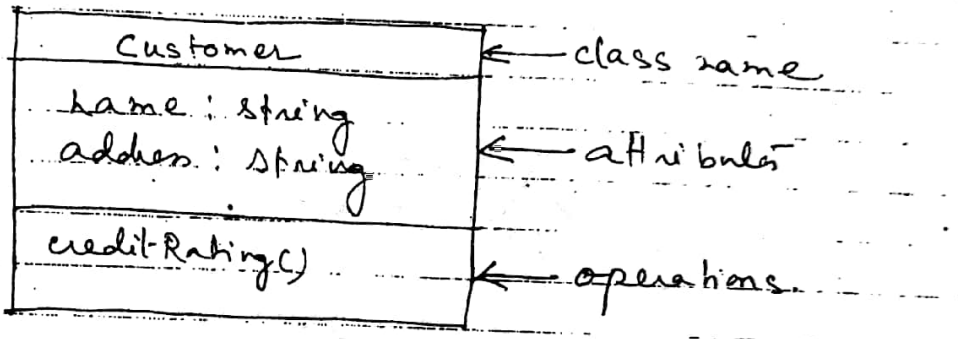
Conceptual Data Modelling using UML (Unified modelling language) Class Diagram

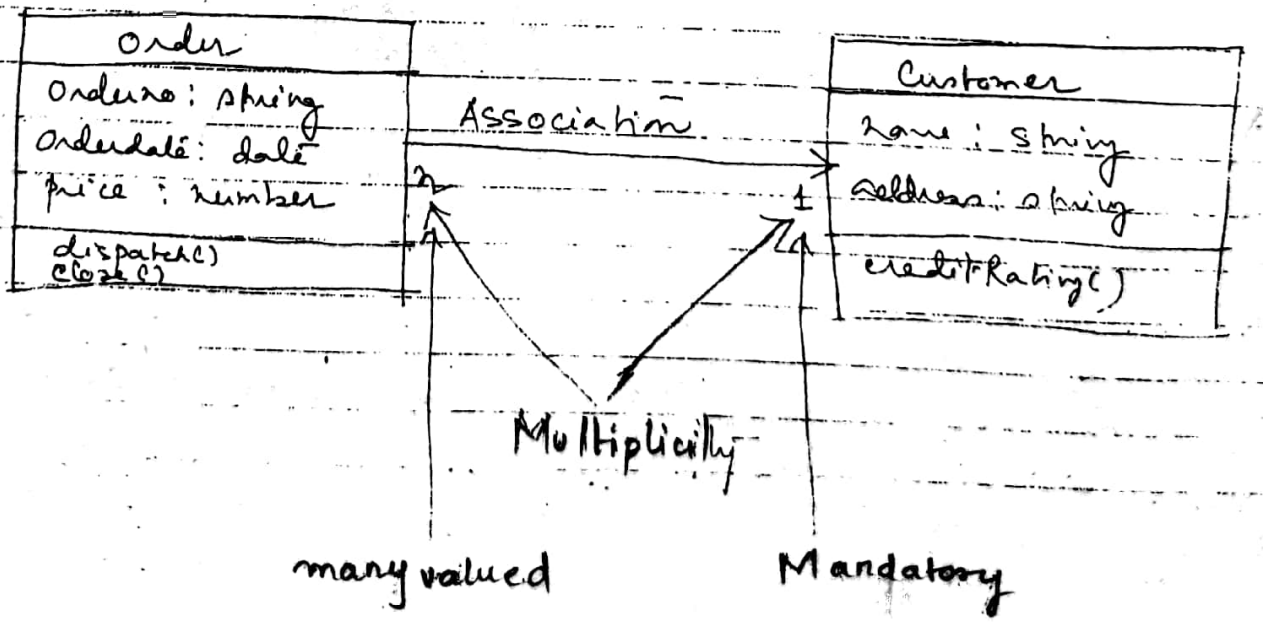
Class diagrams are widely used to describe the types of objects in a system and their relationship. Class diagrams show class structure and contents using design elements like classes, packages & objects.

Class diagrams describes three perspectives when designing a system and they are conceptual, specification and implementation. These perspectives become evident as the diagram is created and help to finalize the design.

