**OOPS CONCEPTS**

## **What is OOPS?**

****OBJECT ORIENTED PROGRAMMING (OOP)**** is a programming concept that works on the principles of abstraction, encapsulation, inheritance, and polymorphism. It allows users to create the objects that they want and then, create methods to handle those objects. The basic concept of OOPs is to create objects, re-use them throughout the program, and manipulate these objects to get results.

**OOPs Concepts:**

* [Polymorphism](https://www.geeksforgeeks.org/polymorphism-in-java/)
* [Inheritance](https://www.geeksforgeeks.org/inheritance-in-java/)
* [Encapsulation](https://www.geeksforgeeks.org/encapsulation-in-java/)
* [Abstraction](https://www.geeksforgeeks.org/abstraction-in-java-2/)
* [Class](https://www.geeksforgeeks.org/classes-objects-java/)
* [Object](https://www.geeksforgeeks.org/classes-objects-java/)
* [Method](https://www.geeksforgeeks.org/methods-in-java/)



## **Core OOPS concepts are**

### **1) Class**

The class is a group of similar entities. It is only an logical component and not the physical entity. For example, if you had a class called “Expensive Cars” it could have objects like Mercedes, BMW, Toyota, etc. Its properties(data) can be price or speed of these cars. While the methods may be performed with these cars are driving, reverse, braking etc.

### **2) Object**

An object can be defined as an instance of a class, and there can be multiple instances of a class in a program. An Object contains both the data and the function, which operates on the data. For example - chair, bike, marker, pen, table, car, etc.

### **3) Inheritance**

Inheritance is an OOPS concept in which one object acquires the properties and behaviors of the parent object. It’s creating a parent-child relationship between two classes. It offers robust and natural mechanism for organizing and structure of any software.

### **4) Polymorphism**

Polymorphism refers to the ability of a variable, object or function to take on multiple forms. For example, in English, the verb *run* has a different meaning if you use it with *a laptop*, *a foot race*, and *business*. Here, we understand the meaning of *run* based on the other words used along with it.The same also applied to Polymorphism.

### **5) Abstraction**

An abstraction is an act of representing essential features without including background details. It is a technique of creating a new data type that is suited for a specific application. For example, while driving a car, you do not have to be concerned with its internal working. Here you just need to concern about parts like steering wheel, Gears, accelerator, etc.

### **6) Encapsulation**

Encapsulation is an OOP technique of wrapping the data and code. In this OOPS concept, the variables of a class are always hidden from other classes. It can only be accessed using the methods of their current class. For example - in school, a student cannot exist without a class.

