SYSTEM DESIGN BASICS

PLACEMENT PREPARATION

[EXCLUSIVE NOTES]

[SAVE AND SHARE]

Curated By-

HIMANSHU KUMAR(LINKEDIN)

https://www.linkedin.com/in/himanshukumarmahuri

TOPICS COVERED-

PART-1 :-

- Introduction to Objects
- Classes & Objects
- Virtual Bank - Classes & Objects
- Problem Solving Process
- Software Development Process
- Why UML?
- UML Diagrams

CHECKOUT AND DOWNLOAD MY ALL NOTES

***LINK*- https://linktr.ee/exclusive_notes**

Introduction to Objects

What is an Object?

Before diving into object-oriented Analysis, we should first discuss what actually an object is?
In this section I will start with the basic definition of an object, then I will refine this definition with the help of examples.

Generally, an object is anything that occupies space in the universe.
For example: The room you are sitting in contains several objects. The laptop you are working on is an object, the table you have kept your laptop on is an object. The chair on which you are sitting is also an object.

Laptop → Object

Table → Object

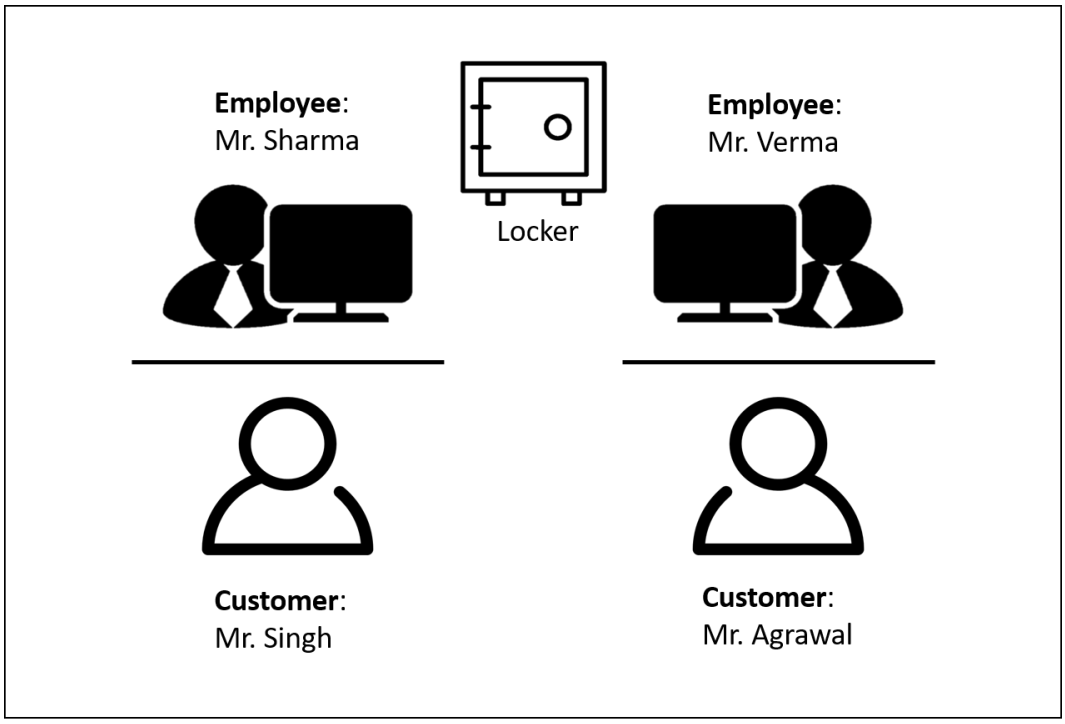
Chair → Object

And, interestingly you are also an example of an object.
We can say, any tangible item which occupies space in the universe is an object.

Now a question must come to your mind what about intangible items? Can we have intangible objects?

I will try to explain this with the help of an example:
You just imagine that you are visiting your nearby bank. You are standing at the door of the bank. What things do you observe?

Definitely, you can see bank employees interacting with bank customers and the locker that the bank uses to keep the money. Also, you can notice that bank employees are using their computers to update the bank accounts of the customers. This is how a real bank looks like:



XYZ Bank

Tangible Items:

Tangible Items

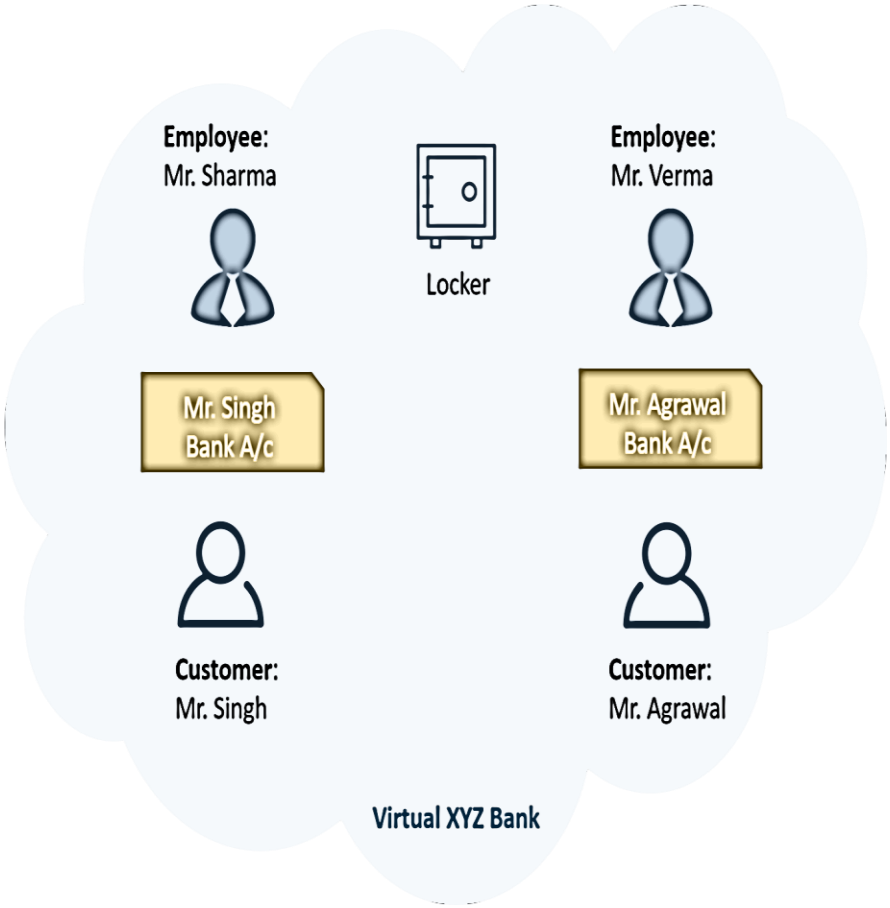
Now just give a thought, how can I make this bank work virtually or in other words I want to implement a virtual bank on my computer which can do everything that a real bank can do.
To achieve this, I first need to identify what all real-world entities need to be present in the virtual world.

