Introduction to Object-Oriented Programming

What is OOP?

- Object-Oriented Programming or OOP is a programming paradigm which consists of programming languages that use objects and classes in programming.
- It is one of the most popular and commonly used programming paradigms due to its capabilities
- Although a single programming language can exhibit characteristics of more than one programming paradigms, the most popular OOP programming languages include:
    - Java
    - C++
    - Ruby
  - Others such as python, Javascript, and Visual Basic.Net only exhibit some features of OOP but they are not purely OOP

Cont.

- OOP plays a very critical role in human computer interaction since it implements real-world entities in programming.
    - i.e It has different components that takes real world objects and performs actions on them, making live interactions between man and the machine.
- It is important to note that different OOP languages vary in terms of complexity, library collection, capabilities, among others.
- For instance Java creates very powerful GUIs Thus making it possible to create user-friendly applications. but it is “wordy” (uses many lines of code.
- On the other hand python can analyze large amounts of data with very few lines of code.

Characteristics of OOP

- OOP paradigm describes a real-life system where interactions are among real objects.
- It models applications as a group of related objects that interact with each other.
- Programming starts with the concept of real world objects and classes.
- Application is divided into numerous packages.
- A package is a collection of classes.
- A class is an encapsulated group of similar real world objects.

OOP concepts

- OOP concepts are the features that differentiate OOP languages from other paradigms. They include:
  - Objects
  - Classes
  - Inheritance
  - Abstraction
  - Encapsulation
  - Polymorphism
  - Message Passing

OOP Concepts: Object

- It is a basic unit of Object-Oriented Programming and represents the real-life entities.
- Any real-life entity can become an object in OOP.
- An object is also an instance of a class meaning that it emanates from a class. Therefore, each object must belong to a class.
- When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.
- Each object contains data and code to manipulate the data.
- Objects can interact without having to know details of each other's data or code.

Cont.

- An object has three main characteristics:
  - Identity – this is the unique name used to identify the object
  - State- features or characteristics that define an object.
  - Behavior- Functionality of an object. i.e what an object does
- For example a dog is a real-life entity.
  - Identity- name of the dog (E.g Bosco)
  - State include things such as- Breed, Age, color,
  - Behavior- what a dog does such as (barking, wagging tale, sleep, Eat)
- Try and think of other real-life entities that can form objects then enumerate their identity, state, and behavior.

OOP Concepts: Classes

- A class is a collection of similar objects which is created by a programmer as a user-defined data type.
- A class can consist as many objects as possible so long as those objects share common characteristics.
- The programmer has to be careful when creating a class because the class influences the objects that can be contained in it.
- For example: think of a programmer who wants to develop a program that deals with automobiles.
  - If the programmer creates a class known as Vehicles, then inside that class there can be objects such as cars, lorries, vans, buses, pick-ups, trailers. This is so because all these objects have common character tics that qualify them to be vehicles such as they all have engines, wheels, windscreen, steering wheel pedals etc.
  - Therefore the class Vehicles is more of a general class.

Cont.

- Approach two if the programmer wants to be more specific towards a particular category of vehicles such as cars.
  - Then he can create a class known as Cars and then inside it have objects such as Toyota Corolla, BMWX1, Audi A3, Volkswagen Golf, etc. all these cars have some common characteristics such as they all have four wheels.
- A class consists of data members and member functions, which can be accessed and used by creating an instance of that class.
  - i.e elements of a class can only be accessed by creating an object of that class.
- It represents the set of properties or methods that are common to all objects of the same type.

Cont.