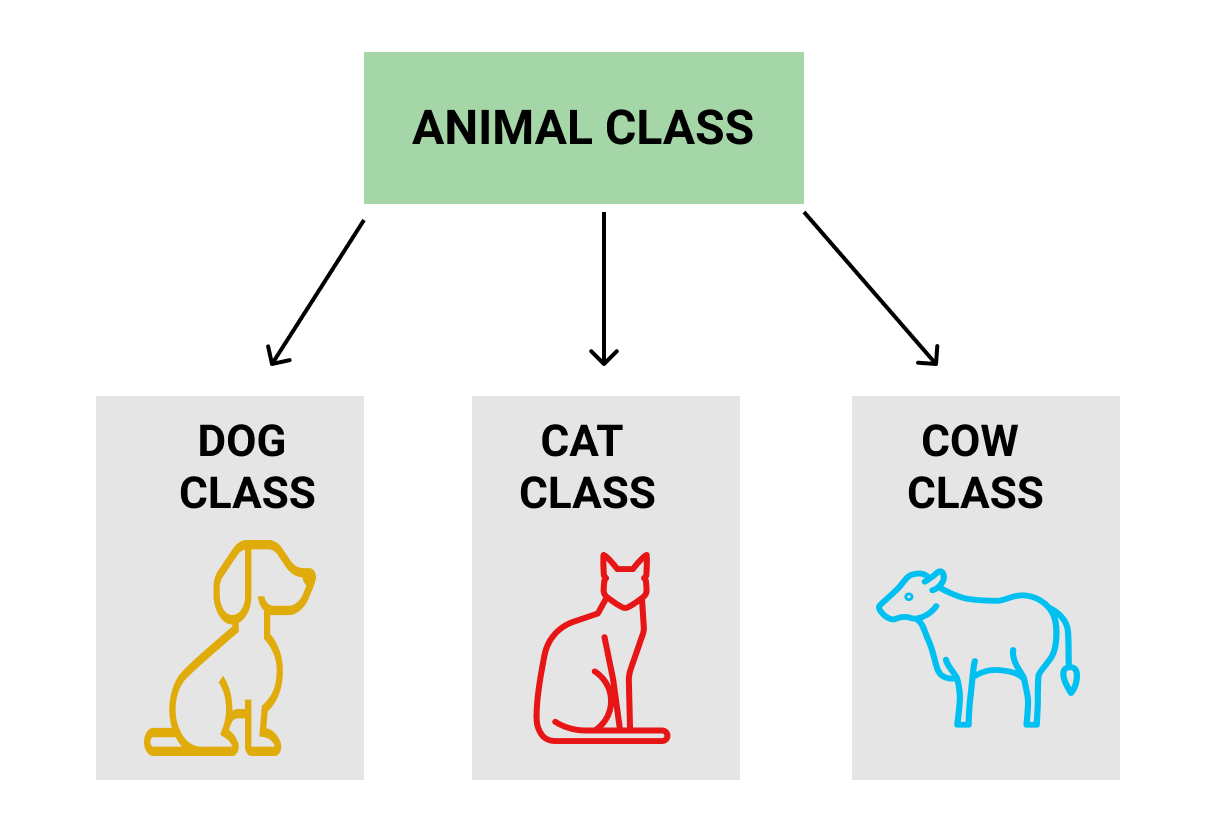
## **Difference: Abstraction and Encapsulation**

|  |  |
| --- | --- |
| ****Abstraction**** | ****Encapsulation**** |
| Abstraction solves the issues at the design level. | Encapsulation solves it implementation level. |
| Abstraction is about hiding unwanted details while showing most essential information. | Encapsulation means binding the code and data into a single unit. |
| Abstraction allows focussing on what the information object must contain | Encapsulation means hiding the internal details or mechanics of how an object does something for security reasons. |

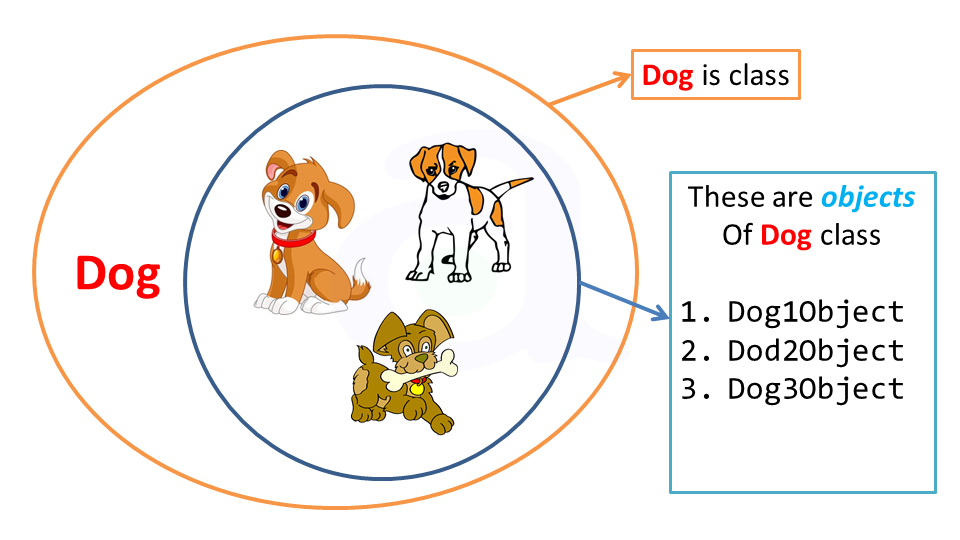
## **Difference : Abstract Class and Interface**