Let's try to move things one by one.
Do we need the locker in our virtual bank? Yes, It is needed to keep track of the amount with the bank.

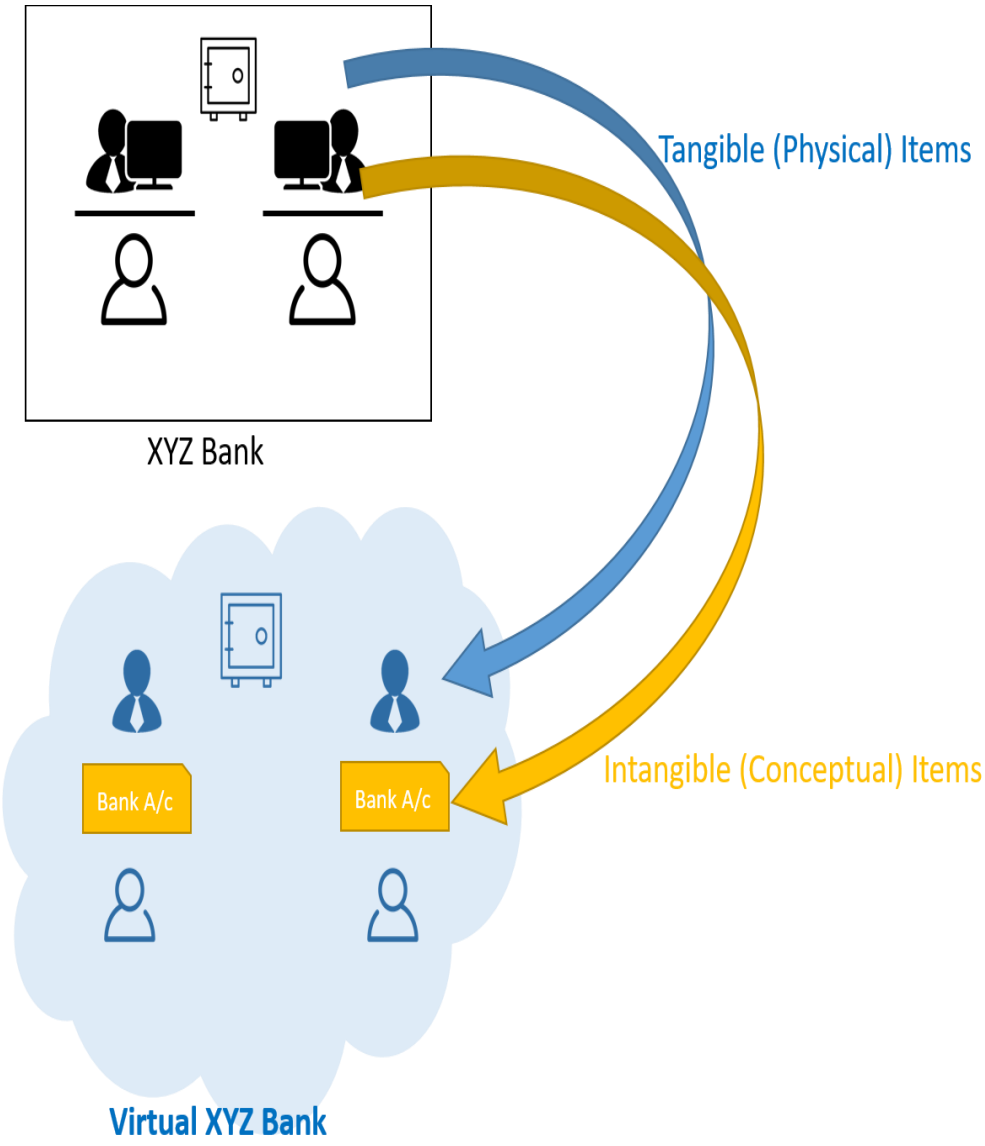
Do we need employees? Yes because the administration and management of banks can be handled by employees only.

Do we need customers? Yes because customers will interact with the bank accounts.

Do we need personal Computers? Just Try to recall that Employees were using personal computers to update the bank accounts of the customers. So we don't need personal computers in the virtual world. But we will definitely require the bank accounts of customers in our virtual world.



You can see that we have moved some physical and some conceptual items from the real-world to the virtual world to simulate the functionality of the real world.



