Object Oriented Programming

Rahee sutrave.

Information about Object Oriented Programming:

- An object-oriented programming is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.

- Object-oriented programming is a programming paradigm that provides a means of structuring programs so that properties and behaviors are bundled into individual objects.

- It is an approach for modeling concrete, real-world things, like cars, as well as relations between things, like companies and employees, students and teachers, and so on. OOP models real-world entities as software objects that have some data associated with them and can perform certain functions.

Major principles of object-oriented programming system are given below.

- Class
- Object
- Method
- Inheritance
- Polymorphism
- Data Abstraction
- Encapsulation

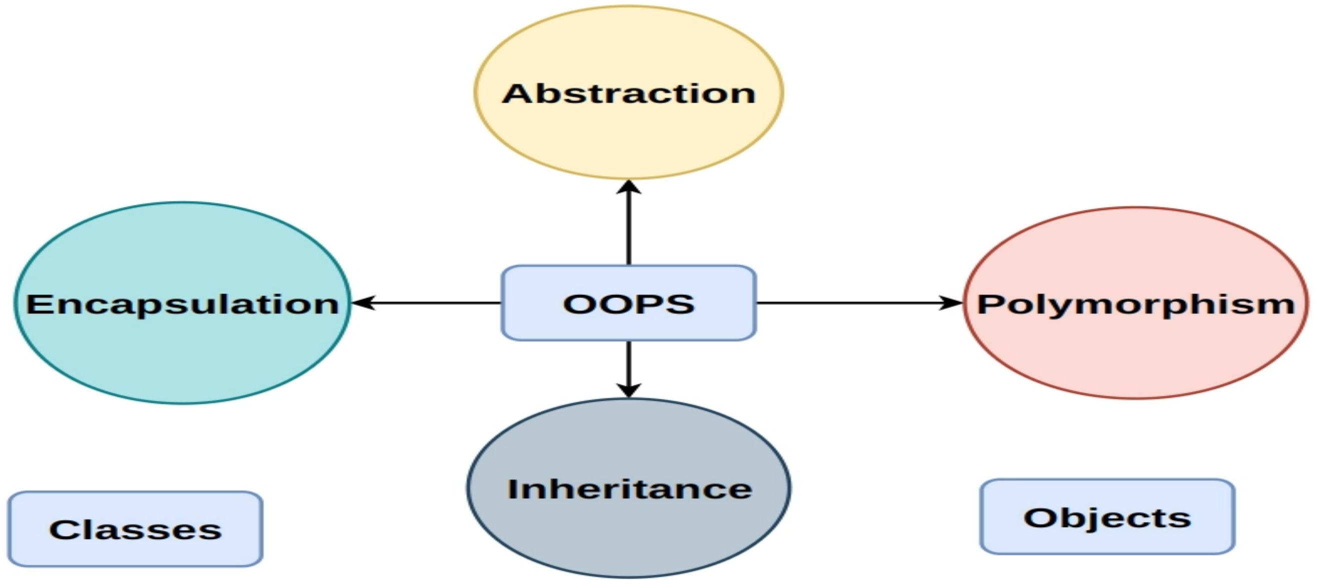
Usability of OOP's:

- In python, we can easily create and use classes and objects. By which it focuses on writing the reusable code.
- It is a widespread technique to solve the problem by making things. Python works fine with or without the oops concept with reliable and short syntax .

- It provide the following benefits to the coder:

  - Easy to Learn & Code
  - Free and High Level Language
  - Portable
  - Large Collection of Packages and Modules
  - Interpreted

Feature of OOP's:



Class :

- A class is a template that consists of the data members or variables and functions and defines the properties and methods for a group of objects.
- The compiler does not allocate memory whenever you define a class.

- Ex: You can define a class called Vehicle. Its data fields can be vehicle_name, model_number, color, date_of_manufacture, etc.

Object:

- An object is nothing but an instance of a class. Each object has its values for the different properties present in its class.
- The compiler allocates memory for each object.

- Ex: The different objects of the class Vehicle can be Car, Bike, Bicycle, etc. Each of them will have its values for the fields like color, model_number, etc.

Abstraction:

- The literal meaning of abstraction is to remove some characteristics from something to reduce it to a smaller set.
- Object Oriented Programming abstraction exposes only the essential information of an object to the user and hides the other details.

