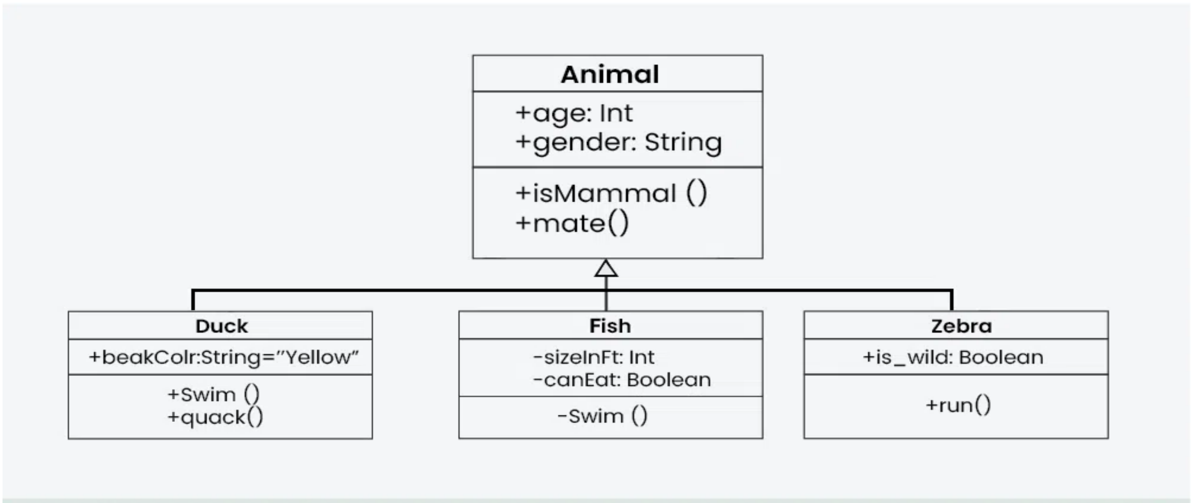
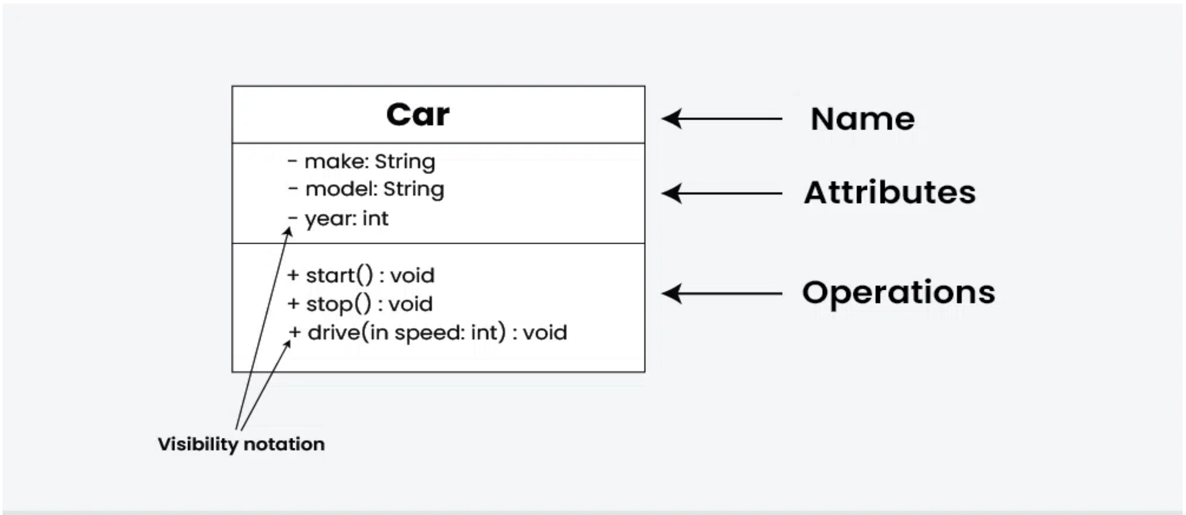
Class diagrams

Class diagrams are a type of UML diagram used in software engineering to visually represent the structure and relationships of classes within a system i.e. used to construct and visualize object-oriented systems. Class diagrams provide a high-level overview of a system's design, helping to communicate and document the structure of the software. They are a fundamental tool in object-oriented design and play a crucial role in the software development lifecycle.