Now we are at a point where we can give a more precise definition to the term object i.e.,

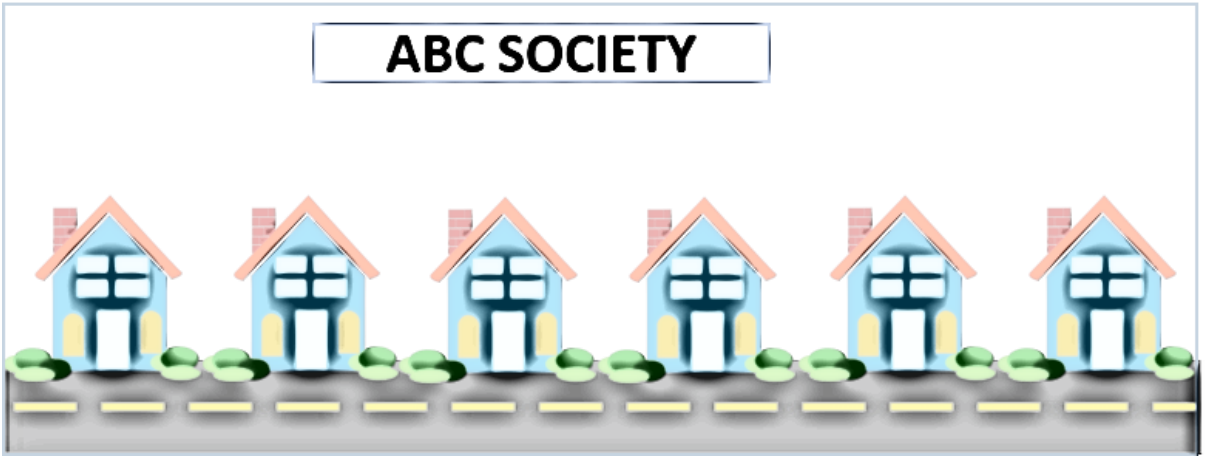
An Object is a real-world element in an object-oriented environment that may have physical or conceptual existence.

Classes & Objects

Classes & Objects

In this section, I will help you to understand the concept of class with the help of some examples. Then we will go back to the bank example discussed in the previous section to filter out classes and objects in the virtual bank.

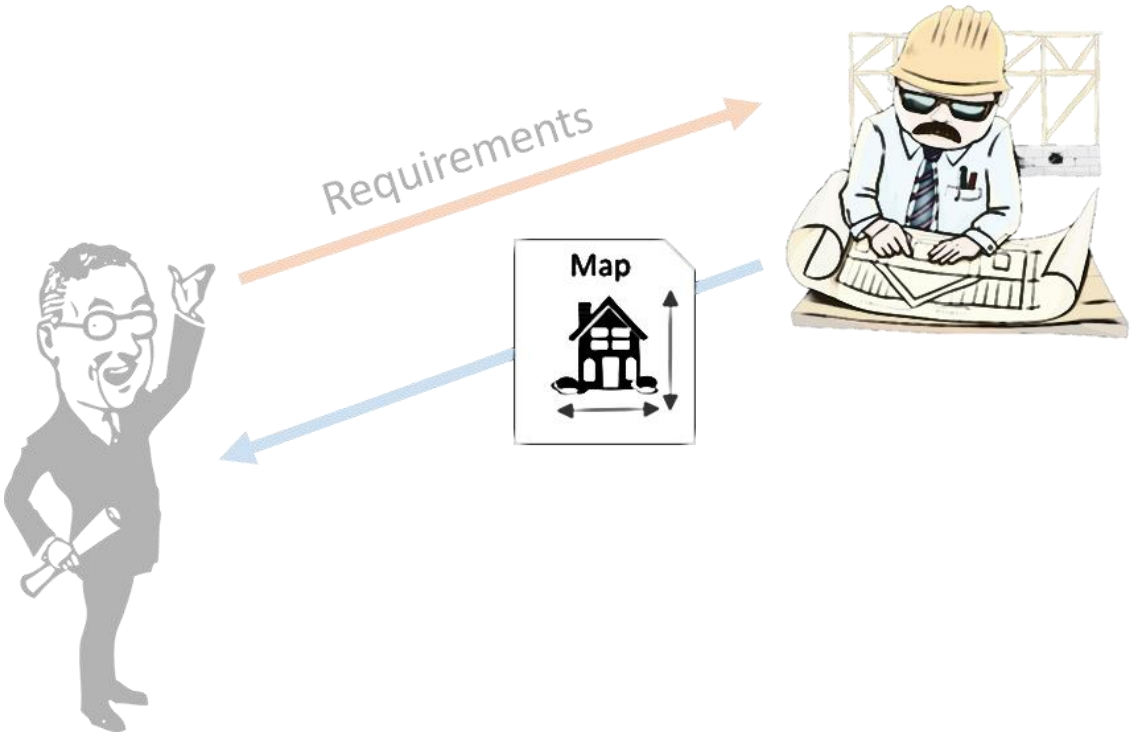
If you have got a chance to visit any of the societies, you might have noticed that all the houses have the same structure. Have you ever wondered that how is it possible to create several similar houses?



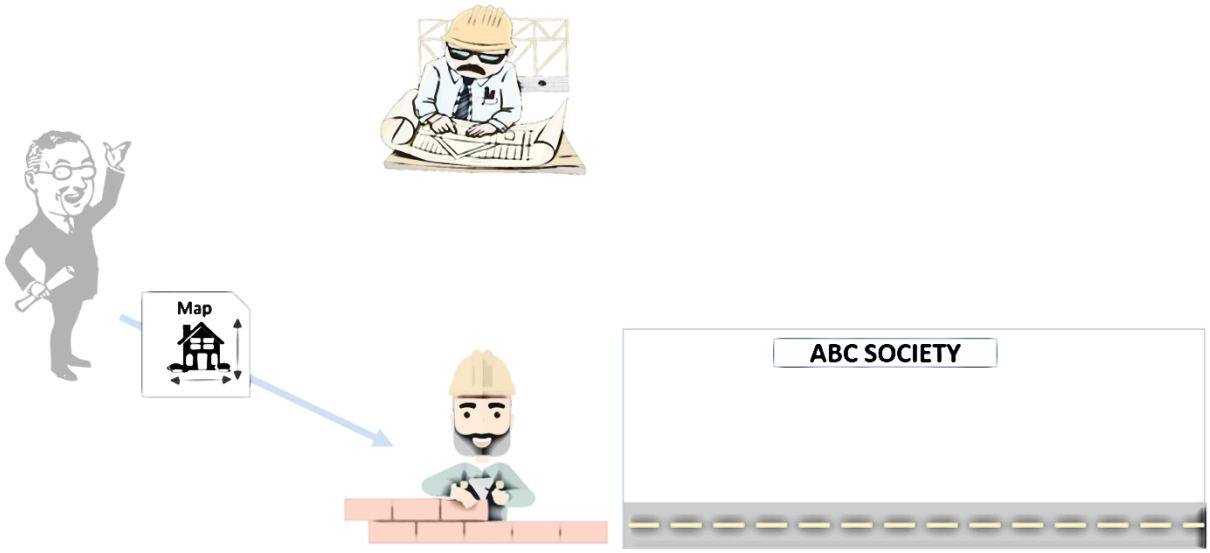
It is very simple:

- The builder of society goes to the architect with his requirements.
- On the basis of requirements, the architect creates a map containing all the details of the house.
- Then, the builder hands-over this map to the contractor.

