Class Diagram

A UML Class Diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them. It helps everyone involved in a project—like developers and designers—understand how the system is organized and how its components interact.

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.

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.

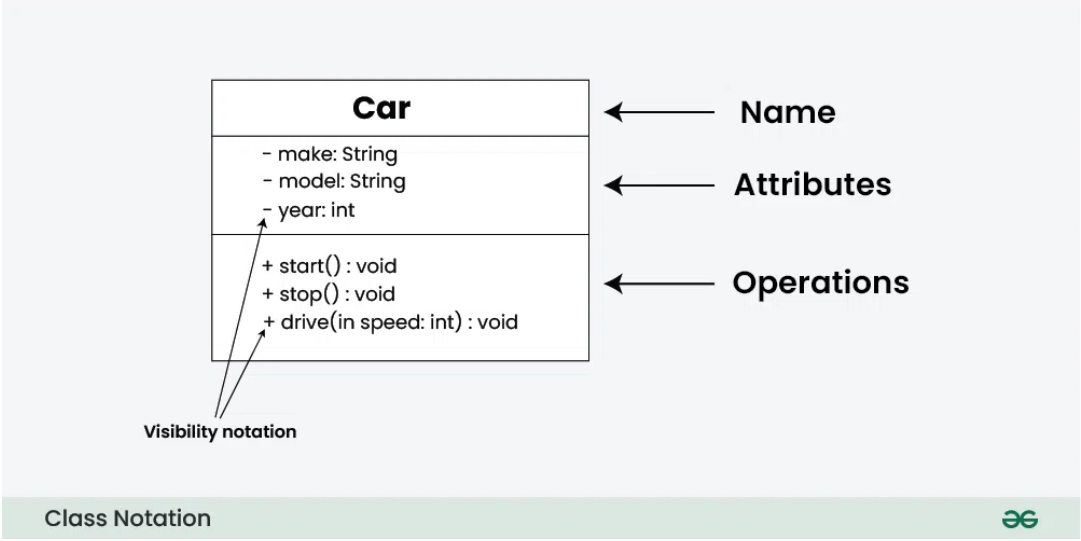
UML Class Notation

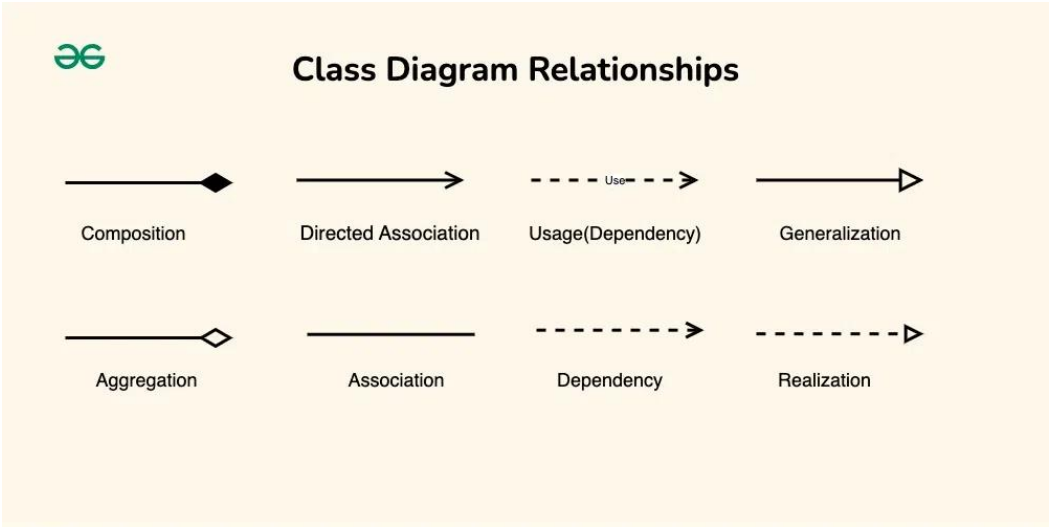
class notation is a graphical representation used to depict classes and their relationships in object-oriented modeling.

1. **Class Name:**
    - The name of the class is typically written in the top compartment of the class box and is centered and bold.
2. **Attributes:**
    - Attributes, also known as properties or fields, represent the data members of the class. They are listed in the second compartment of the class box and often include the visibility (e.g., public, private) and the data type of each attribute.
3. **Methods:**
    - Methods, also known as functions or operations, represent the behavior or functionality of the class. They are listed in the third