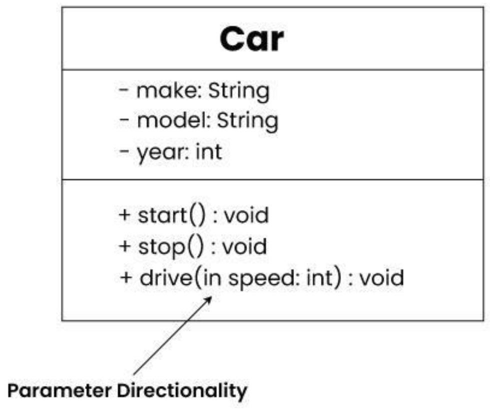


Class Diagram example

In object-oriented programming (OOP), a class is a blueprint or template for creating objects. Objects are instances of classes, and each class defines a set of attributes (data members) and methods (functions or procedures) that the objects created from that class will possess. The attributes represent the characteristics or properties of the object, while the methods define the behaviors or actions that the object can perform.

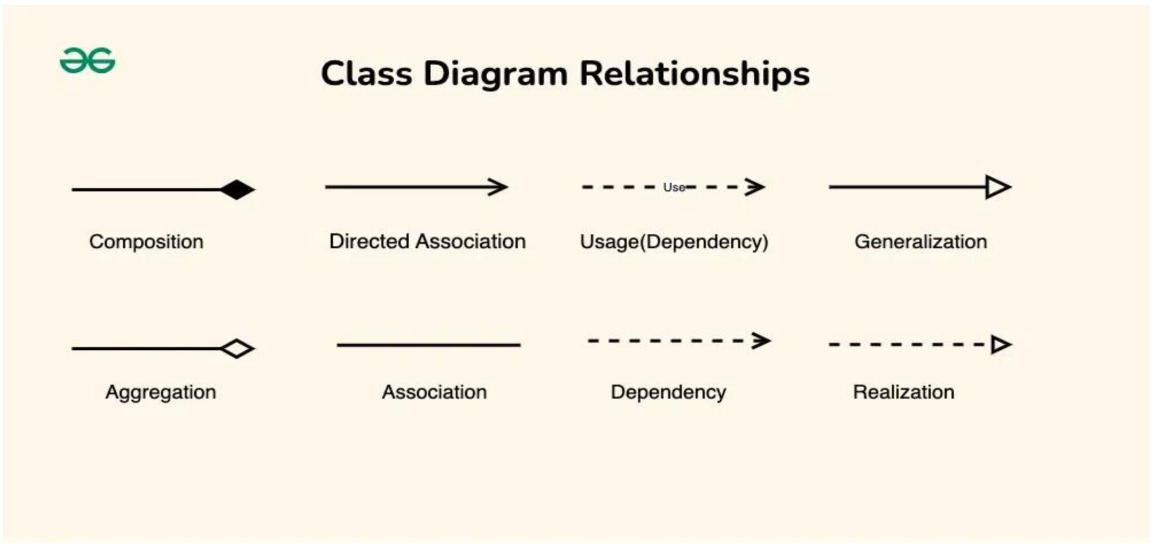


Class Notation



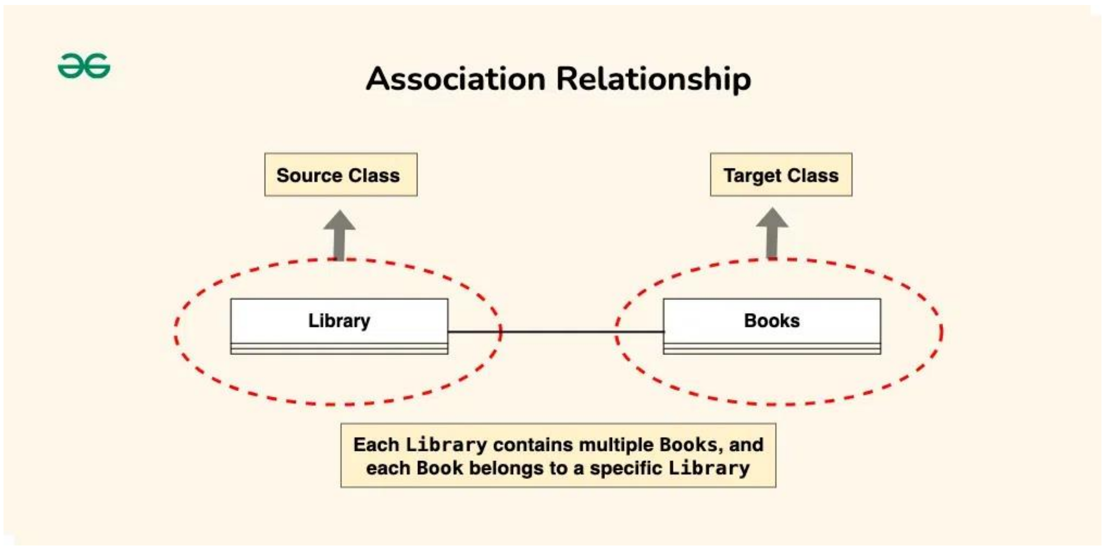
class notation with parameter directionality

