1. Class Diagram

A **Class Diagram** is a static diagram in UML (Unified Modeling Language) that describes the structure of a system by showing its classes, their attributes, methods, and relationships between the classes. It provides a blueprint for the system's object-oriented design, allowing developers to understand how the system is organized and how the objects interact.

Key Components of a Class Diagram:

- **Class**: Represented by a rectangle divided into three sections: the name of the class, its attributes, and its methods (or operations).
- **Attributes**: Variables that define the state of the object.
- **Methods**: Functions or operations that define the behavior of the class.
- **Relationships**:
    - **Association**: A simple line between two classes indicating that they are related.
    - **Aggregation**: A special type of association that represents a "whole-part" relationship, where the part can exist independently of the whole.
    - **Composition**: A stronger form of aggregation where the part cannot exist without the whole.
    - **Inheritance**: A line with a triangle arrowhead showing that one class is a subclass of another (i.e., it inherits attributes and behaviors).
    - **Interface realization**: A dashed line with a triangle showing that a class implements an interface.

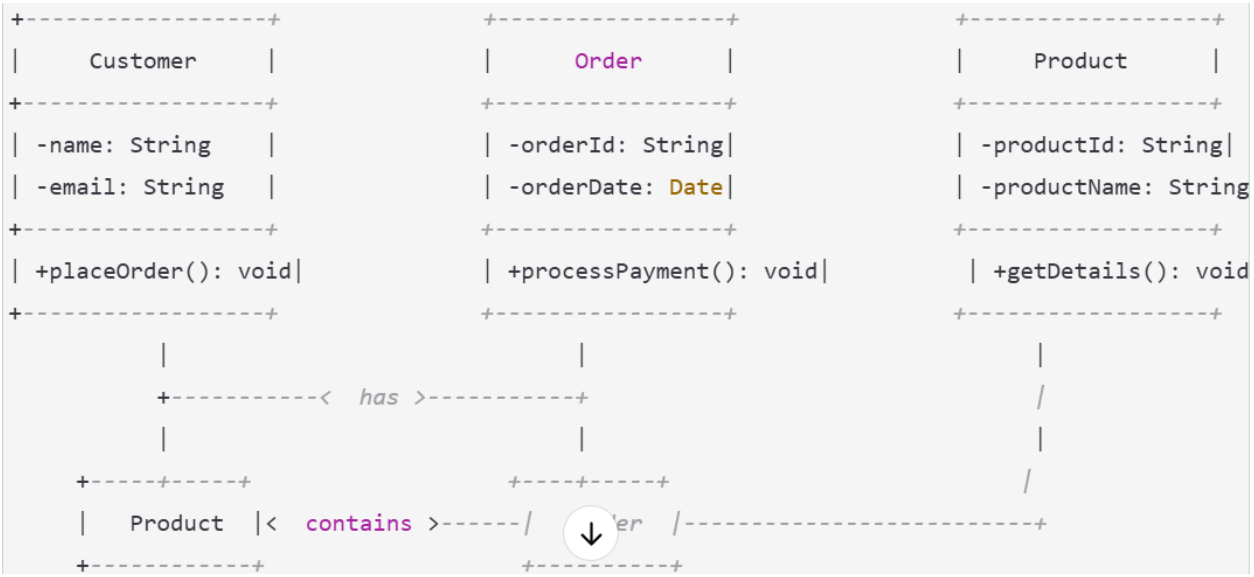
Example:

Consider a simple example of an **Online Shopping System**:

- **Customer** class has attributes like `name`, `email`, and methods like `placeOrder()`.
- **Order** class has attributes like `orderId`, `orderDate`, and methods like `processPayment()`.
- **Product** class has attributes like `productId`, `productName`, and methods like `getDetails()`.

Class Diagram for the example:

In this diagram:



- The `Customer` class can place an order (indicated by the association between `Customer` and `Order`).
- The `Order` class contains multiple `Product` objects.

2. Sequence Diagram

A **Sequence Diagram** is a type of interaction diagram that shows how objects interact in a particular scenario of a use case. It highlights the order of method calls between objects and the flow of messages in the system.