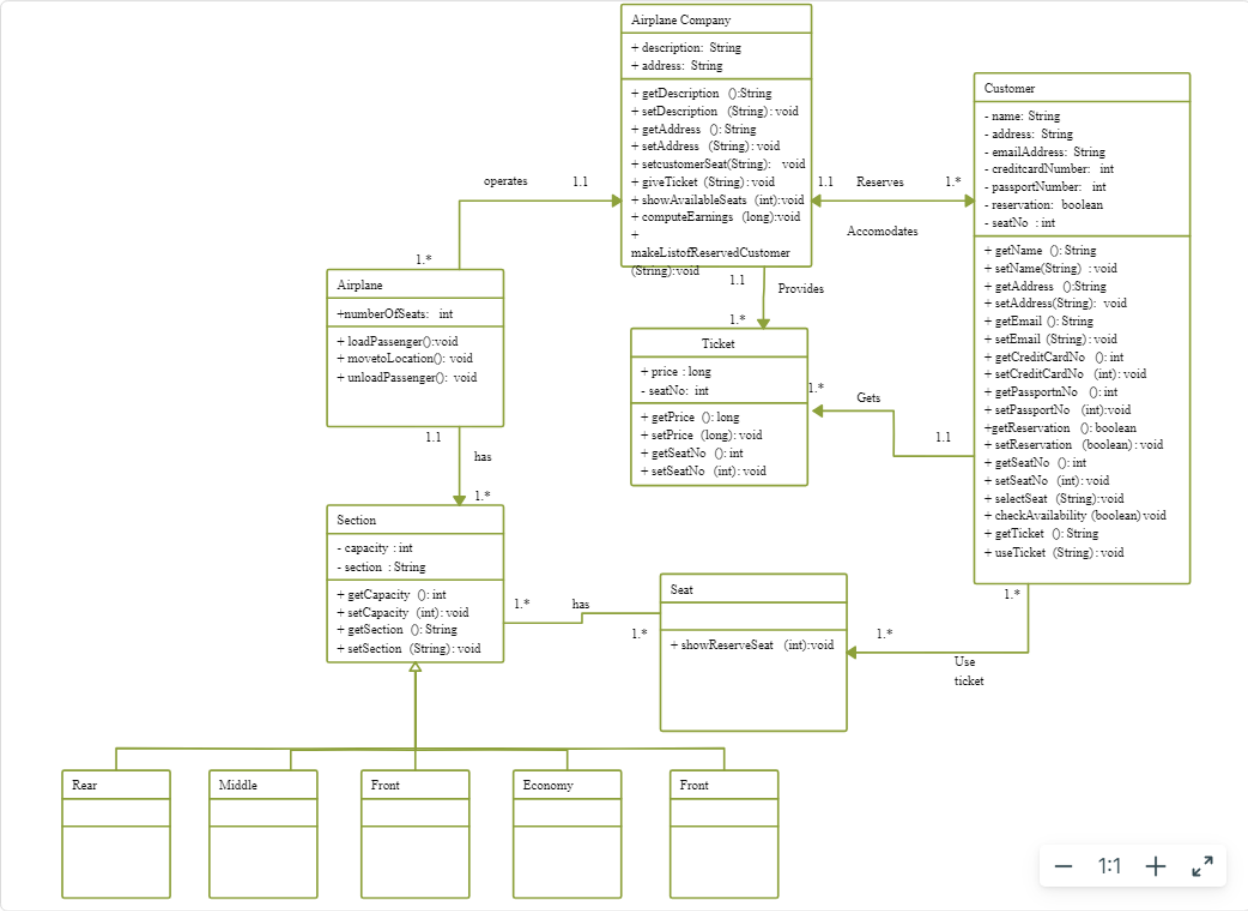
compartment of the class box and include the visibility (e.g., public, private), return type, and parameters of each method.

4. **Visibility Notation:**
    - Visibility notations indicate the access level of attributes and methods. Common visibility notations include:
        - + for public (visible to all classes)
        - - for private (visible only within the class)
        - # for protected (visible to subclasses)
        - ~ for package or default visibility (visible to classes in the same package)





Example


Airplane Company
+ description: String
+ address: String
+ getDescription ():String
+ setDescription (String): void
+ getAddress (): String
+ setAddress (String): void
+ setcustomerSeat(String): void
+ giveTicket (String): void
+ showAvailableSeats (int):void
+ computeEarnings (long):void
+ makeListofReservedCustomer (String):void
Customer
- name: String
- address: String
- emailAddress: String
- creditcardNumber: int
- passportNumber: int
- reservation: boolean
- seatNo : int
+ getName (): String
+ setName(String) : void
+ getAddress ():String
+ setAddress(String): void
+ getEmail (): String
+ setEmail (String): void
+ getCreditCardNo (): int
+ setCreditCardNo (int): void
+ getPassportNo (): int
+ setPassportNo (int):void
+getReservation (): boolean
+ setReservation (boolean): void
+ getSeatNo (): int
+ setSeatNo (int): void
+ selectSeat (String):void
+ checkAvailability (boolean) void
+ getTicket (): String
+ useTicket (String): void
operates
1.1
1.1
Reserves
1.*
Accomodates
1.*
Airplane
+numberOfSeats: int
+ loadPassenger():void
+ movetoLocation(): void
+ unloadPassenger(): void
1.1
Provides
1.*
Ticket
+ price : long
- seatNo: int
+ getPrice (): long
+ setPrice (long): void
+ getSeatNo (): int
+ setSeatNo (int): void
1.*
Gets
1.1
1.1
has
1.*
Section
- capacity : int
- section : String
+ getCapacity (): int
+ setCapacity (int): void
+ getSection (): String
+ setSection (String): void
1.*
has
1.*
Seat
+ showReserveSeat (int):void
1.*
1.*
Use
ticket
Rear
Middle
Front
Economy
Front
1:1