In class diagrams, relationships between classes describe how classes are connected or interact with each other within a system. There are several types of relationships in object-oriented modeling, each serving a specific purpose.



Association

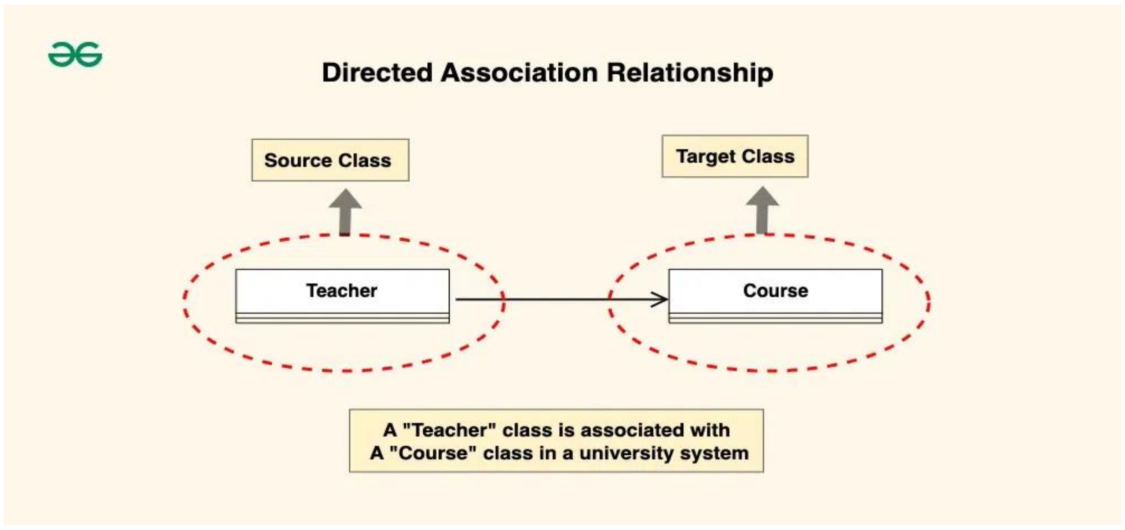
An association represents a bi-directional relationship between two classes. It indicates that instances of one class are connected to instances of another class.


Association Relationship
Source Class
Target Class
Library
Books
Each Library contains multiple Books, and each Book belongs to a specific Library

The "Library" class can be considered the source class because it contains a reference to multiple instances of the "Book" class. The "Book" class would be considered the target class because it belongs to a specific library.

Directed Association

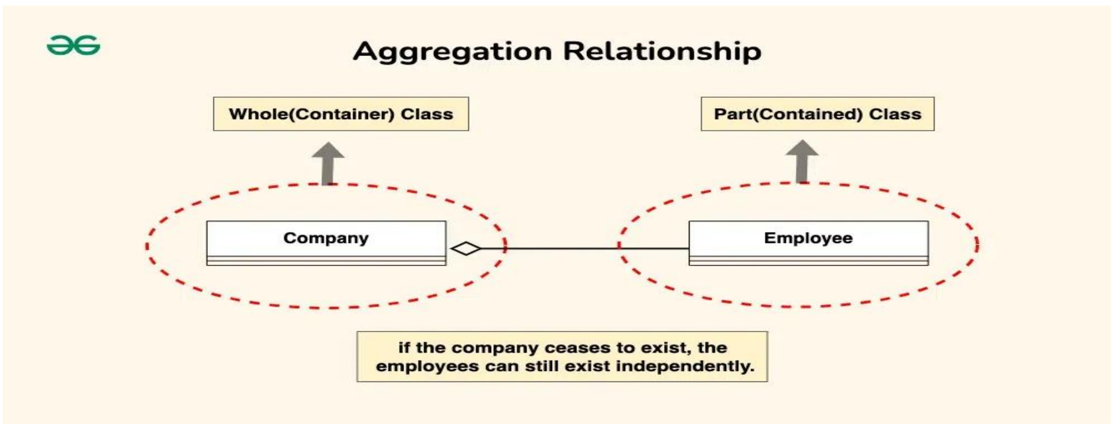
A directed association in a UML class diagram represents a relationship between two classes where the association has a direction, indicating that one class is associated with another in a specific way.


Directed Association Relationship
Source Class
Target Class
Teacher
Course
A "Teacher" class is associated with A "Course" class in a university system

Aggregation

Aggregation is a specialized form of association that represents a "whole-part" relationship. It denotes a stronger relationship where one class (the whole) contains or is composed of another class (the part). Aggregation is represented by a diamond shape on the side of the whole class. In this kind of relationship, the child class can exist independently of its parent class.

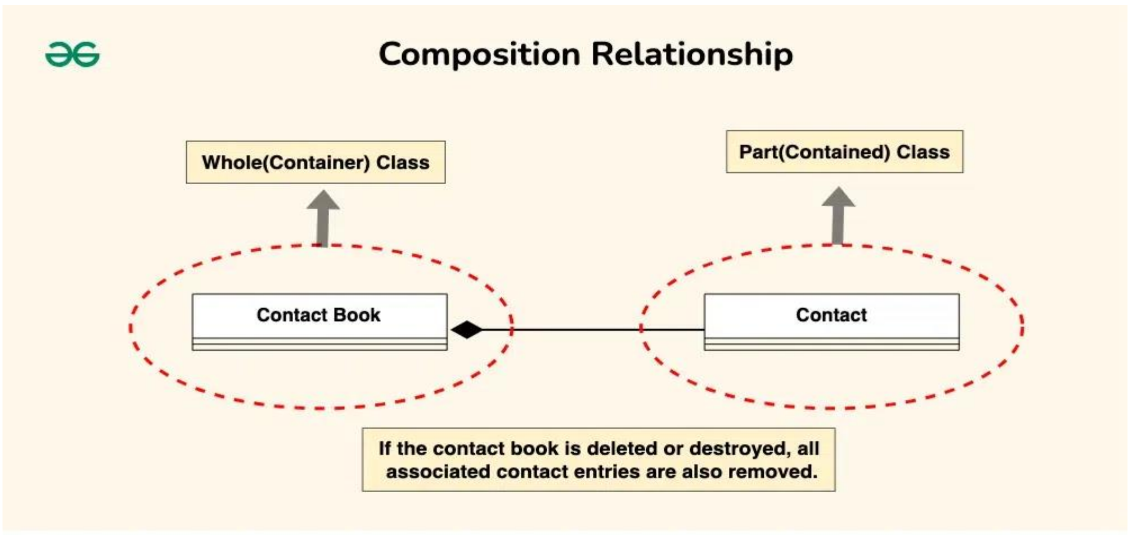
The company can be considered as the whole, while the employees are the parts. Employees belong to the company, and the company can have multiple employees. However, if the company ceases to exist, the employees can still exist independently.



Composition

Composition is a stronger form of aggregation, indicating a more significant ownership or dependency relationship. In composition, the part class cannot exist independently of the whole class. Composition is represented by a filled diamond shape on the side of the whole class.

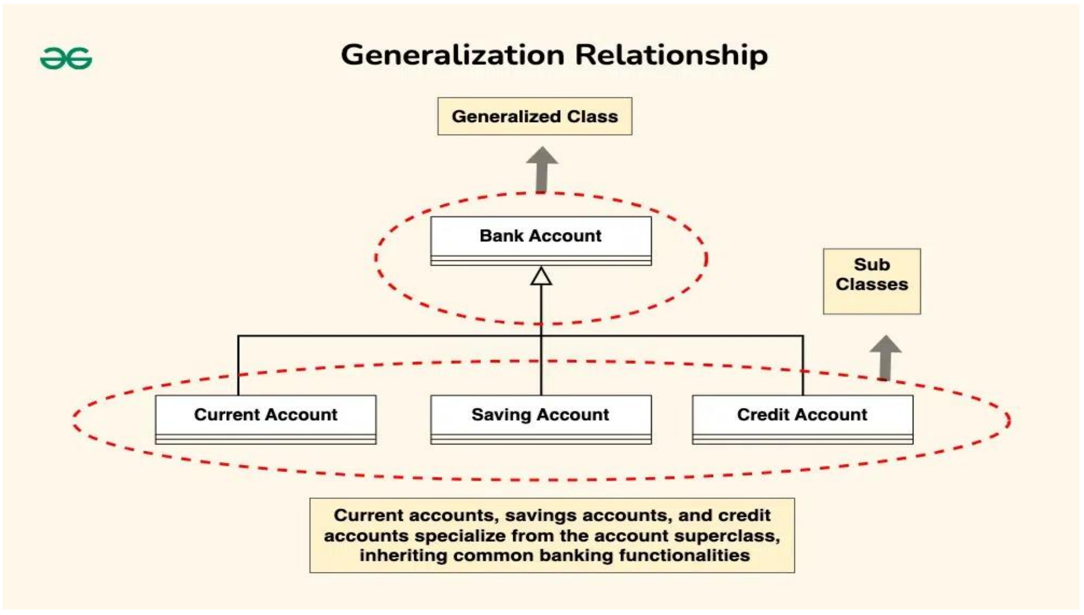
Imagine a digital contact book application. The contact book is the whole, and each contact entry is a part. Each contact entry is fully owned and managed by the contact book. If the contact book is deleted or destroyed, all associated contact entries are also removed.


Composition Relationship
Whole(Container) Class
Part(Contained) Class
Contact Book
Contact
If the contact book is deleted or destroyed, all associated contact entries are also removed.

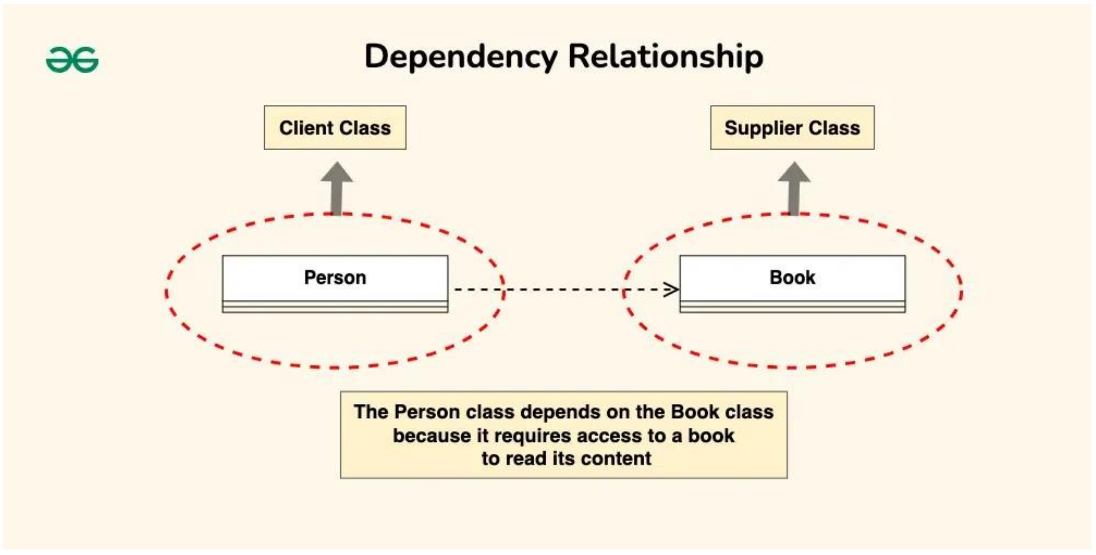
Generalisation (Inheritance)

Inheritance represents an “is-a” relationship between classes, where one class (the subclass or child) inherits the properties and behaviors of another class (the superclass or parent). Inheritance is depicted by a solid line with a closed, hollow arrowhead pointing from the subclass to the superclass.