Advantages of abstraction:
•It enables code reuse by avoiding code duplication.
•It enhances software security by making only necessary information available to the users and hiding the complex ones.

Inheritance:

- Inheritance is one of the most important features of object oriented programming. It allows a class to inherit the properties and methods of another class called the parent class, the base class, or the super-class.
- The class that inherits is called the child class or sub-class.

- Ex: A parent class called “Shape” and other classes like Square, Circle, Rectangle, etc. Since all these are also shapes, they will have all the properties of a shape so that they can inherit the class Shape.

Polymorphism:
- The word polymorphism means to have many forms. So, by using polymorphism, you can add different meanings to a single component.

There are two types of polymorphism:

1. Run-time polymorphism:
    - Ex: A function “add” is created. Now, when two integers pass to this function, it will return their sum, while on passing two strings, it will return their concatenation.

2. Compile-time polymorphism:

- Ex: A parent class called “Shape” with a method named “findArea” that calculates and returns the area of the shape. Several sub-classes inherit from the “Shape,” like Square, Circle, Rectangle, etc. Each of them will define the function “findArea” in its way, thus overriding the function.

Encapsulation:

- Encapsulation is also an essential aspect of object-oriented programming.
- It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.AD

All About Inheritance:

- Inheritance is a way of representing real-world relationships between the two.
- Inheritance is the procedure in which one class inherits the attributes and methods of another class.
- The class whose properties and methods are inherited is known as the parent class or superclass. And the class that inherits the properties from the parent class is the child class or derived class.

There are broadly five forms of inheritance based on the involvement of parent and child classes:

1. Single Inheritance:

This is a form of inheritance in which a class inherits only one parent class. This is the simple form of inheritance and hence, is also referred to as simple inheritance.

2. Multiple Inheritance:

- An inheritance become multiple inheritance when a class inherits more than one parent class. The class after inheriting properties from various parent classes has access to all of its object.
- Ex. Suppose 3 parent class P1,P2,P3 all have different properties, and child class inherits P1 and P2 , another child class of P2,P3 are multiple inheritance.

3. Multi-level Inheritance:

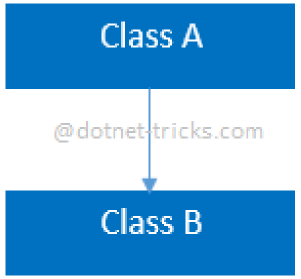
- Each class is inherited by followed class is known as multi-level inheritance.
- Ex: A parent class P1 inherited by child class C1, this child class inherited by another class P2 and this process goes on.

4. Hierarchical inheritance:

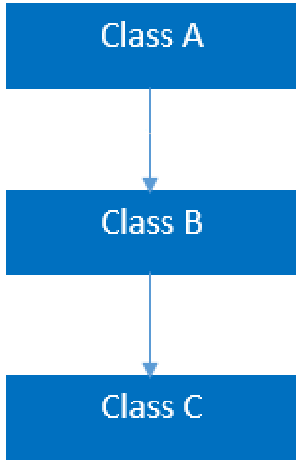
- It is more than one derived class are created from a single base. In this, various Child classes inherit a single Parent class.
- Ex: classes BMW and Audi inherit class Car.

5. Hybrid inheritance:

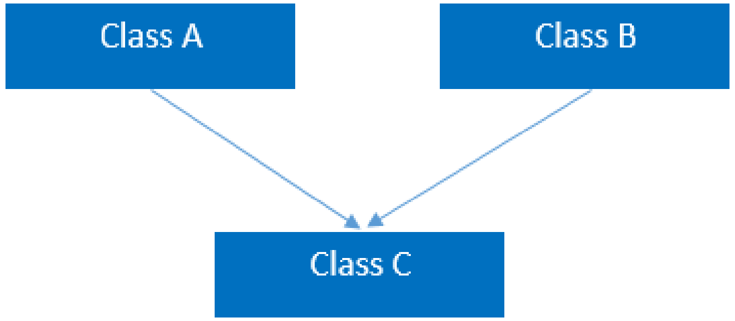
- This form combines more than one form of inheritance. Basically, it is a blend of more than one type of inheritance.
- Ex: If two classes, ‘Child_1’ and ‘Child_2’, are derived from the base class ‘Parent’ using hierarchical inheritance. Another class, ‘Child_3’, is derived from classes ‘Child_1’ and ‘Child_2’ using multiple inheritances. The class ‘Child_3’ is now derived using hybrid inheritance.



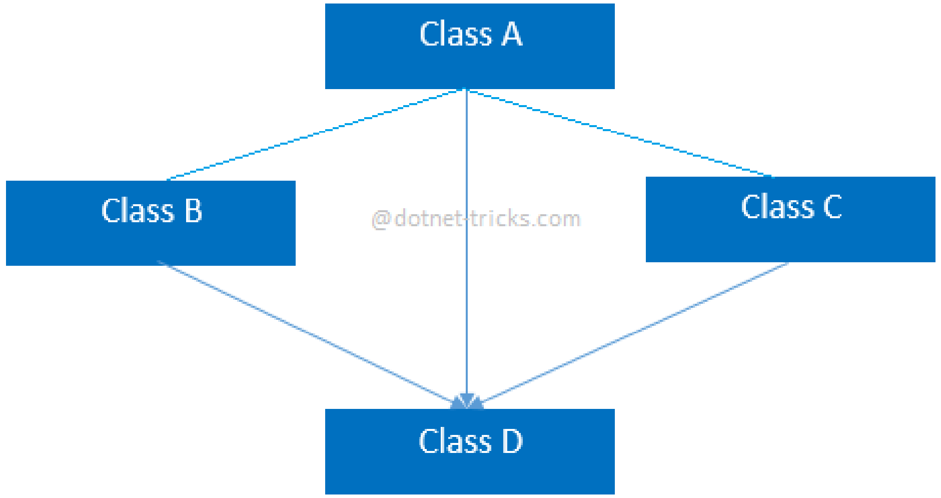
Single Inheritance



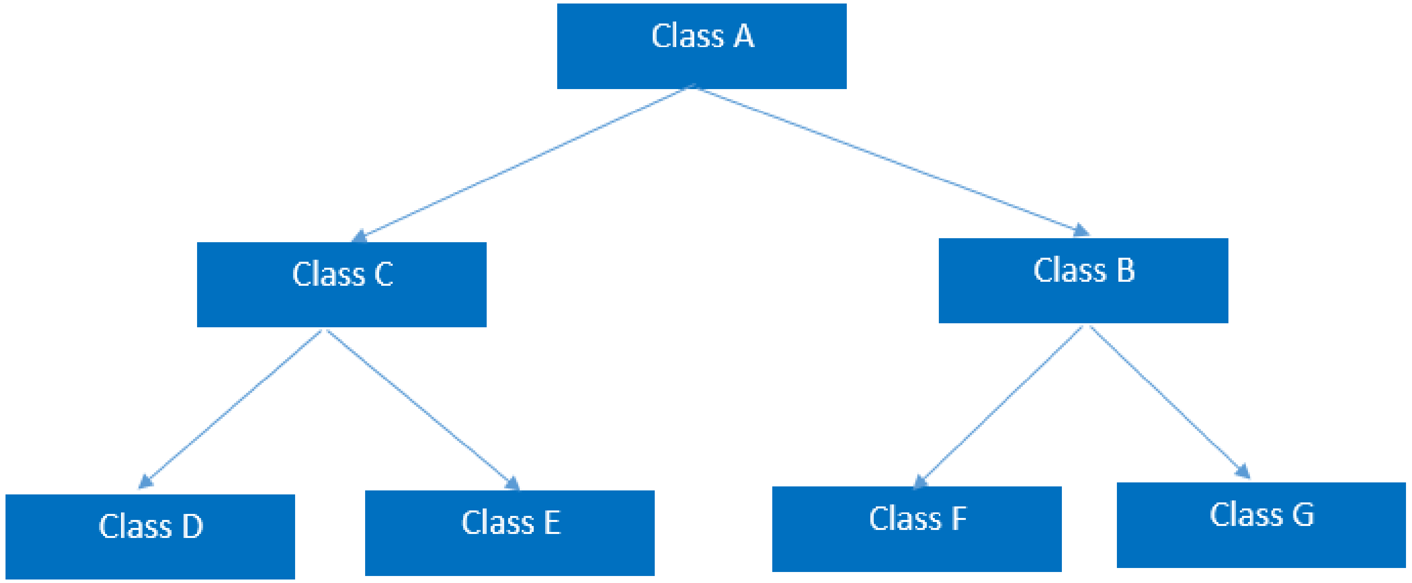
Multi-Level Inheritance



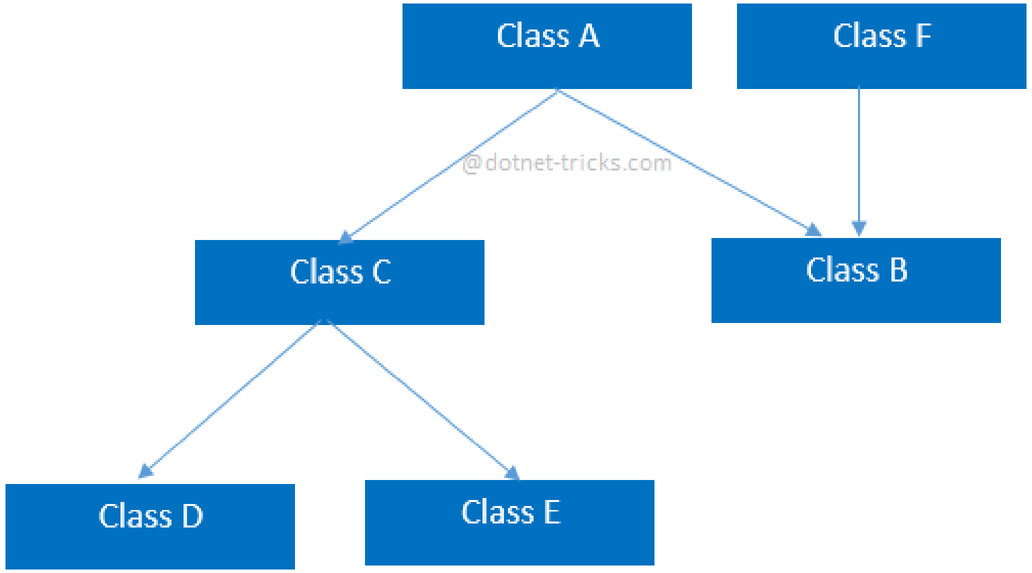
Multiple Inheritance



Multipath Inheritance

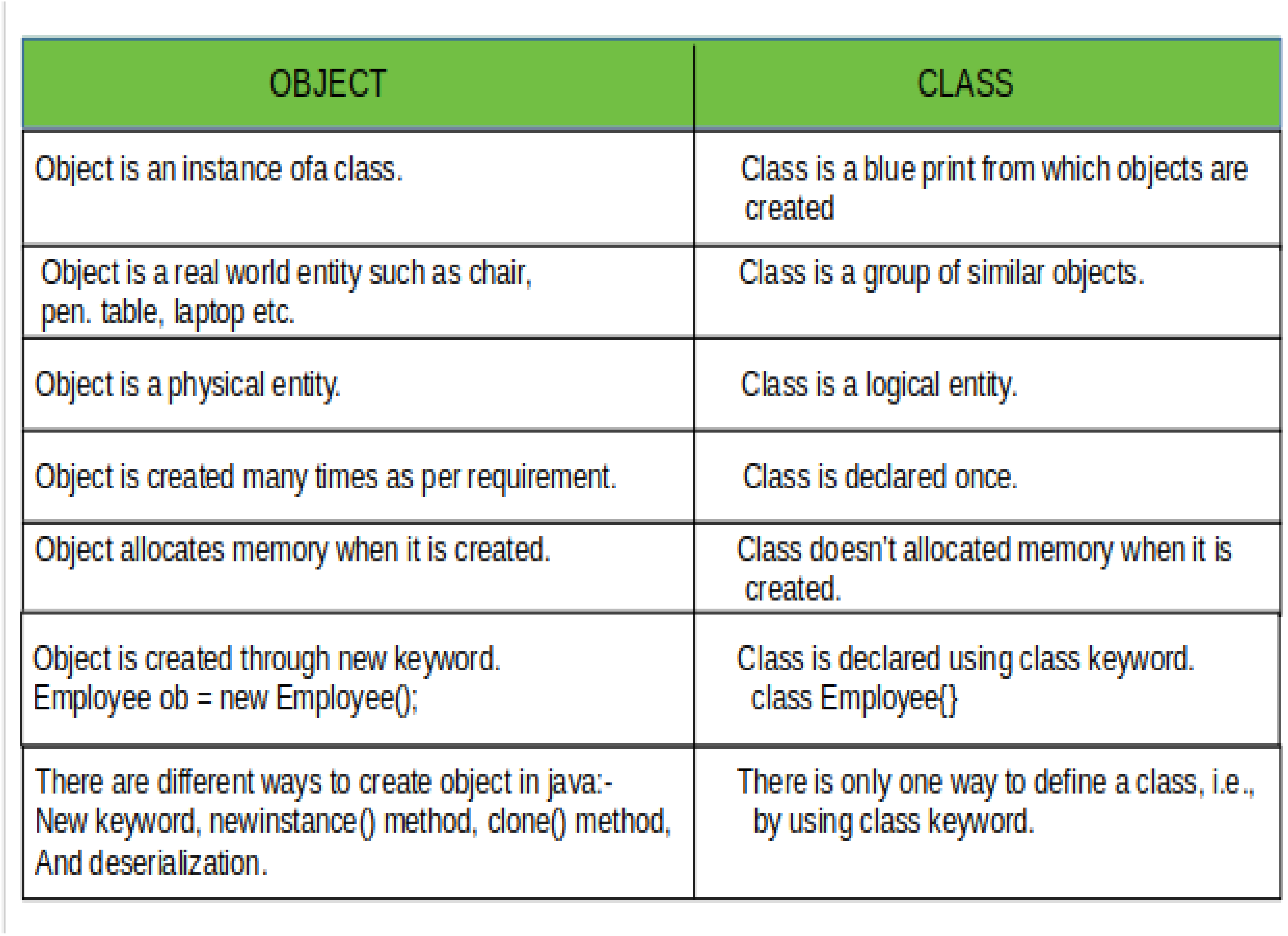