Classes are composed of three elements: a class name, attribute and operations. Below is an example of a class



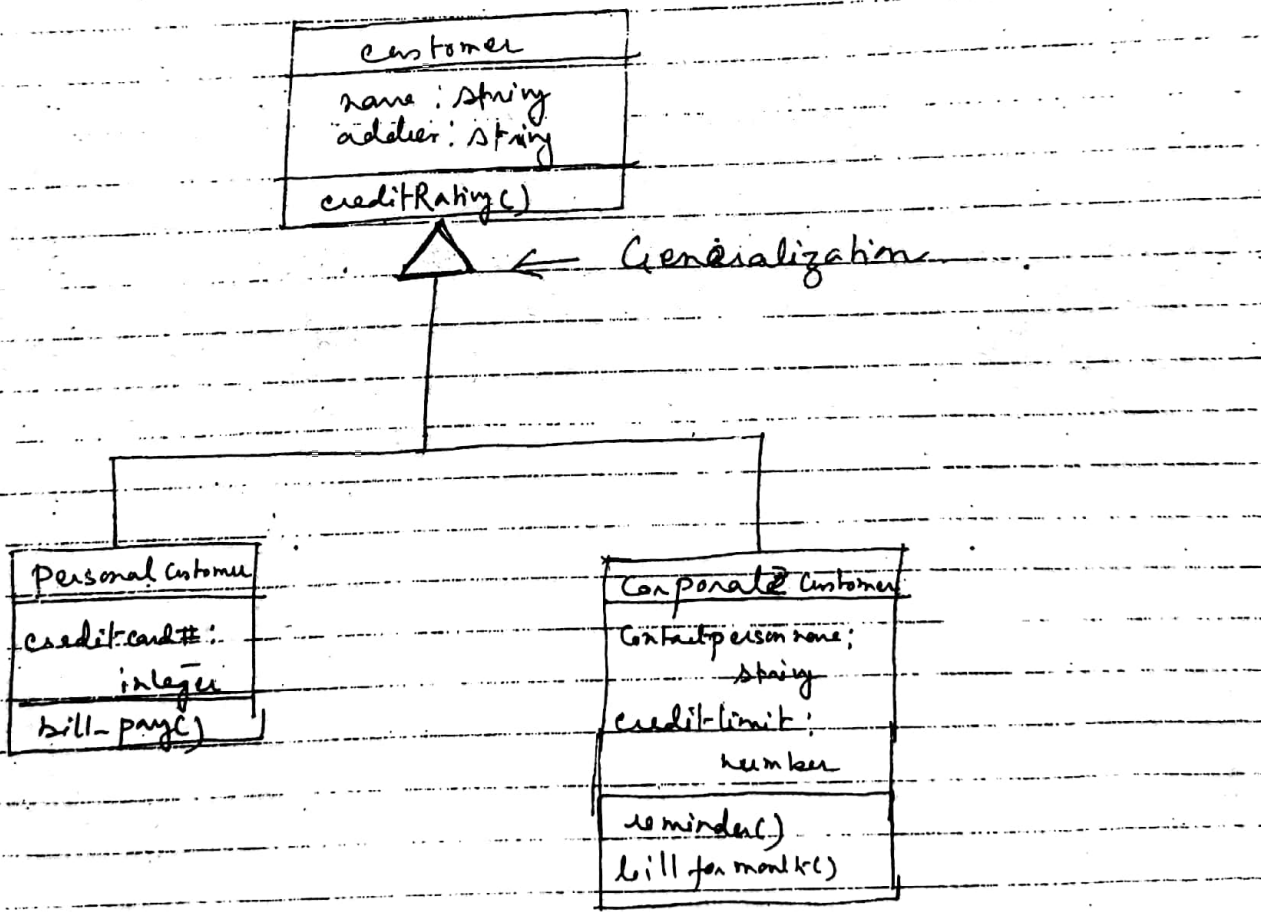
Class diagram also display relationships such as containment, inheritance, associations. Below is an example of associative relationship



The association relationship is the most common relationship in class diagram. The association shows the relationship between instances of classes. For the above diagram, the class order is associated with the class customer. The multiplicity of the association denotes the no. of objects that can participate in the relationship. For example, an order object can be associated to only one customer object, but a customer object can be associated with many order objects.

Another common relationship in class diagram is a generalization. A generalization is used when two classes are similar but have some differences.

For example:



In this example the classes personal customer & corporate customer have some similarities such as

name & address, but each class has some of its own attributes & operations. The class customer is a general form of both personal & Corporate customer classes. This allows the designer to focus the customer class for modules and do not require in-depth representation of each type of customer.

Class diagrams are used in nearly all Object oriented software design. When designing classes we should consider what attributes & operations it will have. Then we determine how instances of these classes interact with each other. Using these basic technique one can develop a complete view of the system software.

For example: