2023.05.19

> classes and class diagrams

- There are several components like people, locations, instances in a real life scenario.
- A category of componants we are declare in our system is an object.
- people, location, physical objects, events likewise we can identyfy 8 different elemant / component category that needs to include in a system.
- we need to identify objects before design a system.
- After that we can idetify the classes. these objects can be categorize into.
- we categorize the objects into classes based on their characteristics, features and similarities in the system.

- Class :- A category or classification of a set of objects.
- object :- The components we categorize in to classes.
- Domain class :- classes that describe the objects from the problem domain.
- class diagram :- A diagram consisting of classes and Associations among the classes.

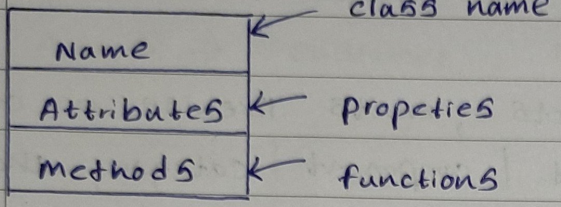
ex:- objects you can see in a hospital management system.

1) List of the objects,
   i) people → patients, doctors, nurses
   ii) Things → medicine, Surgical equipment
   iii) Events → Surgery, Scan, checkups
   iv) locations → wards, theaters.

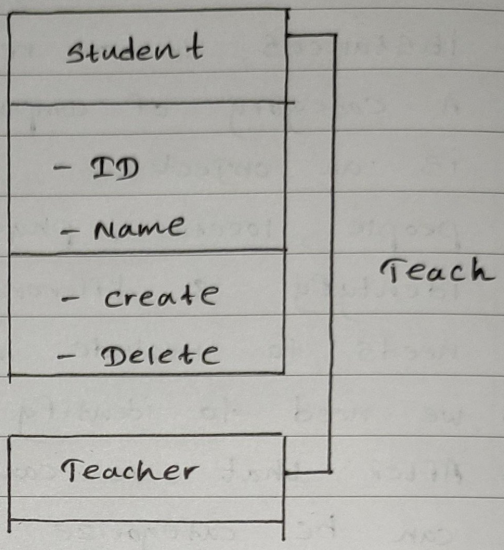
➤ Class diagram.

- In class diagram you can use several notations,

1) class Symbol



ex:-



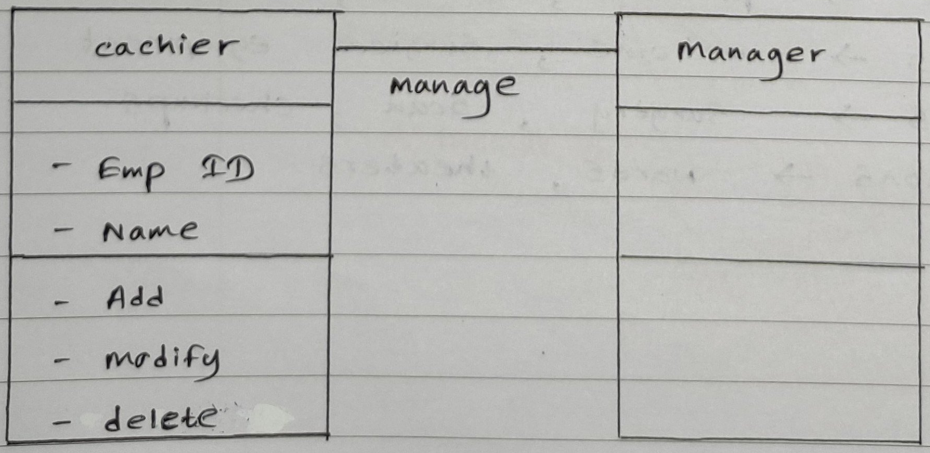
- purpose of class diagram.

- Analysis and design of the static view of an application.
- Describes reponsibilities of a system.
- Base for component and deployment diagram.
- Forward & Reverse engineering.

➤ Relationship types. in class diagram.

1) Association

- Represent the relationship between 2 classes. (Link 2 classe)
- represent by a solid line. (—)

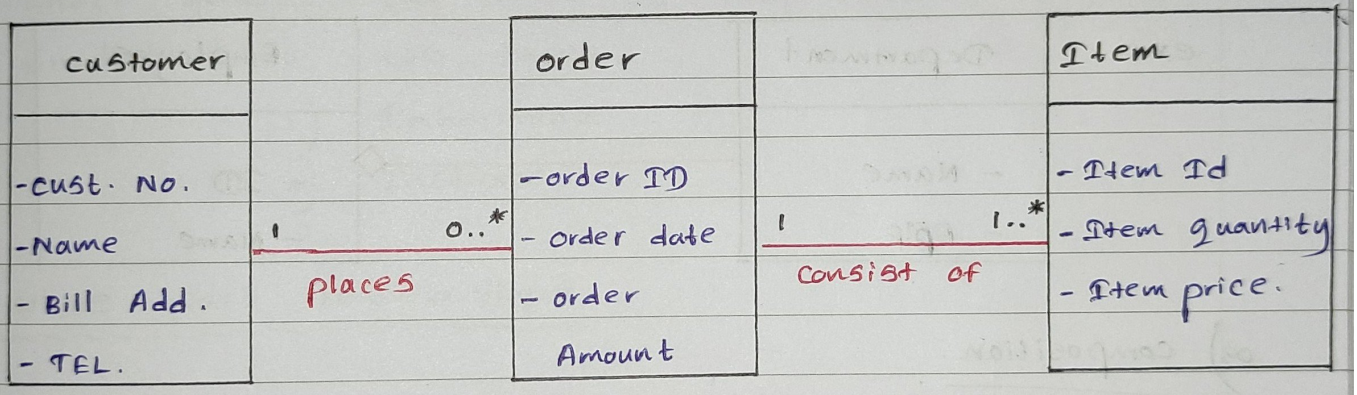


- class diagram are similar to ER Diagrams as they show their attributes and functionalities.
- you have to show the number instances porticipating in the relationship (like relationship schema in ER) in class diagram. and it's called 'multiplicity'.
- Just like in ER diagram 1:1 and 1:M we need to show multiplicity using numbers.

- multiplicity -

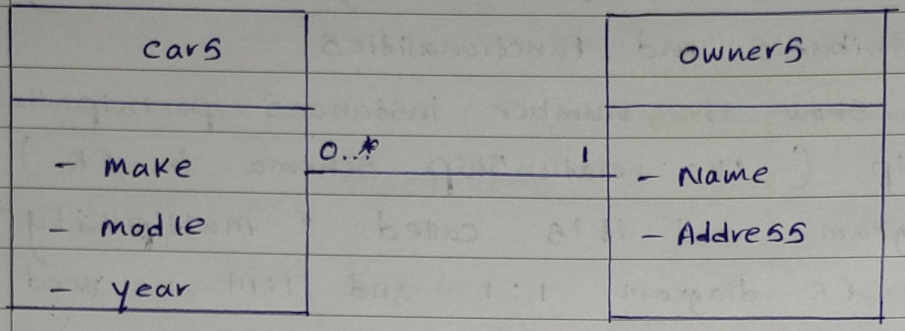
i) 1 - one & only one
ii) 0..* - zero and more
iii) 1..* - one or more

ex:-



- The above diagram notation shows that, each customer can place many orders and that each order is placed by one customer.
- A customer can exist in the system without placing an order.
- A customer can place one or many orders.
- one order can have 1 or more order items.
- An order must have an order item.
- The '*' mark next to the order indicates many orders.
- The second Association shows that one order consist of one or more Items and each item Associated with one order.

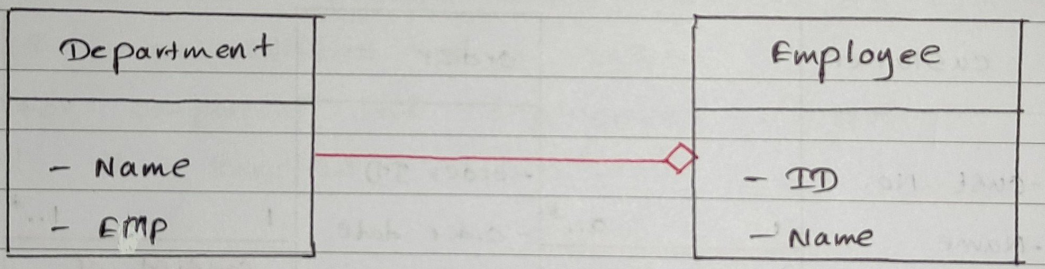
Exercise.



2) Aggregation.

- Represents a relationship where one class contains or is composed of other classes.
- It implies that the parts can exists independantly of the whole.

ex:-



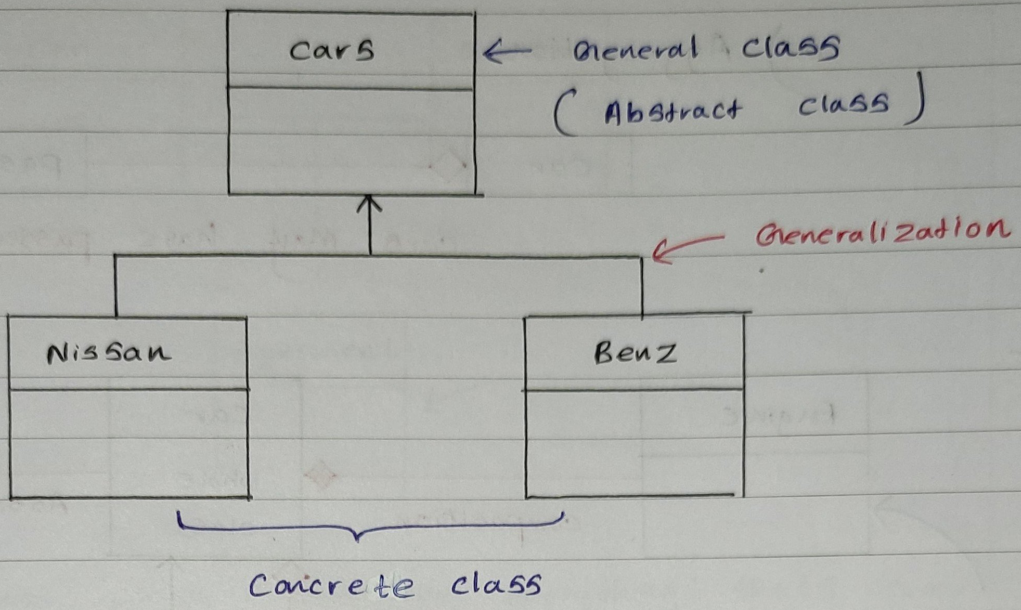
03) Composition.

- A whole part relationship where the parts cannot exist without the whole.

04) Generalization.

- The specialization between two or more classes can be shown by the generalization relationship.
- This shows the inheritance. Therefore the attribute of the parent class is common to the child class.

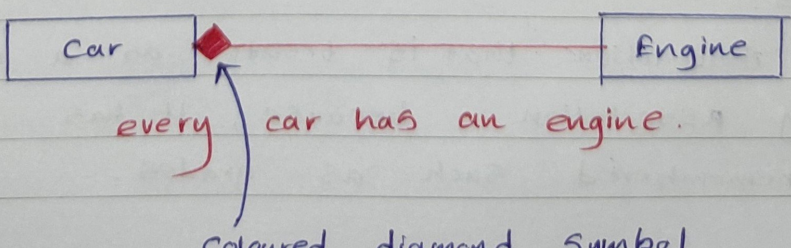
ex:-



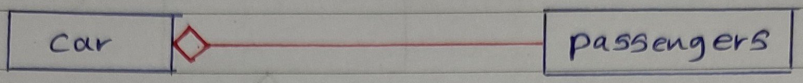
- As cars are divided into 2 groups (types) there are no categories or any instances or any real object instances under general car class as they have already been divided into two types.
  1) Inheritance
  2) Instances

> whole - part relationships

- A relationship between classes in which, one class is a part of another class.
- occurs between two classes.
- If a class in the system becomes a part of another class in the same system then this can represent as whole part relationship between two classes.
- There are 2 types of relationships,
  1) composition

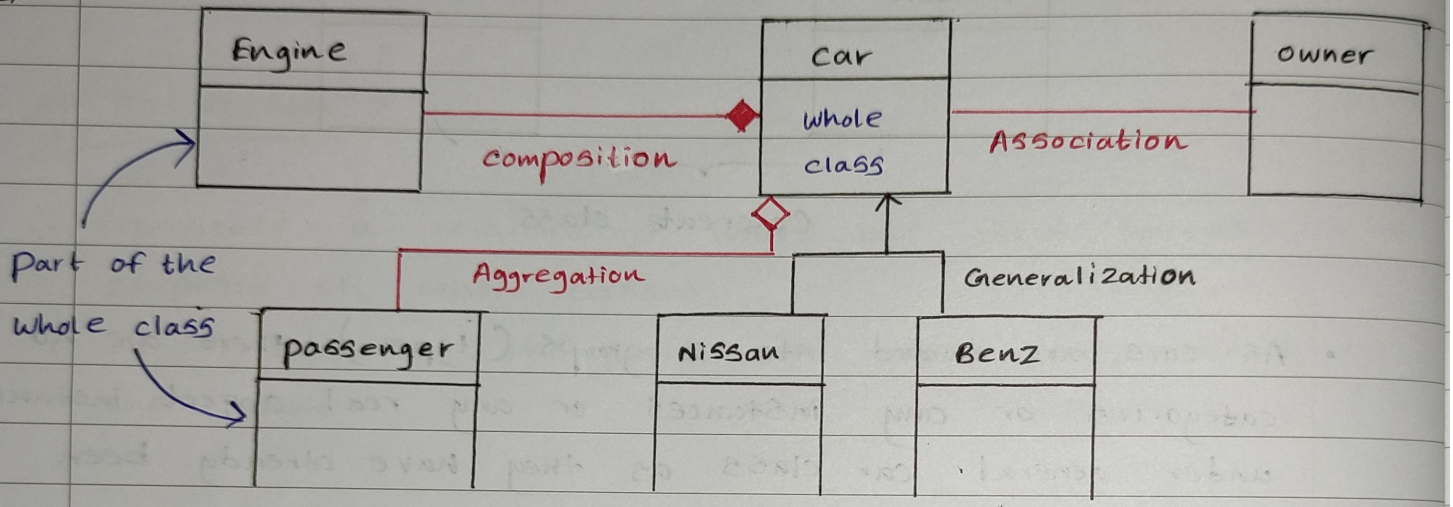


02) Aggregation



cars may have passengers

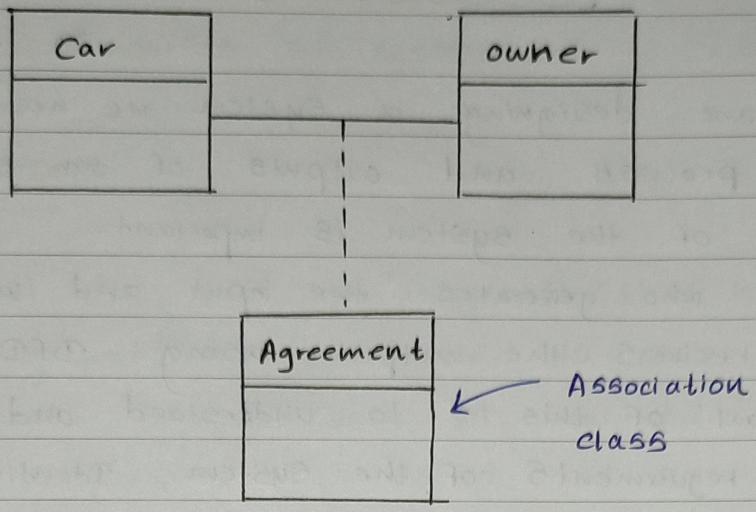
ex:-



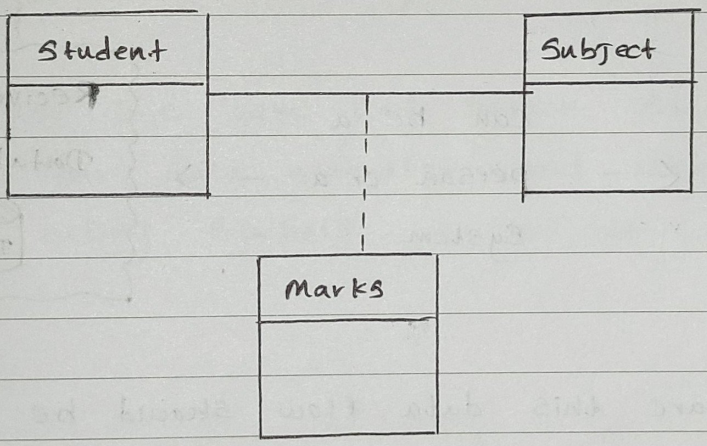
- when considering a ride passenger becomes a part of car.
- without engine car won't be whole but even without passenger car can exists as a whole.
- without a car keeping an engine is useless and such a relationship is called ' composition '.
- Even when we remove the whole class 'car' we can still keep passengers and such relation called ' Aggregation'.

> Association class.

- Some attributes or methods may occur due to the association.
- Does not fully belong to either one of the associated class.
- An Association that is treated as a class in a many to many association. because, it has attributes that need to remembered such as grades.



* The agreement class shows the everything thats common to the two classes car and owner and belong to both two classes.



* These classes can't be identified at first but once the association class marked they can be idetified.
* The 'Marks' class attributes are bathe belong to the 'Student' and 'Subject' classes.

* Reflexive association occurs when a class may have multiple functions or responsibilities.

exercise

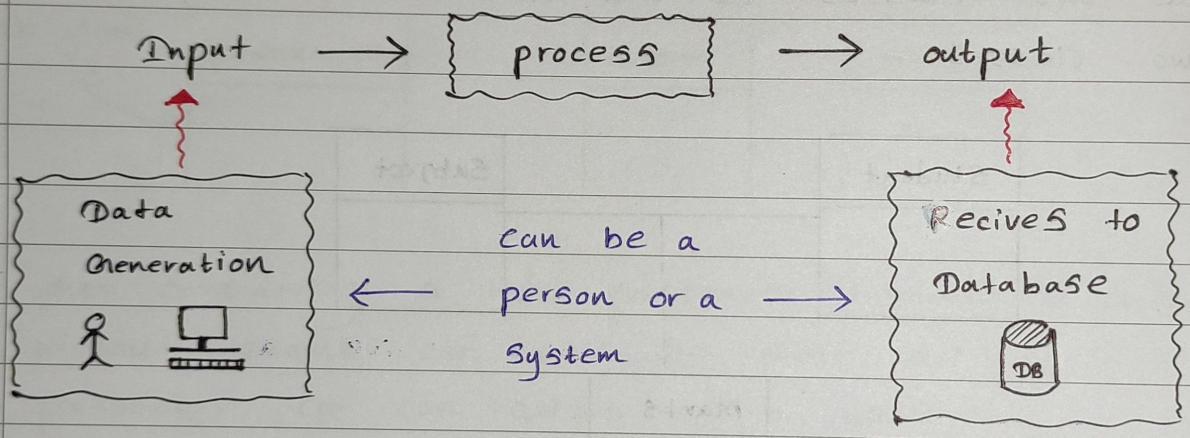
- watch slides.

2023.0?.01

NSBM GREEN UNIVERSITY TOWN

> process modeling using DFD

- When we are designing a system we need to identify inputs, process and outputs of our system.
- Data flow of the system is important.
- we discuss who generates the input and who or what systems recives the output using DFD's.
- primary goal of this is to understand and analysis the functional requirments of the system, Identify the main processes and the data flows and define the boudaries of the system.



- In any software this data flow should be discussed.
- There are 2 types of DFD Diagrams.
  1) Logical DFD
  2) physical DFD

1) Logical DFD

- In logical DFD we show data flow in a conceptual way.
- who are the data generators, what are the processes, how the processes are working, what are the outputs and who recieves it in a logical manner.
- understand and analysis the functional requirments of the System, Identify main process and data flow and

define the boundaries of system.

2) physical DFD

- In physical dfd the infrastructure that needs to fascilitate this conceptual or logical dfd is also shown.
- This provides detailed view of how data moves between various components, processes and external entities.
- often used daring the implementation phase.

* There are two standerds to create a DfD,
  1) Gane and Sarson standerds
  2) Demarco and -youdan Standerd

- we will use the gane and Sarson Standerd.
- to show our data generator and recivers we can use external entity symbols. write entity name within it.

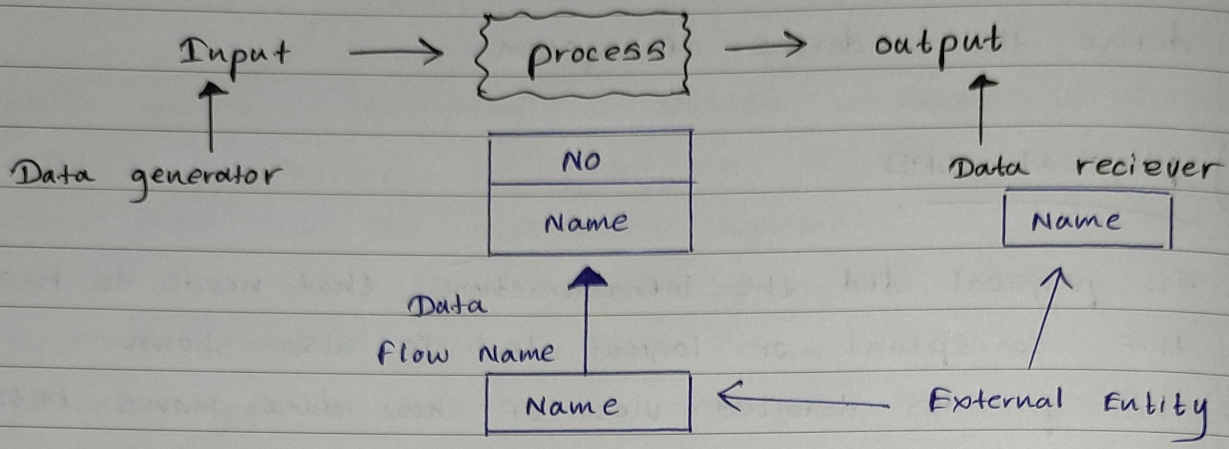
- Elements.

* process → | NO. | ← process number
  | Name | ← process name

* Data flow → Name →

* Data Store → | D1 | Name | * Databas shown here

* External Entity → | Name |



- Extenal entity symbol either shows a process or another system connected to the system.

DFD Levels

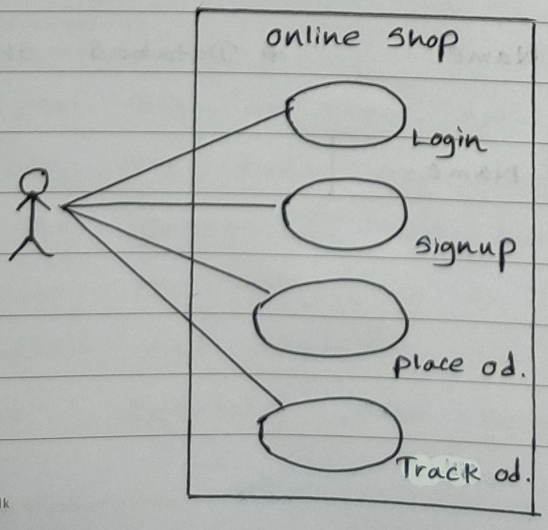
1) **context level DFD**

- provides an overview of the system interactions with external entity.

2) **Level 0 DFD**

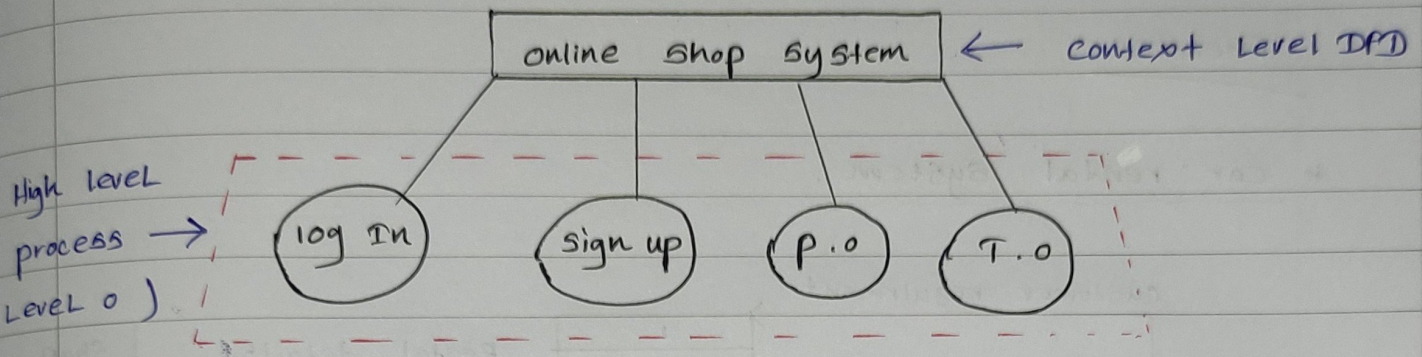
- provides a detailed overview of major processes and data flows within the system.

ex:- use case :



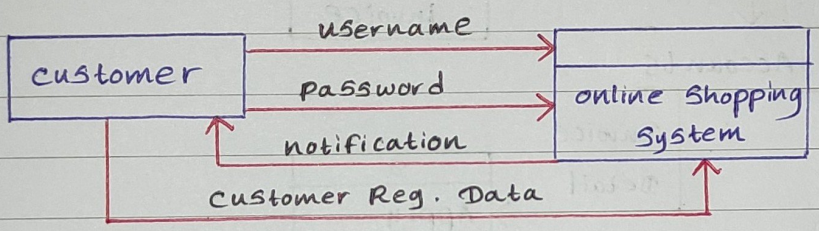
* There are 4 high-level process in the above online shopping system.

* use case diagram should match with the DFD.

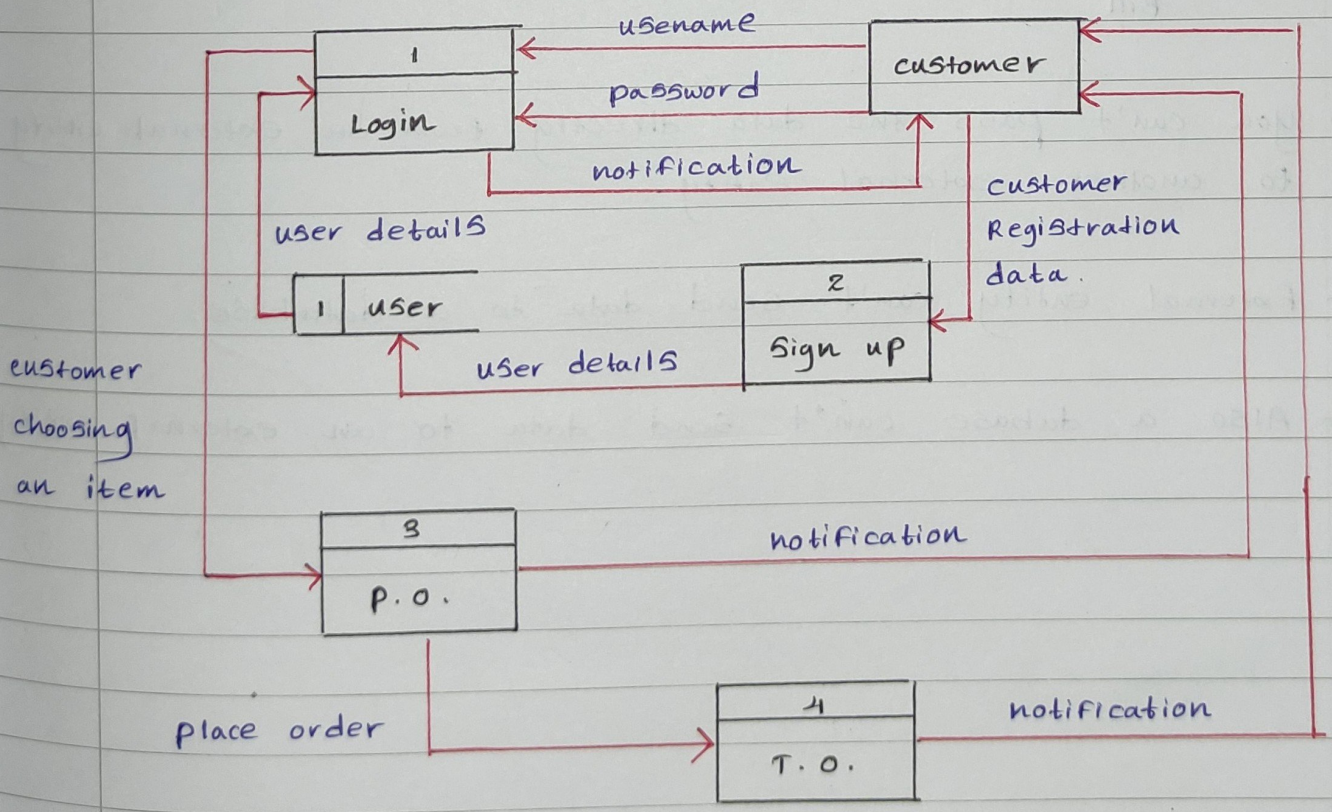


* Break down of high level processes is shown using Level 0 DFD.

- **context level DfD**



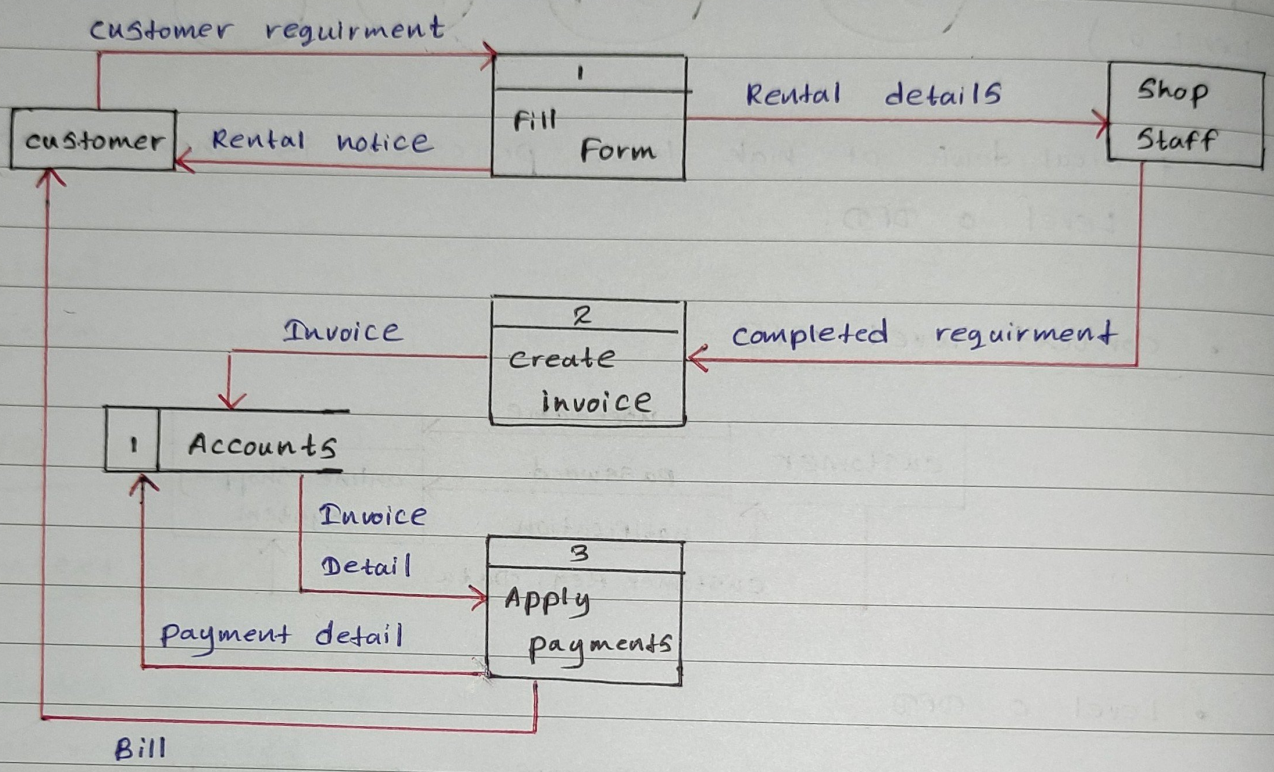
- **Level 0 DFD**



* can't have same data as both input and output in a process.

Exercise.

* car rental system.



* You can't pass the data directly from an external entity to another external entity.

* External entity can't send data to a database.

* Also a dabase can't send data to an external entity.