Hierarchical Inheritance



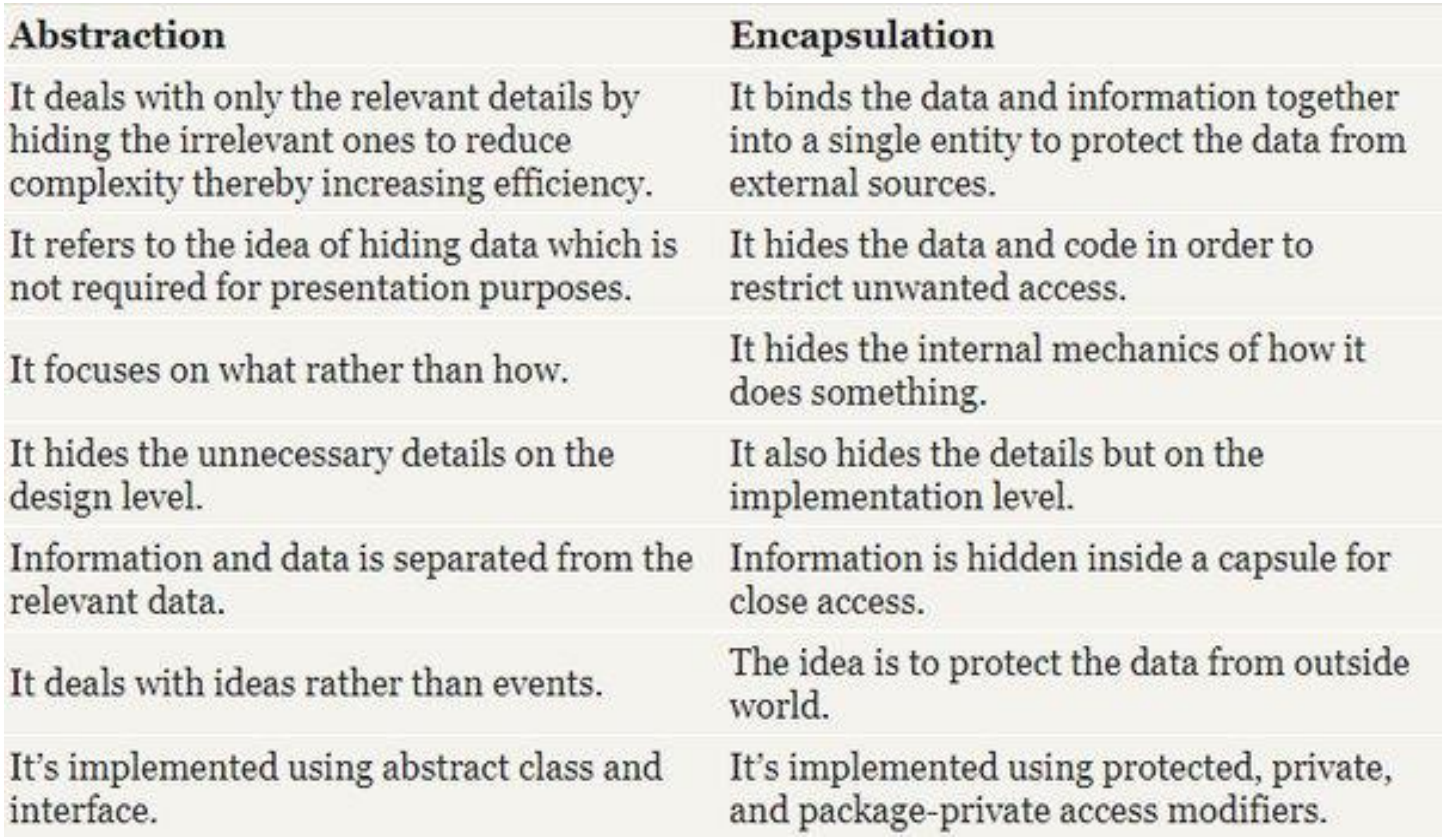
Hybrid Inheritance – (a combination of Hierarchical and multiple)