Key Components of a Sequence Diagram:

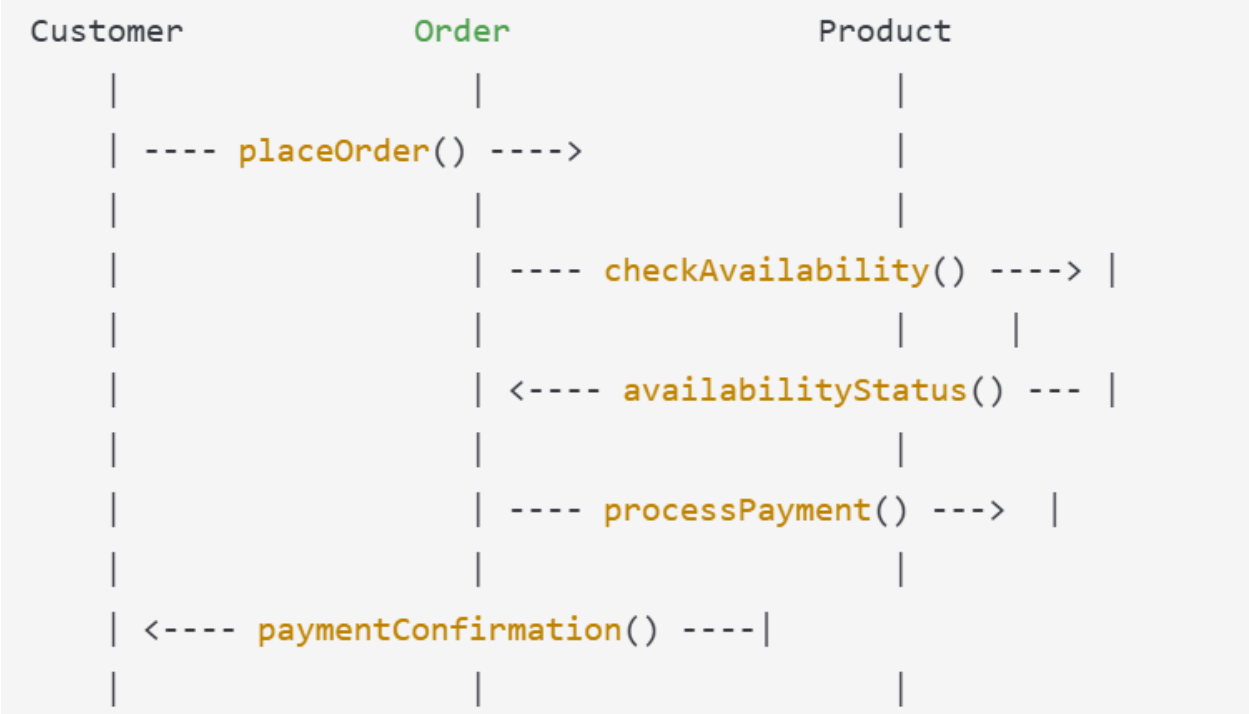
- **Objects**: Represented by boxes, typically at the top of the diagram. These are the participants in the interaction.
- **Lifelines**: Vertical dashed lines that represent the object's existence over time.
- **Messages**: Horizontal arrows between objects showing the interactions (e.g., method calls).
- **Activation bars**: Rectangular boxes on a lifeline that show when an object is active during a message call.
- **Return messages**: Dashed arrows showing the return of control or values after method execution.

Example:

Consider the scenario where a **Customer** places an order for a **Product** in the system:

1. The `Customer` places an order by calling `placeOrder()` on the `Order` object.
2. The `Order` object checks product availability by calling `checkAvailability()` on the `Product` object.
3. If available, the `Order` object processes the payment by calling `processPayment()`.

Here is a simple Sequence Diagram for this flow:



In this diagram:

- The `Customer` initiates the process by calling `placeOrder()`.
- The `Order` calls `checkAvailability()` on the `Product` object to check if the product is in stock.
- Finally, the `Order` processes the payment by calling `processPayment()`.

Summary:

- A **Class Diagram** is used to represent the static structure of a system by showing the classes and their relationships.
- A **Sequence Diagram** illustrates how objects interact dynamically in a system, showing the sequence of method calls and their interactions over time.

Both diagrams are crucial in understanding and designing object-oriented systems, each focusing on different aspects of the system's structure and behavior.