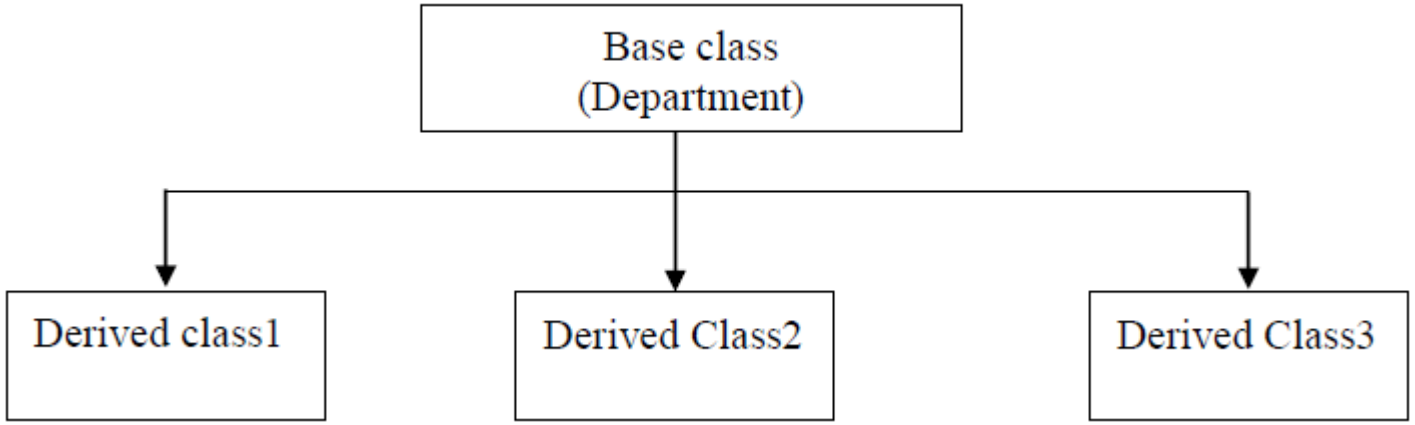
- A class is like a blueprint for an object. Therefore, all characteristics of an object are defined in the class in which the object belongs to.
- For instance:
  - In class Car you can define characteristics such as:
    - passenger capacity
    - Engine capacity
    - Make
- An object can only belong to one class while a class can have more than one objects.

OOP Concepts: Inheritance

- Inheritance is an important pillar of OOP. It is the capability of a class to derive properties and characteristics from another class.
- So when we create a class, we do not need to write all the properties and functions again and again, as these can be inherited from another class that possesses it.
- Thus, Inheritance allows the user to reuse the code whenever possible and reduce its redundancy.
- For example if you have two classes (1) Class Animal and (2) class pets. You don’t have to repeat common characteristics of class animal in class pets inheritance can help class pets to acquire these properties.

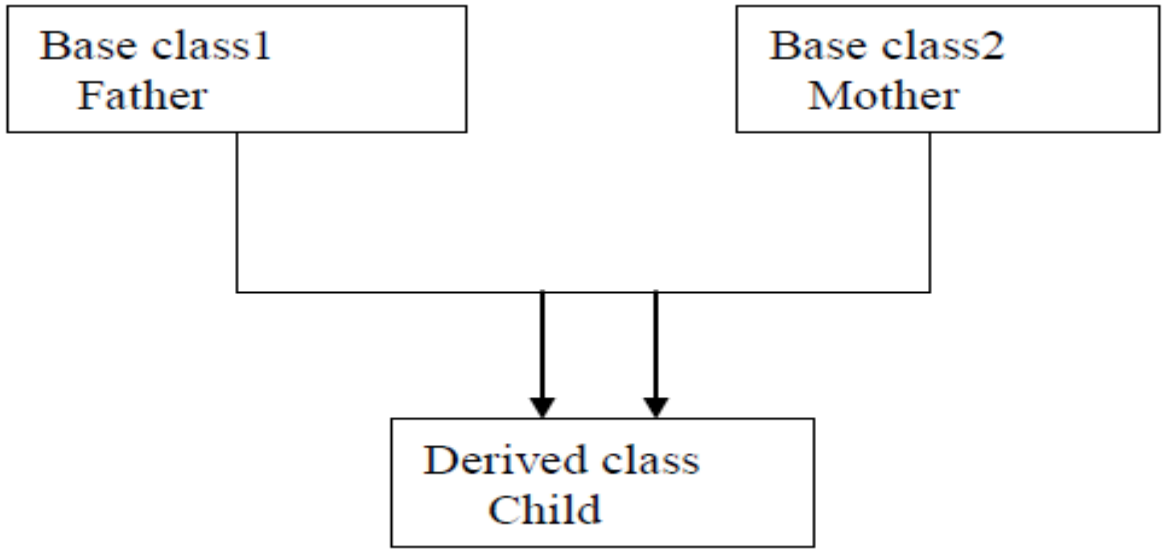
Types of Inheritance

- **Hierarchical Inheritance:**
  - In this relationship, multiple derived classes can inherit features from a single base class Multiple
  - A single derived class can inherit features from multiple base classes.
  - A base class is a class that is inherited from while a derived class is one that inherts from another class.



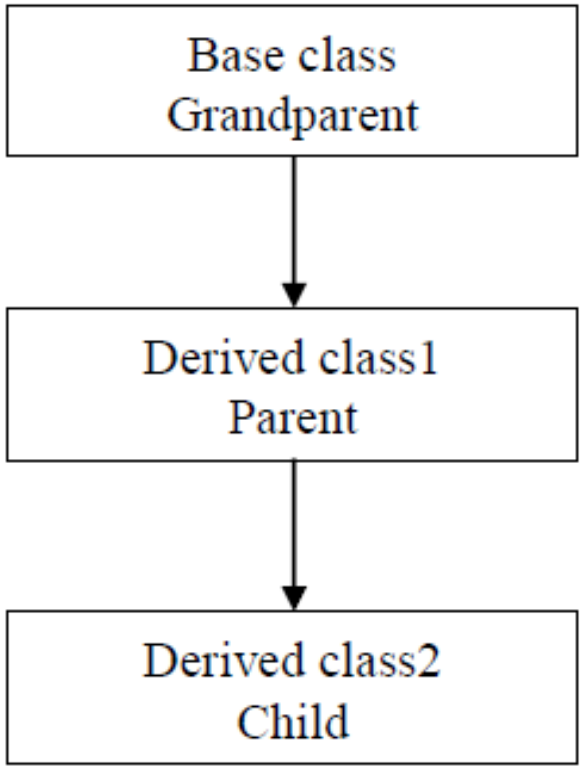
Types of Inheritance Cont.

- **Hierarchical Inheritance cont:**
  - It can also be represented using the diagram below in which a single derived class inherits from multiple base classes.



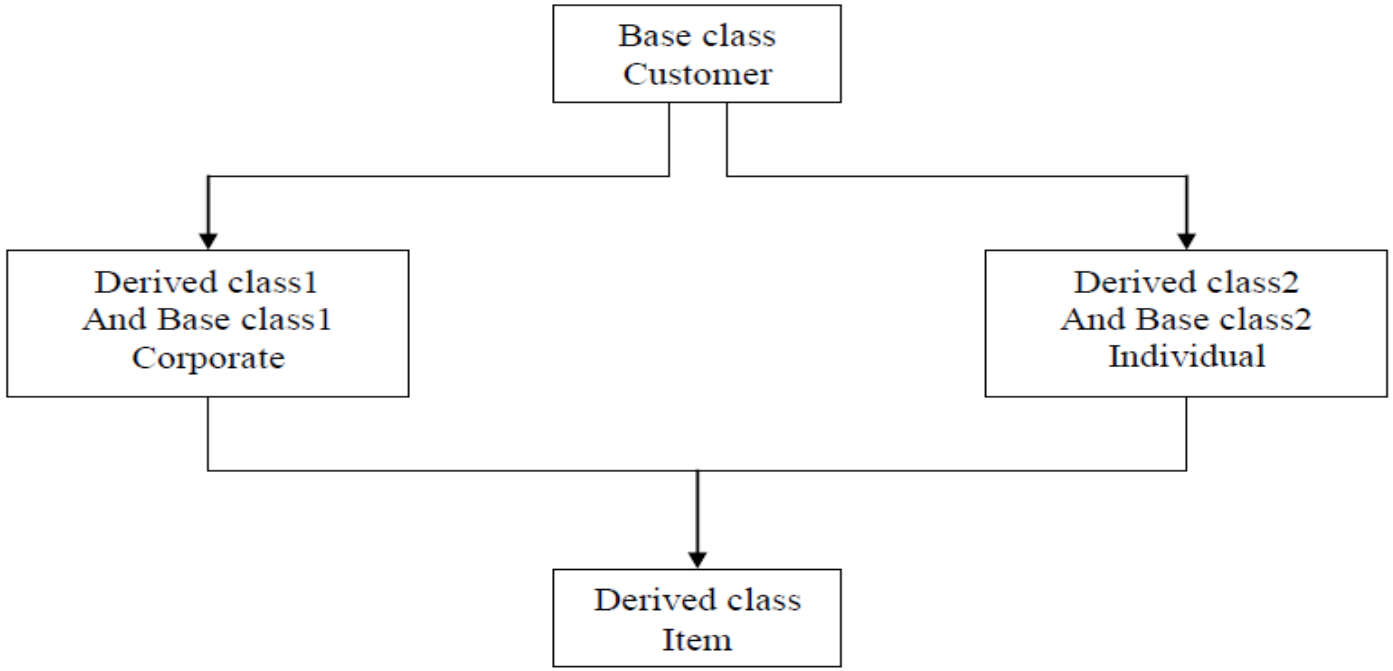
Types of Inheritance Cont.

- **Multi-level inheritance:**
  - This is where a derived class can inherit features from other derived classes



Types of Inheritance Cont.

- **Hybrid inheritance:**
  - This is a combination of more than one types of inheritance. For example a derived class can have both hierarchical and multiple inheritance



OOP Concepts: Abstraction

- Data abstraction refers to providing only essential information about the data to the outside world and hiding the background details or implementation.
  - Consider a real-life example of a person sending money via mpesa, the person only knows that by following a series of steps and finally pressing the send button the money is sent.
  - The user is never concerned about how the money moves from one account to another.
- Data abstraction is one of the most essential features of OOP.
- With abstraction a developer is able to develop software that is user friendly.
- For instance with a GUI a lot of background processing and algorithm is hidden from the user.

Cont.

- Data Abstraction has been applied in each and every software or and even gadgets that we use on a daily basis.
  - For example think of your student portal once you open it a login page appears you input our registration number and password, if they are correct you are logged in if not you are given an error message.
  - So database details, control structures, and other algorithms used are hidden from you as a general user.
- In some OOP languages such as JAVA abstraction within a class can be defined by creating an abstract class.

OOP Concepts: Encapsulation

- Encapsulation is defined as the wrapping up of data under a single unit.
- It is the mechanism that binds together code and the data it manipulates.
- In OOP classes play the vital role of encapsulating data and properties within them.
  - For example if you have a class known as Box with properties such as width, legth, height, color etcs. When you refer to an instance of class box (i.e an object of class box) you as well or automatically consider the properties that are defined in that class.
- In Encapsulation, the variables or data of a class are hidden from any other class and can be accessed only through any member function of their class in which they are declared. So it is also known as **data-hiding**.

