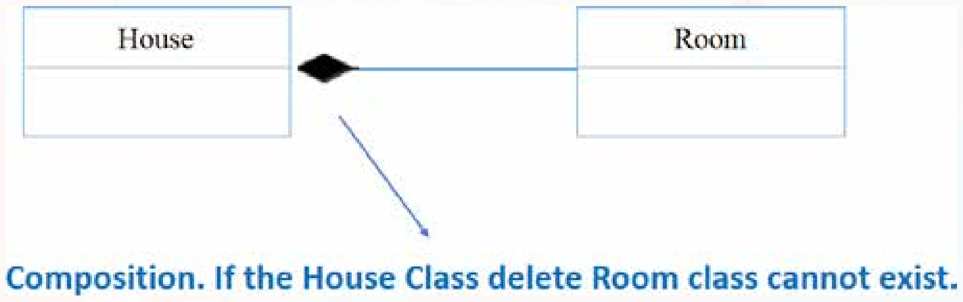
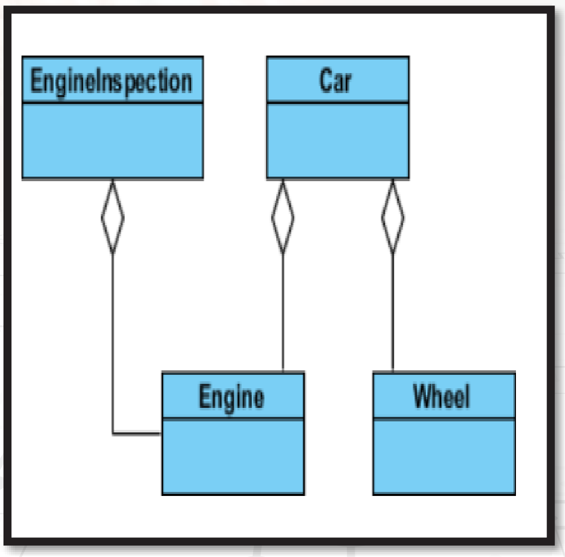
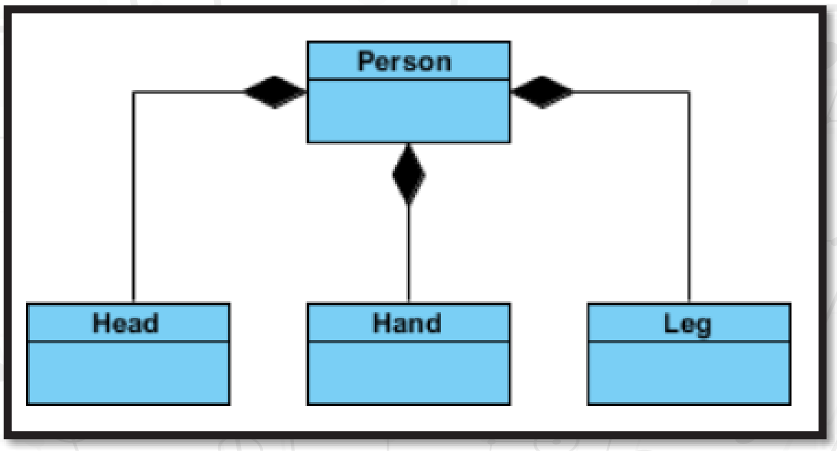

House
Room
Composition. If the House Class delete Room class cannot exist.

Figure 7.0.13 Composition

Aggregation



Composition



Exercise 1: ATM System

Bank has different ATM machines in different locations managed by different branches. Each Bank Branch has a Code and an Address. Using the ATM, Customers can do the Transactions.

Customer has an id, name, address, card number and a pin number. Before do any transactions password should be verified. Customers have Accounts. Account has a number and balance. Deposit and Withdraw can be done through Account.

Through the Account, ATM Transactions can be maintained. For each ATM Transaction date, transaction id, amount, post balance recorded. Savings Accounts and Current Accounts are the two types of Bank Accounts. Both accounts have an account no and balance. When creating a Current Account Bank always checks the Savings in the Saving Account.

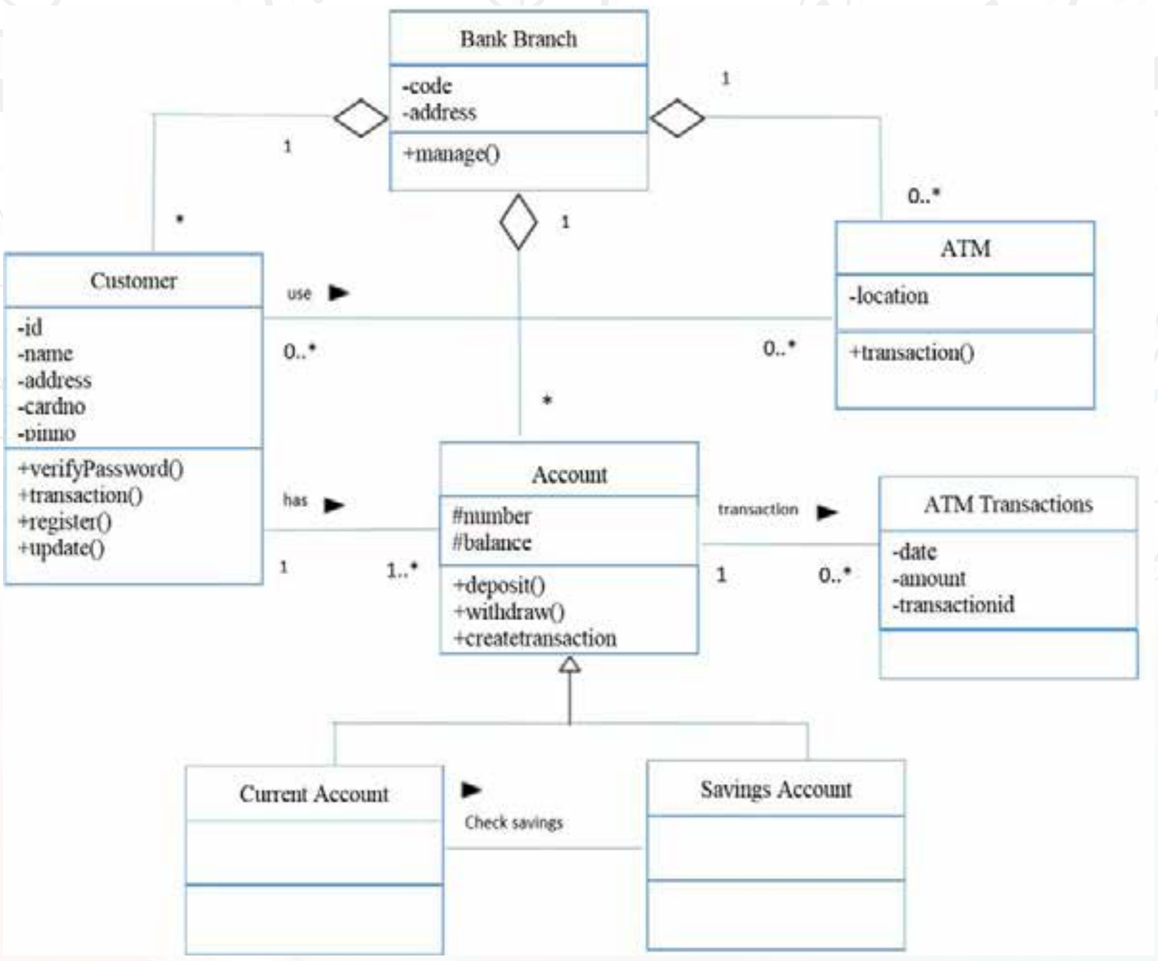


Figure 7.0.14 ATM System – Exercise 1

Exercise 2: Student Management System

At NIBM there are different departments which offer different courses to the Students. Each department and course has its own id and name. A Student can register under a course which offer by specific department and student's id, name, address and NIC should maintain. Students are categorized as Part Time and Full Time students where Part Time student need job experience to follow part time course. In each department there are Lecturers who conduct lectures for different courses. Lecturer's Id, Name, qualification need to be record.

Lecturer create course schedule, send to students and course schedule is depends on the course and the availability of the Hall. Each hall has a hall number and a capacity. Lecturers are responsible for the courses. You have freedom to add required methods (operations) for each Classes by analyzing the question. Ex: In Student Class following methods can be added. Add_Student(), Update_Student(), Inactive_Student(), Transfer_Student etc.

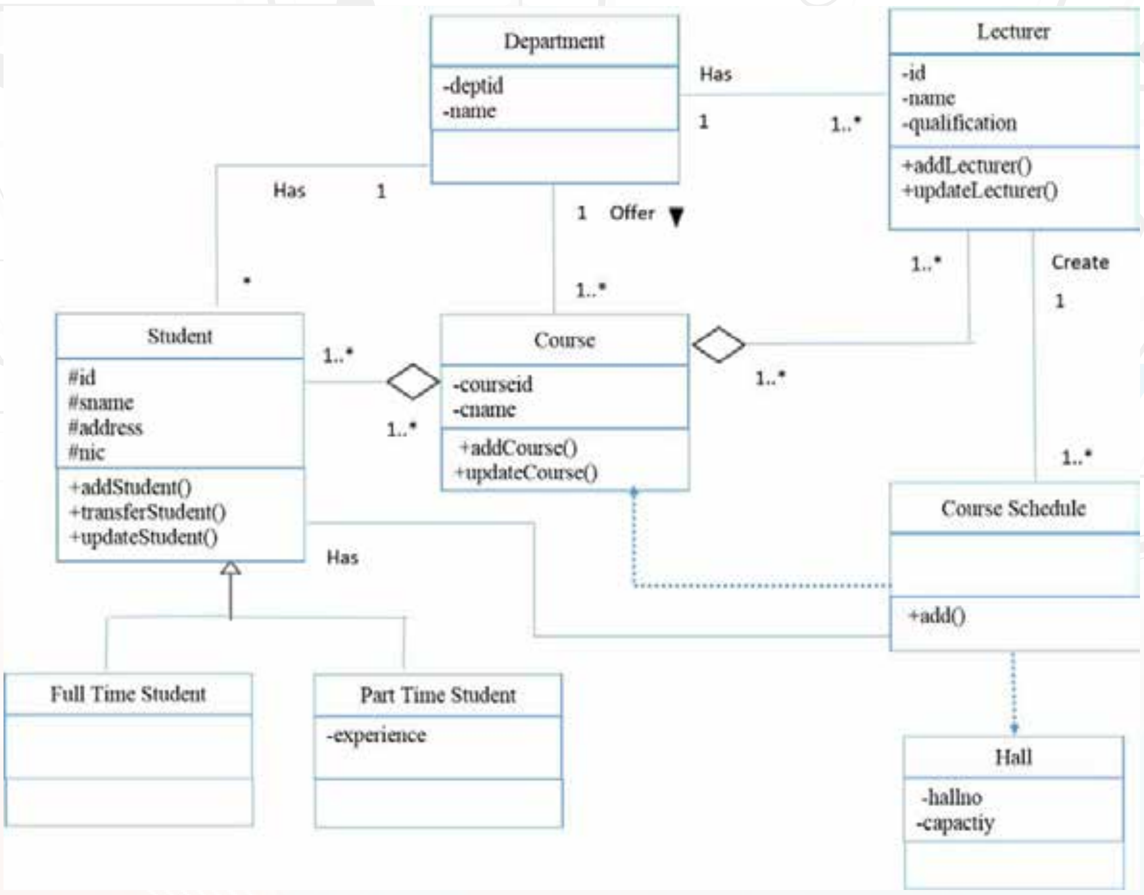


Figure 7.0.15 Student Management System – Exercise 2

Exercise 3: Online Shopping System

Food City wants to develop an Online Shopping System where Customer can register using Customer Name and Contact Number, then Customer Can Place his/her order. The system should record Order Id, Item details and Delivery Address.