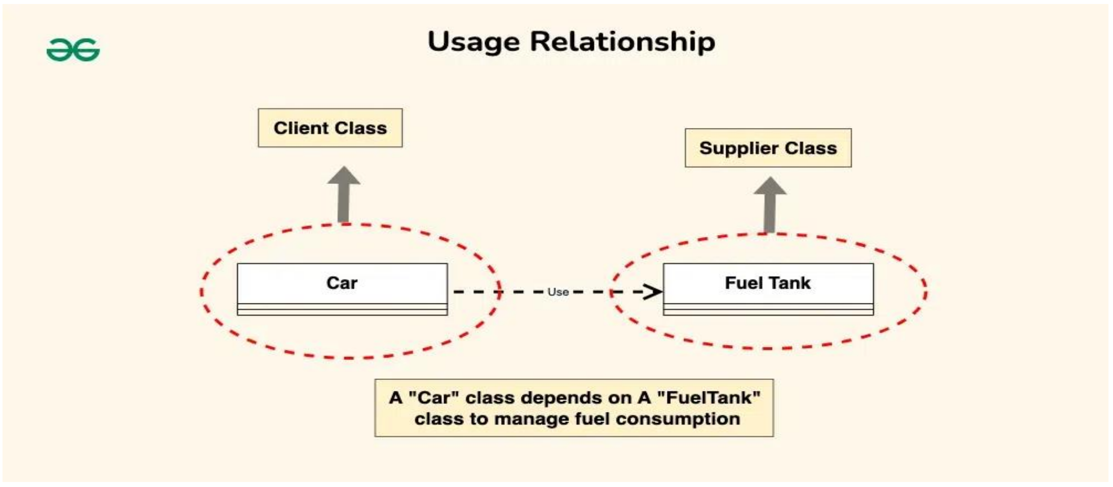
In the example of bank accounts, we can use generalization to represent different types of accounts such as current accounts, savings accounts, and credit accounts.


Generalization Relationship
Generalized Class
Bank Account
Sub Classes
Current Account
Saving Account
Credit Account
Current accounts, savings accounts, and credit accounts specialize from the account superclass, inheriting common banking functionalities

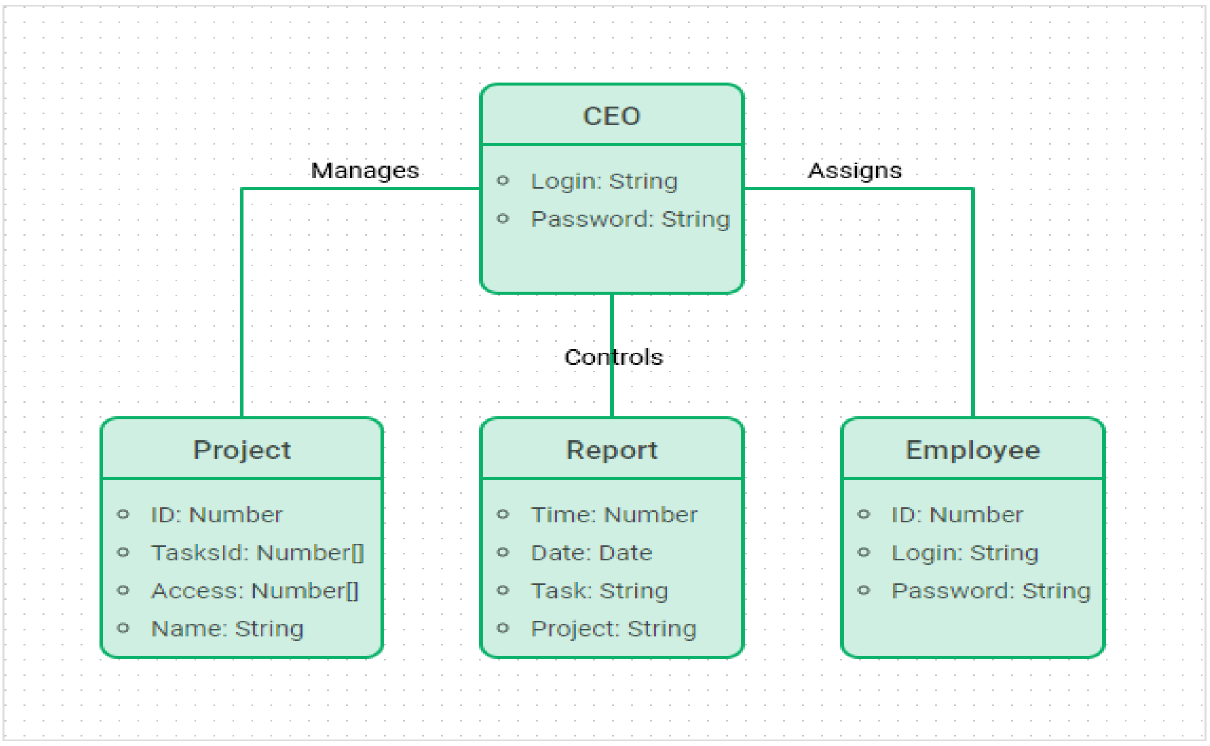
1. Dependency Relationship


Dependency Relationship
Client Class
Supplier Class
Person
Book
The Person class depends on the Book class because it requires access to a book to read its content

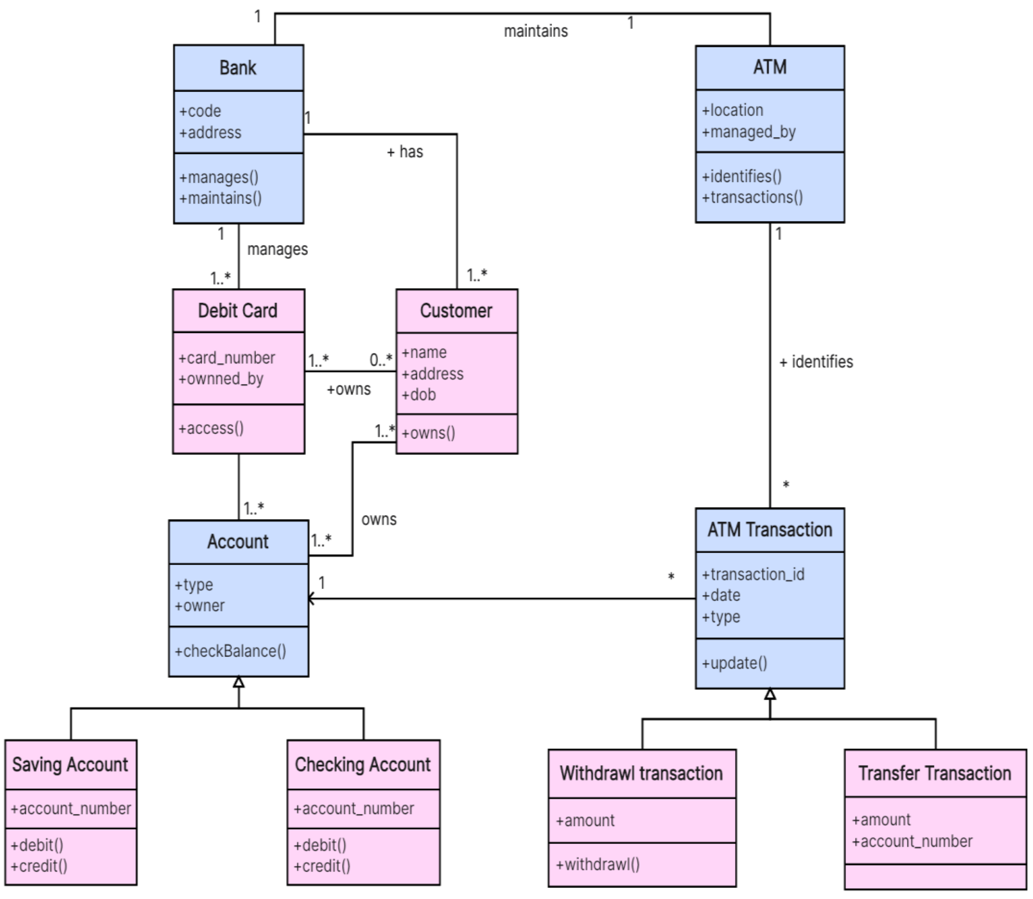
2. Usage Relationship



Usage Relationship
Client Class
Supplier Class
Car
Use
Fuel Tank
A "Car" class depends on A "FuelTank" class to manage fuel consumption



ATM System



Bank
+code
+address
+manages()
+maintains()
ATM
+location
+managed_by
+identifies()
+transactions()
maintains
1
1
+ has
1
1..*
manages
1
1..*
+ identifies
1
*
Debit Card
+card_number
+ownned_by
+access()
Customer
+name
+address
+dob
+owns()
1..*
0..*
+owns
1..*
owns
1..*
1..*
Account
+type
+owner
+checkBalance()
1
*
ATM Transaction
+transaction_id
+date
+type
+update()
Saving Account
+account_number
+debit()
+credit()
Checking Account
+account_number
+debit()
+credit()
Withdrawl transaction
+amount
+withdrawl()
Transfer Transaction
+amount
+account_number