- The contractor follows the details in the map while building each and every house; hence he is able to build several similar houses.
- This map acts as a blueprint for the contractor.



The same process is followed by a car company manufacturing cars. The company shares the design of the car with the manufacturing unit. The manufacturing unit follows the design or blueprints to build several cars with a similar structure.



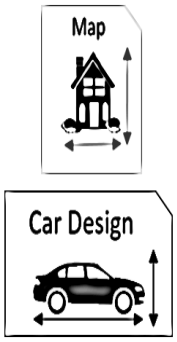
In both examples, we saw that blueprints contain all the information to build the house or to build the cars. But they themselves are neither the house nor the car.

We can easily see, that the contractor used the blueprint to build n number of houses.
Also, the manufacturing unit used the blueprint to build n number of cars.

Now let's place all the blueprints on my right and all the instances on my left.
We can say that all the instances occupy some space in-universe. Hence we can call them objects.

Whereas, the blueprints on my right contains all the information to build these objects. These blueprints are also known as CLASSES.

Blueprints
Or
Classes



Instances
or
Objects

Finally, A class can be **defined as a blueprint or description of objects that can be created from it**.
We have also observed the following properties of a class:

1. NOT REAL: Class is not a REAL THING; it's just a concept, actually the objects created from a class are real.
2. A single class can be used to create any number of instances or objects.

Virtual Bank - Classes & Objects

It's time to look back at the example of virtual banks and refine our concept of classes and objects.

Here it is,
Now we will perform a small activity to classify similar objects in our virtual bank.

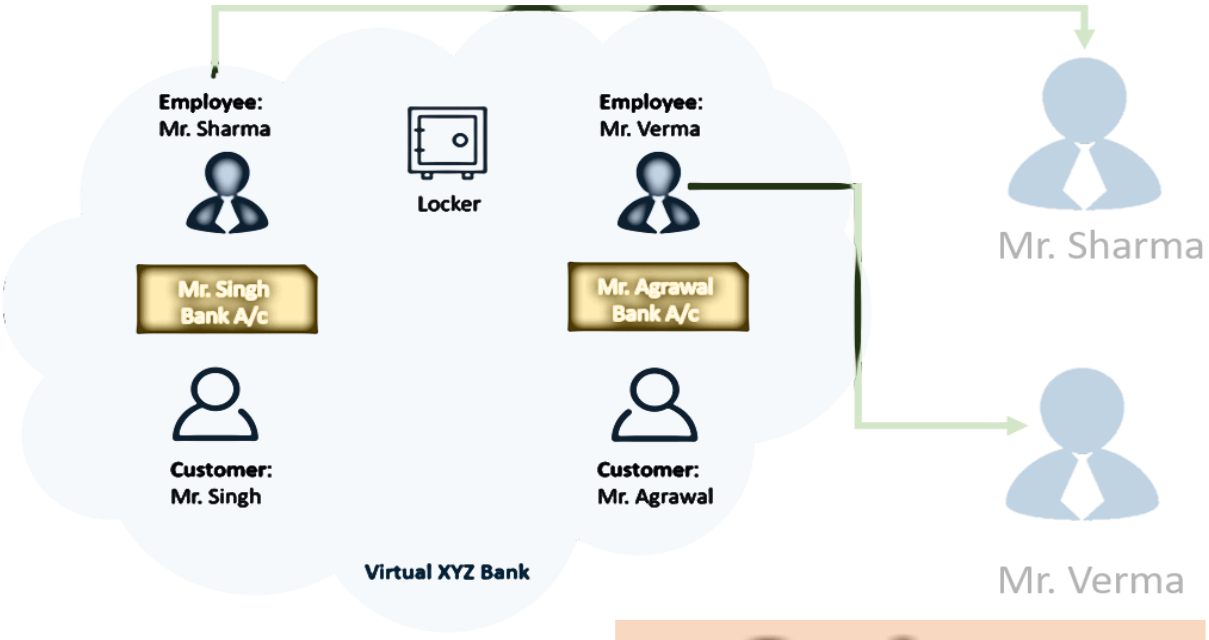
Let's pick Mr. Sharma and Mr. Verma first.

Now because they both are employees, we will write down the things that they have in common
For example, they will have an employee id.

One thing to note here is that Mr. Sharma will have a different employee-id from Mr. Verma, But they both will have some employee id.

Similarly, they will have some designation and of course salaries.

Now we have a set of attributes (properties) that an employee will always have, so we will put a tag over these attributes and name it as Employee.



Can you guess, what actually we have done?

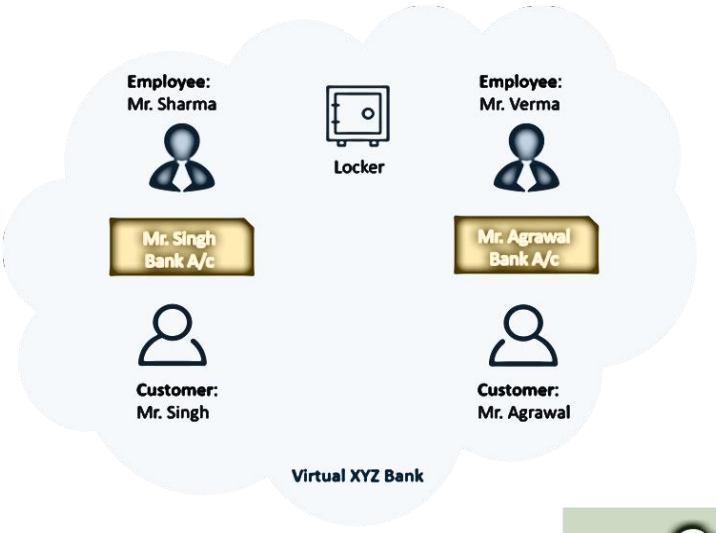
Yes, we have created a blueprint or description for employees. Or we can say, we have created an employee class, and Mr. Sharma and Mr. Verma are objects of Employee class.

Also, we can create as many objects as we want from this employee class.
All newly created employees will have the same properties as

possessed by Mr. Sharma and Mr. Verma.

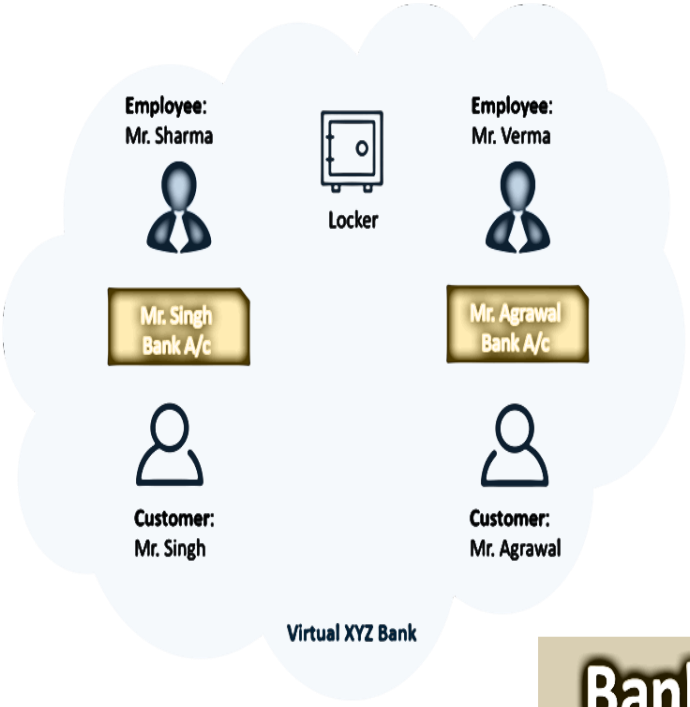
In the same way, let's now pick Mr. Agrawal and Mr. Singh. Their common attributes can be customer_id, account_number, date_of_birth, and address.
Let's put a tag over these attributes as CUSTOMER.

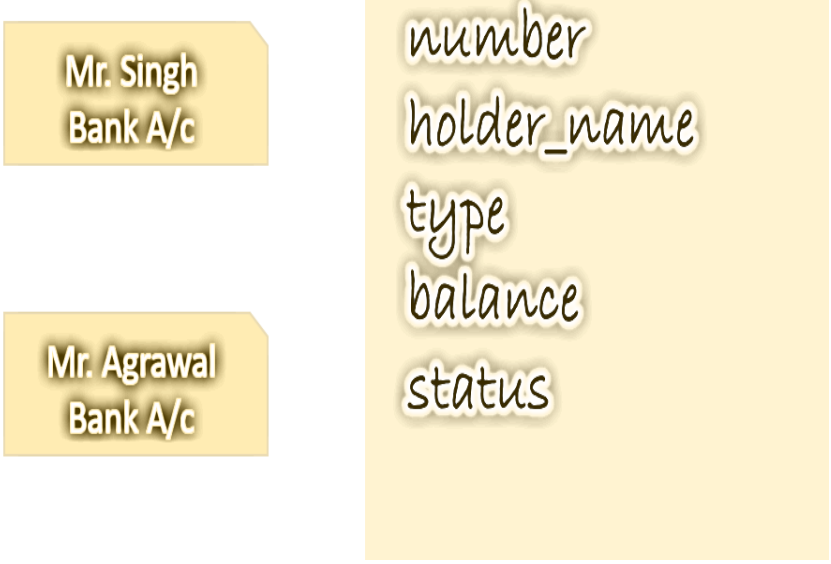
Now every new customer will also have the same properties as possessed by Mr. Agrawal and Mr. Singh.



So far we have created the blueprints for EMPLOYER and CUSTOMER. Let’s create a blueprint for BANK_ACCOUNT as well.

The BANK_ACCOUNT class will look like this and will have account_number, accout_holders_name, the BALANCE in the account, TYPE of account and also it can have the STATUS of the account.

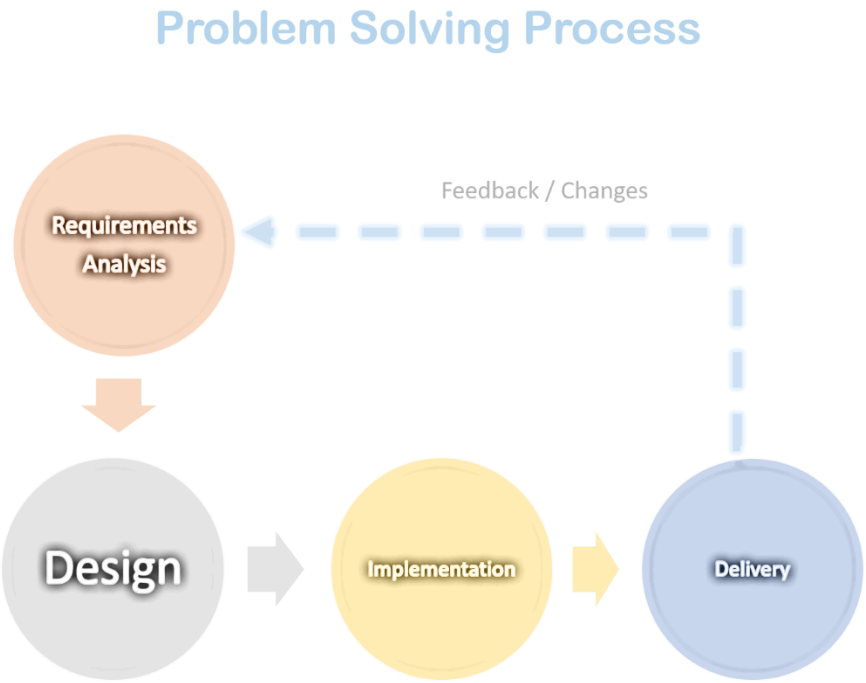




Problem Solving Process

In this section, we will be discussing a general approach to problem-solving.