Difference between class and object:



OBJECT	CLASS
Object is an instance ofa class.	Class is a blue print from which objects are created
Object is a real world entity such as chair, pen. table, laptop etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created many times as per requirement.	Class is declared once.
Object allocates memory when it is created.	Class doesn't allocated memory when it is created.
Object is created through new keyword. Employee ob = new Employee();	Class is declared using class keyword. class Employee{}
There are different ways to create object in java:- New keyword, newinstance() method, clone() method, And deserialization.	There is only one way to define a class, i.e., by using class keyword.

Difference between data abstraction and encapsulation:



Abstraction	Encapsulation
It deals with only the relevant details by hiding the irrelevant ones to reduce complexity thereby increasing efficiency.	It binds the data and information together into a single entity to protect the data from external sources.
It refers to the idea of hiding data which is not required for presentation purposes.	It hides the data and code in order to restrict unwanted access.
It focuses on what rather than how.	It hides the internal mechanics of how it does something.
It hides the unnecessary details on the design level.	It also hides the details but on the implementation level.
Information and data is separated from the relevant data.	Information is hidden inside a capsule for close access.
It deals with ideas rather than events.	The idea is to protect the data from outside world.
It's implemented using abstract class and interface.	It's implemented using protected, private, and package-private access modifiers.

Use of self in python:

- self represents the instance of the class. By using the “self” we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.
- Python does not use the @ syntax to refer to instance attributes. Python decided to do methods in a way that makes the instance to which the method belongs be passed automatically, but not received automatically.
- Self must be provided as a First parameter to the Instance method and constructor. If you don’t provide it, it will cause an error.
- Self is a convention not a python keyword.
- Self is parameter in instance method and user can use another parameter name in place of it.
- Ex:
```
class car( ):
    def __init___(self, model, color):
            self.model= model
             self.color= color
```

Here self is used to assign the values passed as arguments to the object attributes.

Use of __init__:

- The __init__ method in python is equivalent to C++ constructor in an Object Oriented Programming approach.

- __init__ can be use as default constructor . A constructor with no mandatory parameters is called a default constructor.

- This function called as every time an object is created from the class, it uses to initialize the object's attributes and serves no other purpose. It is only used within classes.

- In __init__ method uses keyword Self to assign the values passed as arguments to the object attributes.

- Ex:
```
class car( ):
    def __init___(self, model, color):
            self.model= model
             self.color= color
```

Here __init__ uses as constructor to initialize the objects.

How to create class in python:

- Python is an object oriented programming language.
- Almost everything in Python is an object, with its properties and methods.
- A Class is like an object constructor, or a "blueprint" for creating objects

- To create a class use keyword class:

Ex:

```
Class person:
   def__init__(self, name, age):
          self.name = name
          self.age = age

P1= person(‘Shreya’ , 26)

Print(p1.name)
Print(p1.age)
```