The Order can view and update the order status by the Customer Service. Also they can add, update, delete items from the system. System should register suppliers and supplier will get purchase order whenever food city wants to order items from the supplier. Once the items delivered Good Receive Note created. Once the customer does the payments invoice should be created. Also Customer can return the purchased goods and Food City can return the goods back to the Supplier.

You have freedom to add required methods (operations) for each Classes by analyzing the question. Ex: In Supplier Class following methods can be added. Add_Supplier(), Update_Supplier(), Inactive_Supplier().

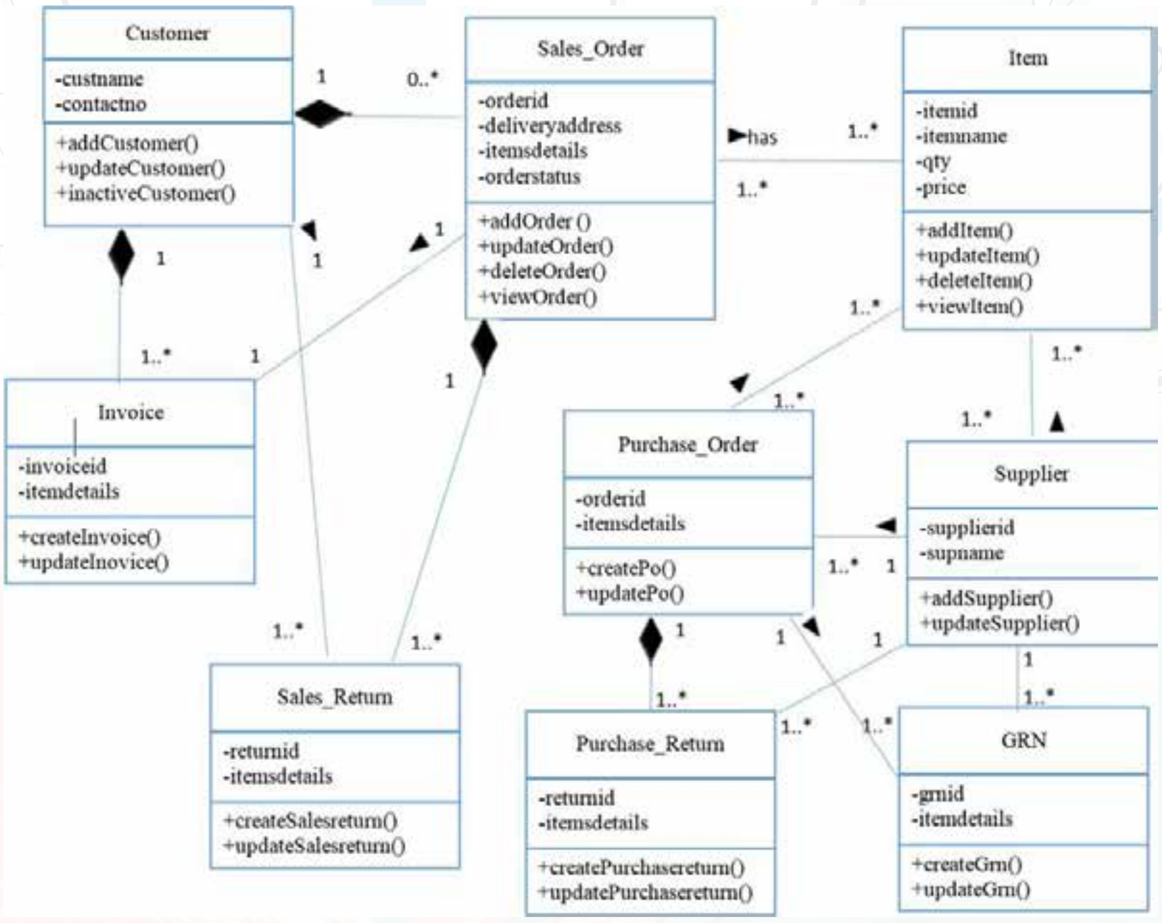


Figure 7.0.16 Online Shopping System – Exercise 3