Let us discuss the steps that are followed in general to solve any problem.



1. First, we focus on the requirements of the problem.
2. Next, on the basis of understanding of requirements, we develop a design or prototype of the solution.
3. Finally, we convert the prototype to the real implementation and deliver it to the customer.

If the customer, after testing the solution, suggests a change,

then the whole process is followed again to reflect that change.

Let's understand this with the help of the previously discussed examples of house construction.

This is how the house is constructed:

1. Initially, all the requirements such as size, area, and no. of the floor are documented.
2. Then a map is created considering these requirements.
3. The details on the map are followed to build real houses.
4. If the owner finds out some flaw after the house is built, he will have to get the new map created with updated requirements, and then only the changes will be made.

The same is the case with car manufacturing:

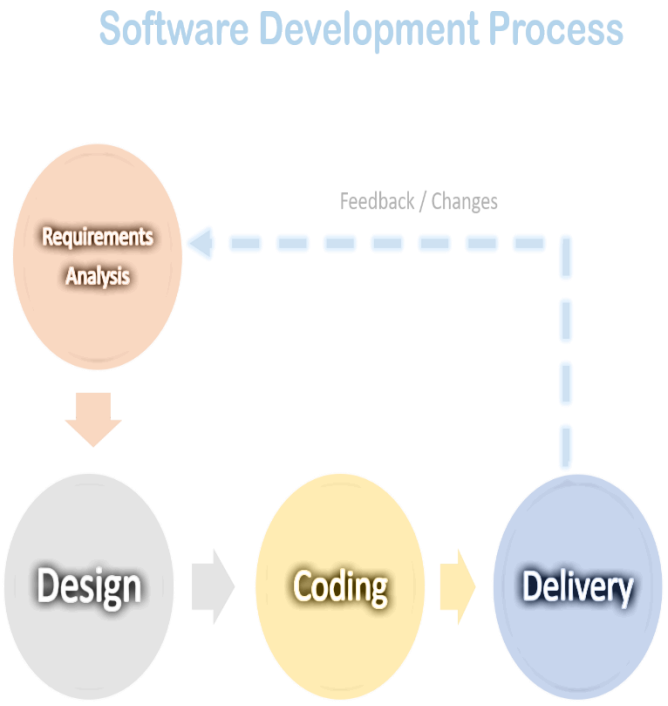
- Requirements to car design.
- Then car design to real car

Software Development Process

We saw the general problem-solving process which consist of 4 steps:

1. Requirement Analysis
2. Design
3. Implementation
4. Delivery

The same set of steps is followed to develop any software. The only difference is that here implementation is the coding phase.



There are **N** number of programming languages in the market that can be used to code software.
You can pick any one of the languages that fits into your portfolio.

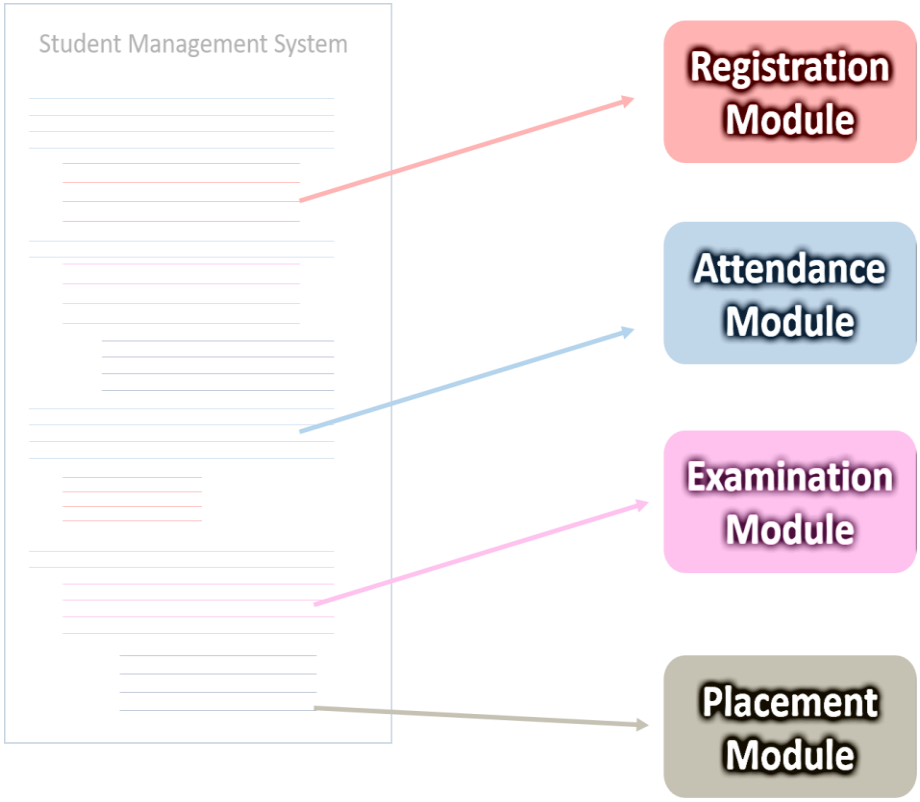
So the coding phase consists of selecting a programming language or framework and converting the software designs to executable code.

Here, one thing that might have made you skeptical for sure is the DESIGN phase.

It is easy to digest that a house can be visualized via a map.

A car can be imagined with a design. But code, How will I visualize the code? How can we create designs for the code?

I will help you to understand this. You must agree that software can never be a single large program, It has to be divided into modules.



Now, a software design or model can be a graphical representation of the Interaction of modules in the software.