|  |  |
| --- | --- |
| ****Abstract Class**** | ****Interface**** |
| An abstract class can have both abstract and non-abstract methods. | The interface can have only abstract methods. |
| It does not support multiple inheritances. | It supports multiple inheritances. |
| It can provide the implementation of the interface. | It can not provide the implementation of the abstract class. |
| An abstract class can have protected and abstract public methods. | An interface can have only have public abstract methods. |
| An abstract class can have final, static, or static final variable with any access specifier. | The interface can only have a public static final variable. |

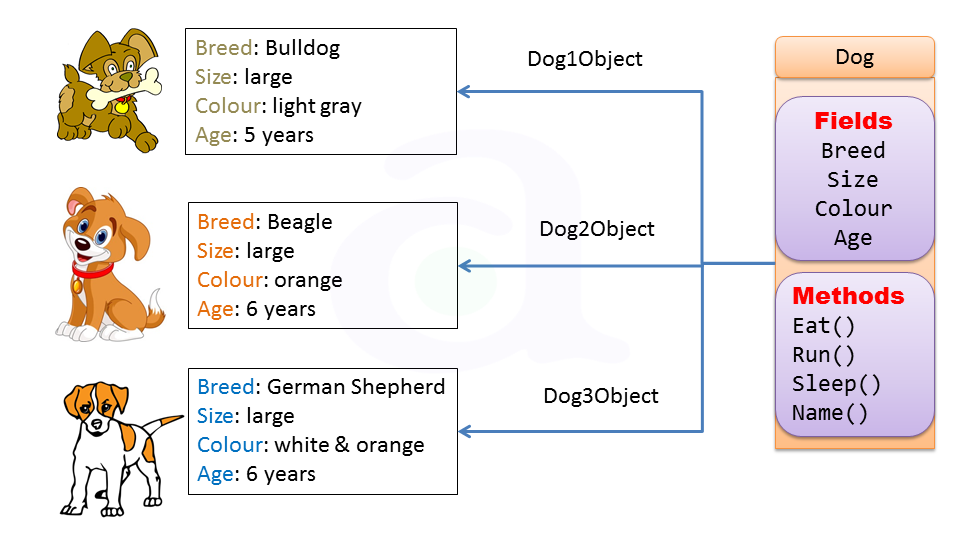
#### CLASS



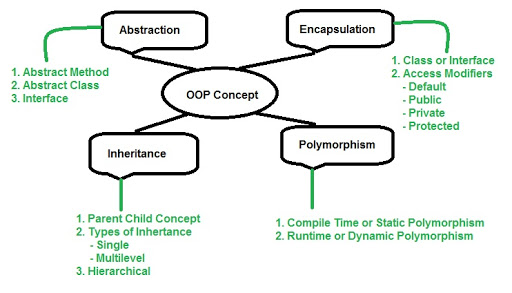
#### OBJECT



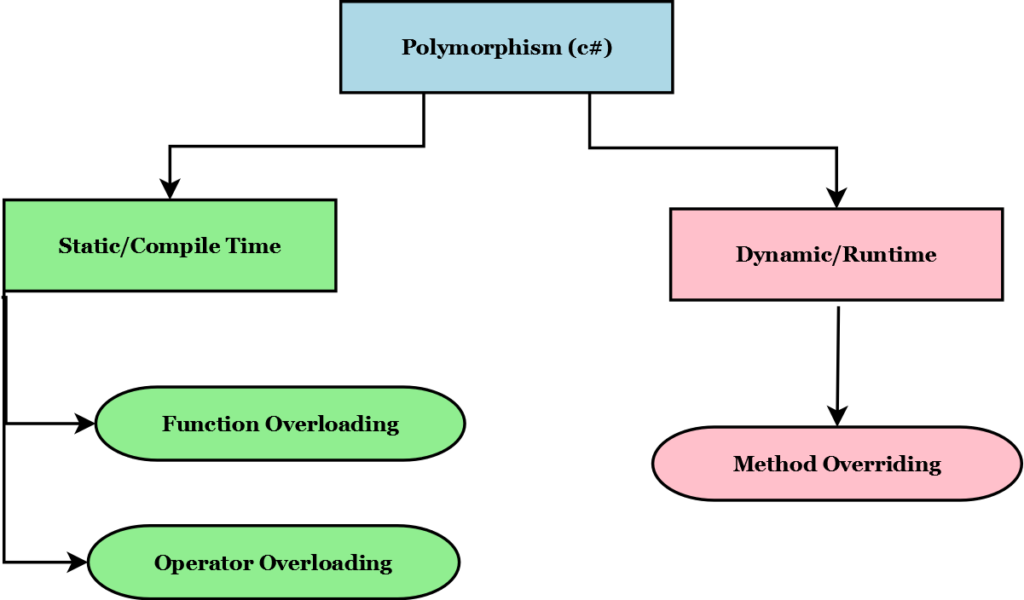
# METHOD



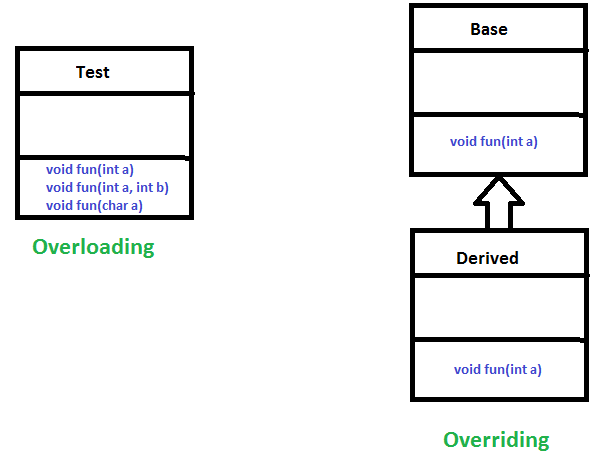
* **OOPS MAIN CONCEPTS**



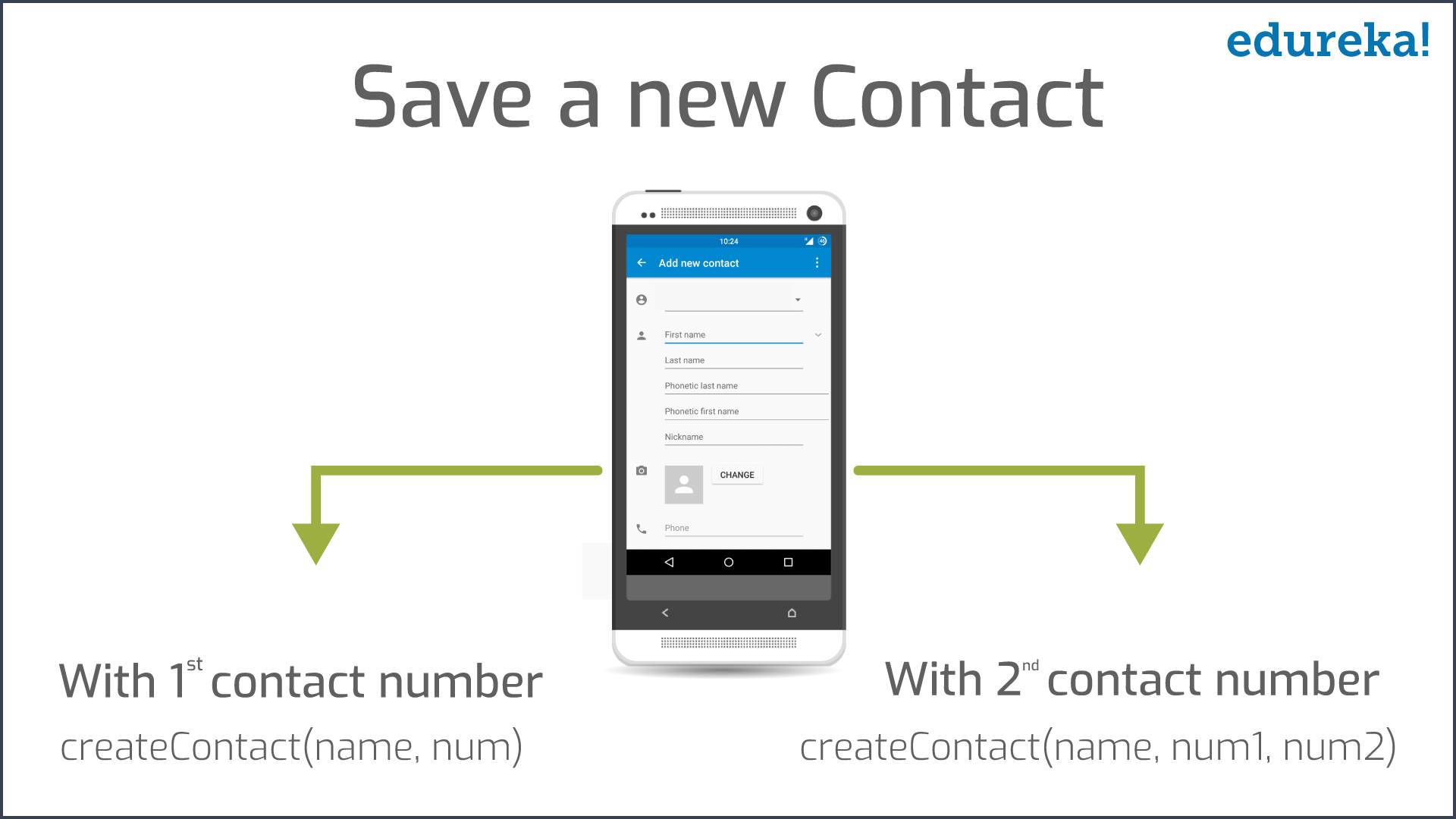
# POLYMORPHISM



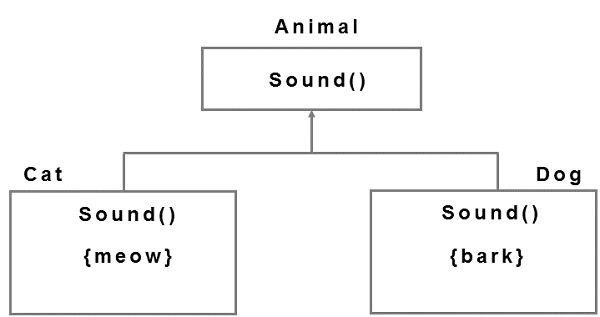
# METHOD OVERLOADING /METHOD OVERRIDING



# METHOD OVERLOADING

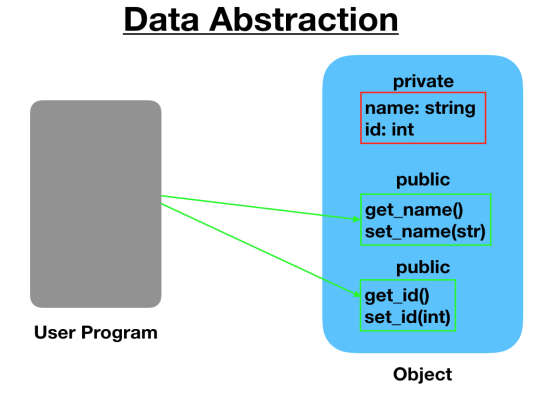
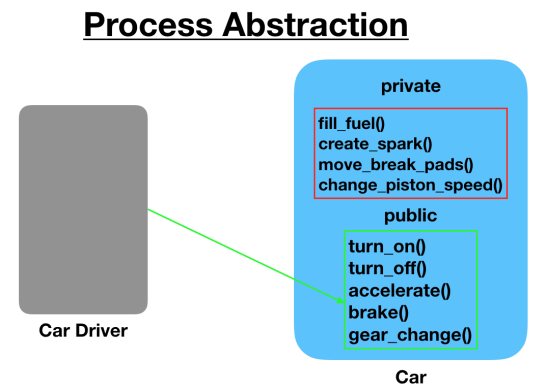


# METHOD OVERRIDING

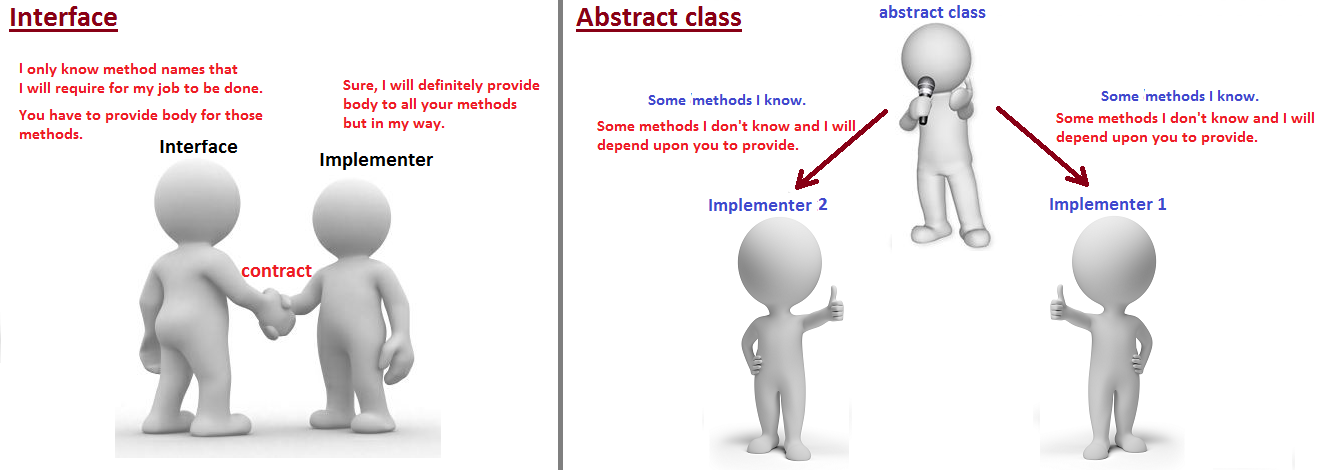


# ABSTRACTION

# 2 Types of Abstraction :

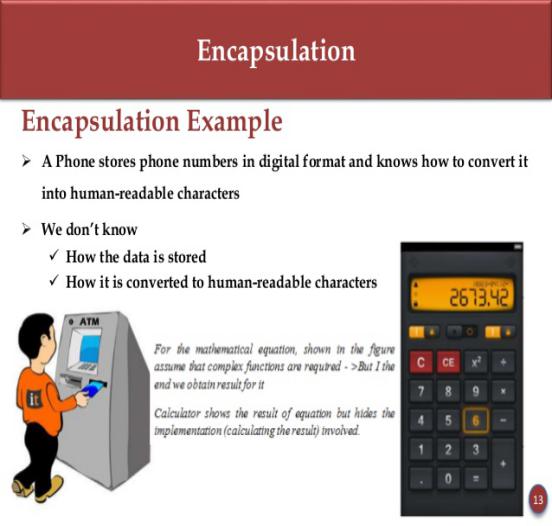
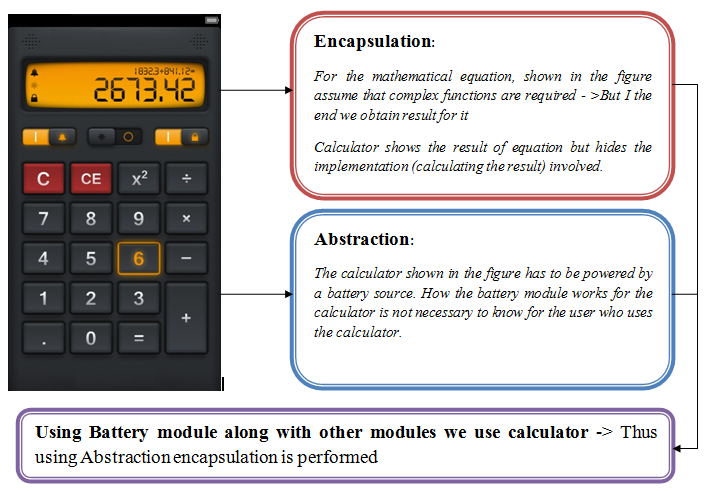
 