Cont. Example of Encapsulation

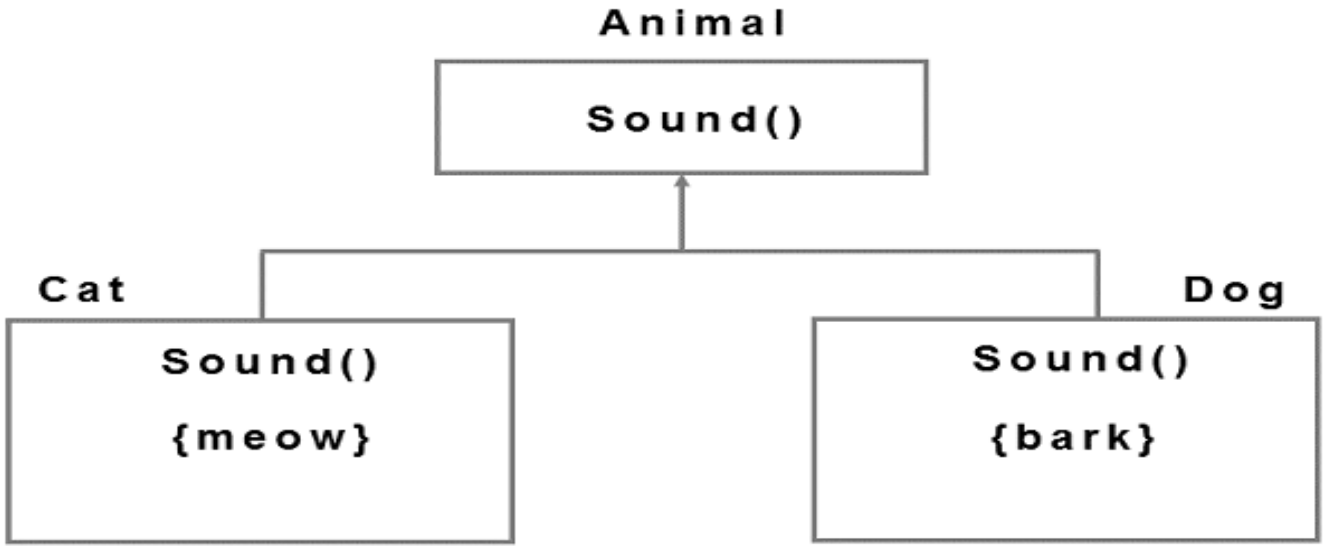
- Consider a real-life example of encapsulation, in a company, there are different sections like the accounts section, finance section, sales section, etc.
- The finance section handles all the financial transactions and keeps records of all the data related to finance.
- Similarly, the sales section handles all the sales-related activities and keeps records of all the sales. Now there may arise a situation when for some reason an official from the finance section needs all the data about sales in a particular month.
- In this case, he is not allowed to directly access the data of the sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data.
- This is what encapsulation is. Here the data of the sales section and the employees that can manipulate them are wrapped under a single name “sales section”.

OOP Concepts: Polymorphism

- The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.
- For example, A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee.
- So the same person posses different behavior in different situations. This is called polymorphism.
- Polymorphism allows for the same method to be used in different parts of a program or in different classes to have different meanings.

Cont. Example

- Here is an example. We have a base class known as animal with a method known as sound () and two derived classes cat and Dog. However, class Cat implements method sound () as {Meow} while in class Dog sound() is implemented as {bark}



OOP Concepts: Message Passing

- It is a form of communication used in object-oriented programming as well as parallel programming.
- Objects communicate with one another by sending and receiving information to each other.
- A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results.
- Message passing involves specifying the name of the object, the name of the function, and the information to be sent.

Advantages of OOP

- **Re-usability**
  - It means reusing some facilities rather than building them again and again. This is done with the use of a class and inheritance.
- **Data Redundancy**
  - This is a condition created at the place of data storage (you can say Databases)where the same piece of data is held in two separate places.
  - So the data redundancy is one of the greatest advantages of OOP. If a user wants a similar functionality in multiple classes, he/she can go ahead by writing common class definitions for similar functionalities and inherit them.
- **Code Maintenance**
  - This feature is more of a necessity for any programming languages; it helps programmers in maintaining code since OOP has sound program structure.

Cont.

- **Security**
  - With the use of data hiding and abstraction mechanism, we are filtering out limited data to exposure, which means we are maintaining security and providing necessary data to view.
- **Design Benefits**
  - Here the Object-Oriented Programs forces the designers to have a long and extensive design phase, which results in better designs and fewer flaws. After a time when the program has reached some critical limits, it is easier to program all the non-OOP's one separately.

Cont.

- **Problems solving**
  - Decomposing a complex problem into smaller chunks or discrete components is a good practice. OOP is specialized in this behavior, as it breaks down your software code into bite-sized –object at a time.
  - The broken components can be reused in solutions to different other problems (both less and more complex), or either they can be replaced by the future modules that relate to the same interface with implementations details.