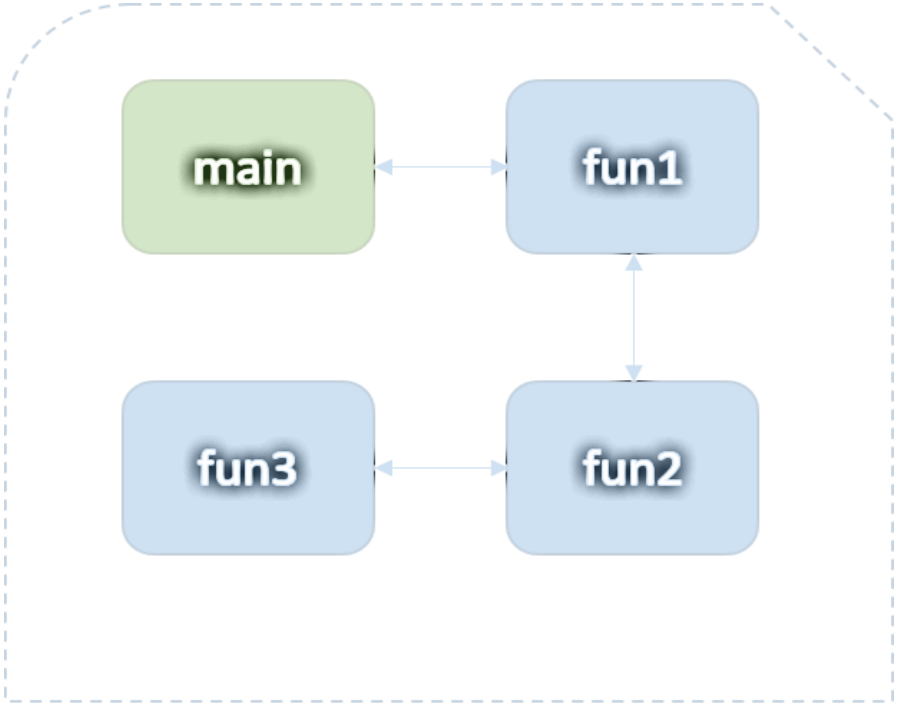
Now, a new question arises i.e., how to divide the software into a module. That means, how to identify the modules in our program.

In the era of C and other procedural languages, the software was considered as the collection of functions or procedures.

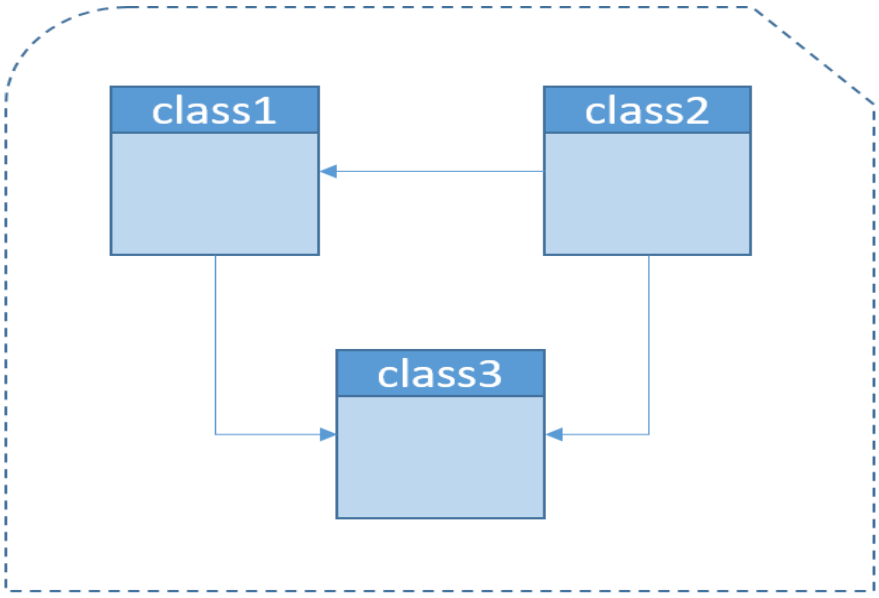
There used to be the main procedure that could interact with other procedures. At that time every procedure was considered as a module.

So when the designs are created in which there is a module for every procedure, they are known as Procedural Designs.

But, this procedural method had several drawbacks.



Procedural Designs



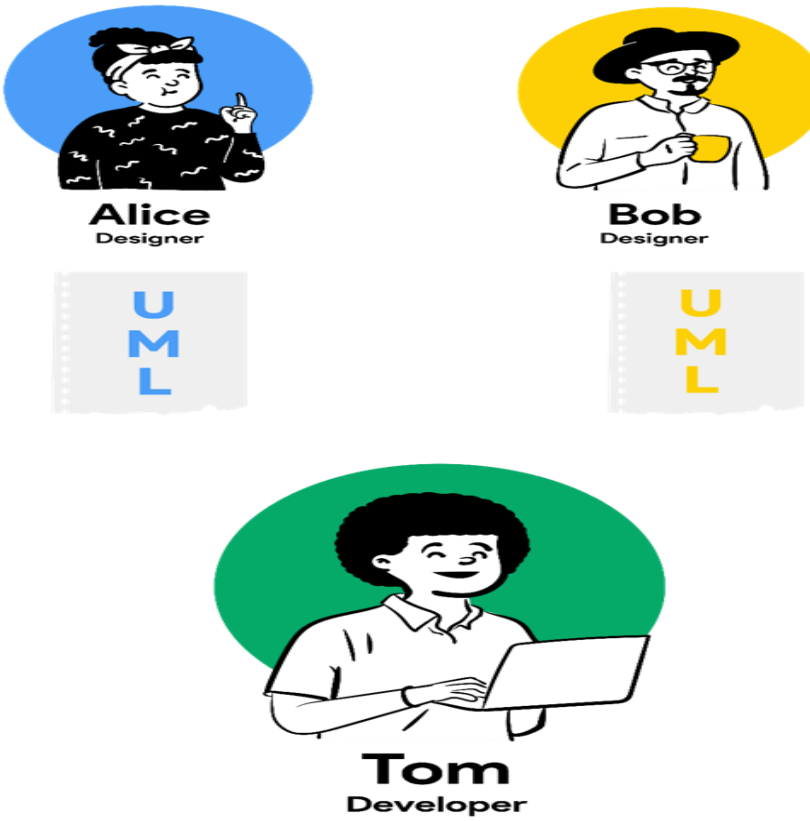
Object Oriented Designs

The evolution of object oriented languages such as C++, and java, gave birth to object oriented designs where classes were considered as modules.

In the object-oriented paradigm, a running program can be seen as a collection of objects collaborating to perform a given task.

Why UML?

Suppose there are two designers Alice and Bob creating designs of different parts of the software.
Now if Alice follows some conventions and Bob follows other conventions, then it becomes very difficult for Tom, the developer, to code these designs correctly.
There should be some standard that must be followed by Alice and Bob, so that their designs are uniform, and seems unambiguous to TOM.



Here comes to the rescue, the UML i.e., **Unified Modelling Language**.

The **Unified Modeling Language (UML)** is a standard graphical language for modeling object-oriented software. It was developed by three software engineers, James Rumbaugh, Grady Booch, and Ivar Jacobson.

In 2004 the Object Management Group (OMG) approved version 2.0 of UML.

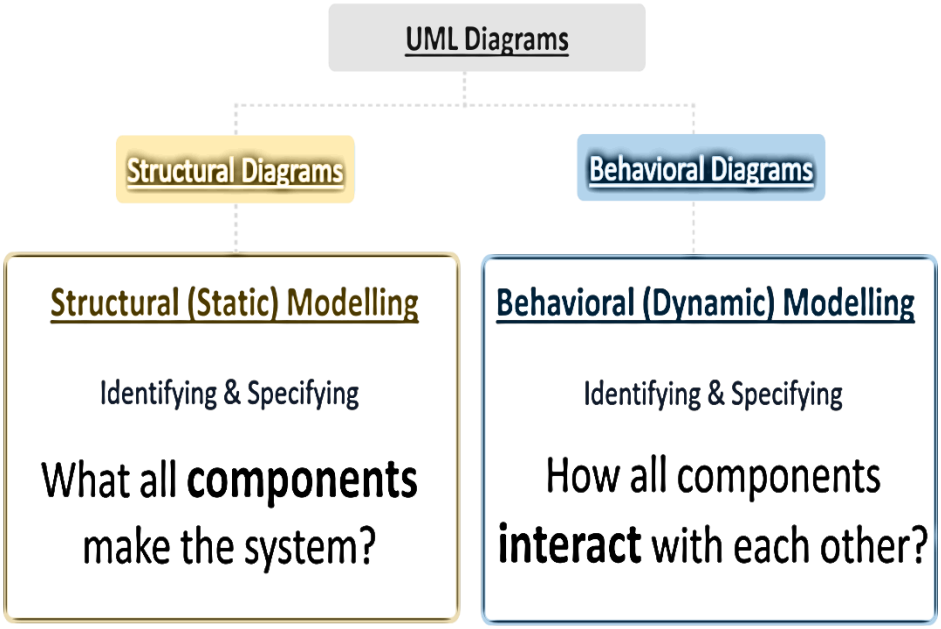
Now, if Alice and Bob create a design by following UML conventions, their designs will be uniform and standard and will make the life of TOM much happier.

UML Diagrams

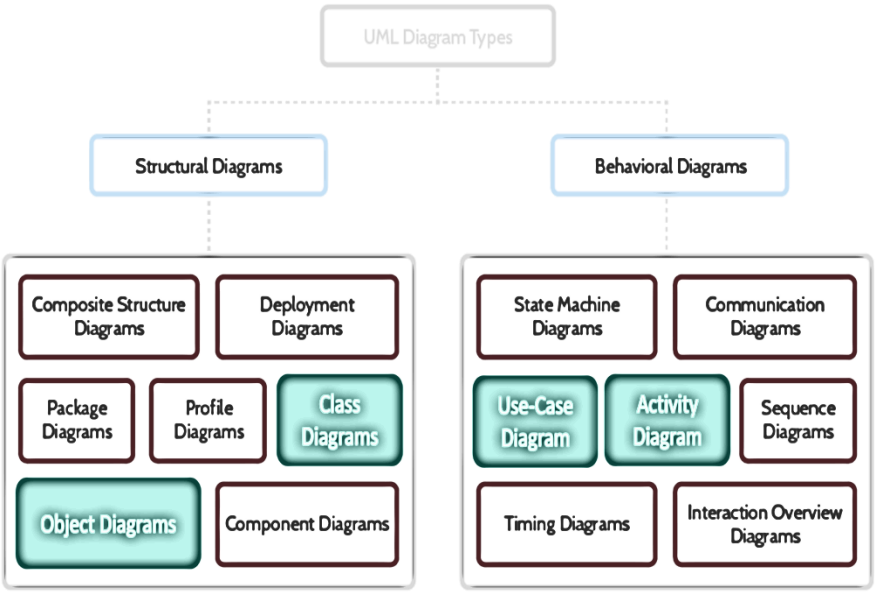
As UML is a graphical language, it supports two types of diagrams that can be classified as Structural and Behavioral.

The structural Diagrams are used to create a static model of the software. That is, it gives an idea about what all components build up the system.

The Behavioral Diagrams are used to create a dynamic model of the software. It tells how the different components or modules interact with each other.



Some of the structural and behavioral diagrams supported by UML are as shown:



Here in this course, we will dive deeper into class diagrams, object diagrams, and also use-case diagrams.

PART-2 :- (UPCOMING)

- Class Diagrams
- Relationships
- Associations
- Object Diagrams
- Multiplicity
- Generalization
- Aggregation
- Composition
- Bank Class Diagram
- Why Use Case Diagrams?
- Use Case Diagrams
- Use Case Diagrams : Advanced Concepts

HIMANSHU KUMAR(LINKEDIN)

https://www.linkedin.com/in/himanshukumarmahuri

CREDITS- INTERNET

DISCLOSURE- THE DATA AND IMAGES ARE TAKEN FROM GOOGLE AND INTERNET.

CHECKOUT AND DOWNLOAD MY ALL NOTES

***LINK*- https://linktr.ee/exclusive_notes**