# 2 Ways to Implement Abstraction :



# ENCAPSULATION



Encapsulation simply means binding object state(data) and behavior(code) together. If you are creating class, you are doing encapsulation.

# INHERITANCE ( extends)

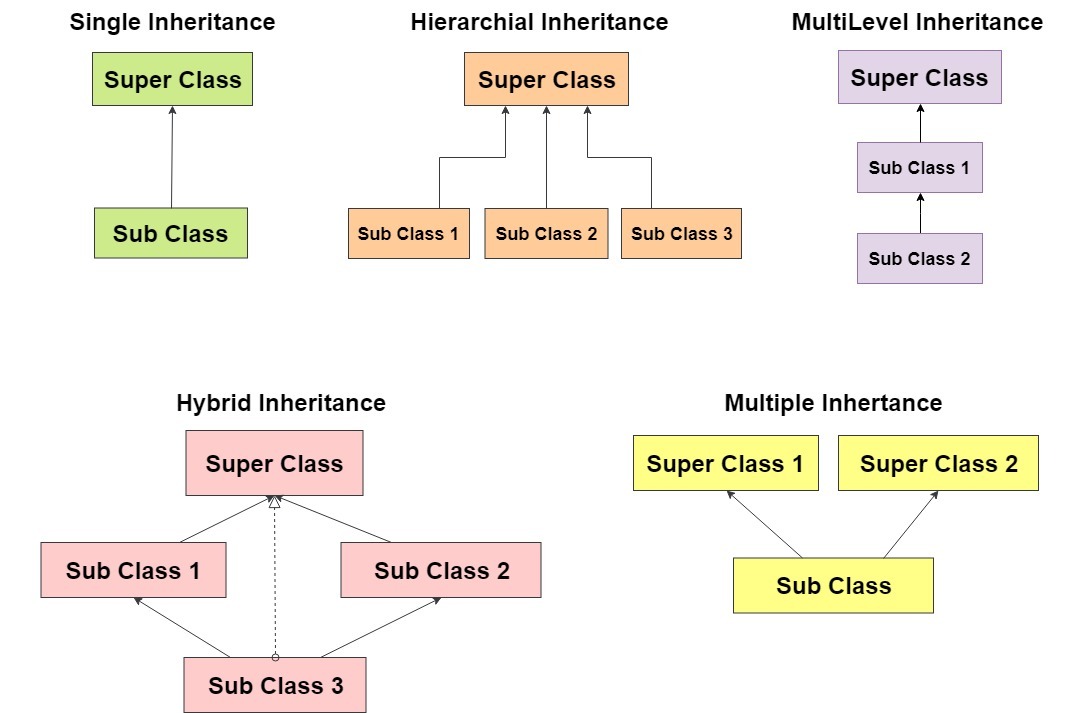
**[Types of Inheritance](https://beginnersbook.com/2013/05/java-inheritance-types/)**:

****Single Inheritance****: refers to a child and parent class relationship where a class extends the another class.

****Multilevel inheritance****: refers to a child and parent class relationship where a class extends the child class. For example class A extends class B and class B extends class C.

****Hierarchical inheritance****: refers to a child and parent class relationship where more than one classes extends the same class. For example, class B extends class A and class C extends class A.

**Multiple Inheritance**: refers to the concept of one class extending more than one classes, which means a child class has two parent classes. **Java doesn’t support multiple inheritance**



# INTERFACE ( implements)

