Principle of OOP

Principle of OOP

1. Class
2. Object
3. Encapsulation
4. Abstraction
5. Inheritance
6. Polymorphism
7. Dynamic Binding
8. Message Passing

<u>Class & Object</u>

Concept of Classes and Objects

- Class is a blueprint of an Object
- Class is a description of Object's property set and set of operations
- Creating class is as good as defining a new data type
- Class is a means to achieve encapsulation
- Object is a run time entity
- Object is an instance of a class

Class

1. A class is a blueprint for the object.
2. The mechanism that allows you to combine data members and the member function in a single unit is called a class.
3. Class is a user defined data type.
4. Class describes both the properties (data) and behaviors (functions) of objects.
5. Classes are not objects, but they are used to instantiate objects.
6. No memory is allocated at the time of declaration

Syntax:

```
class class_name
{
   Data Members;
   Member functions/Methods;
};
```

Object

- A class variable is called object or instance.
- Object is a real world entity.
- An Object in C++ has two characteristics:
    - State
    - Behavior
- Each object has different data variables but they share the member functions.
- Sufficient memory space will be allocated for all the variables/object of class at the time of declaration.

Syntax:

Class_name object_name1, object_name2;

Example of Class and Object

Vechicle class

Example of Class and Object

We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house.

- House is the object.

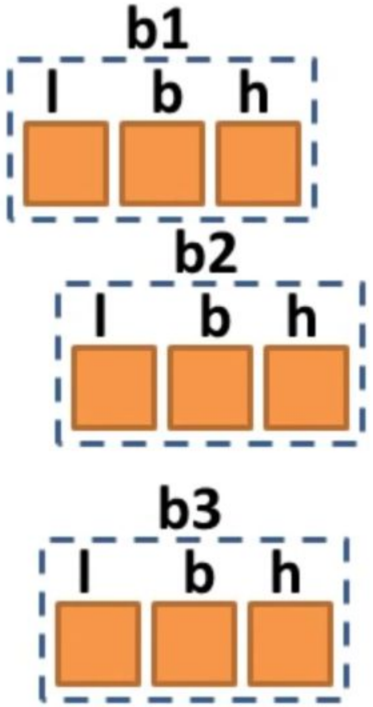
As, many houses can be made from the same description, we can create many objects from a class.

Class vs Object

```
class box
{
   int l,b,h;
   void setDimension(int x,int y, int z)
   {...}
   void showDimension()
   {...}
};
```

Class vs Object

```
class box
{
   int l,b,h;
   void setDimension(int x,int y, int z)
   {...}
   void showDimension()
   {...}
};
```



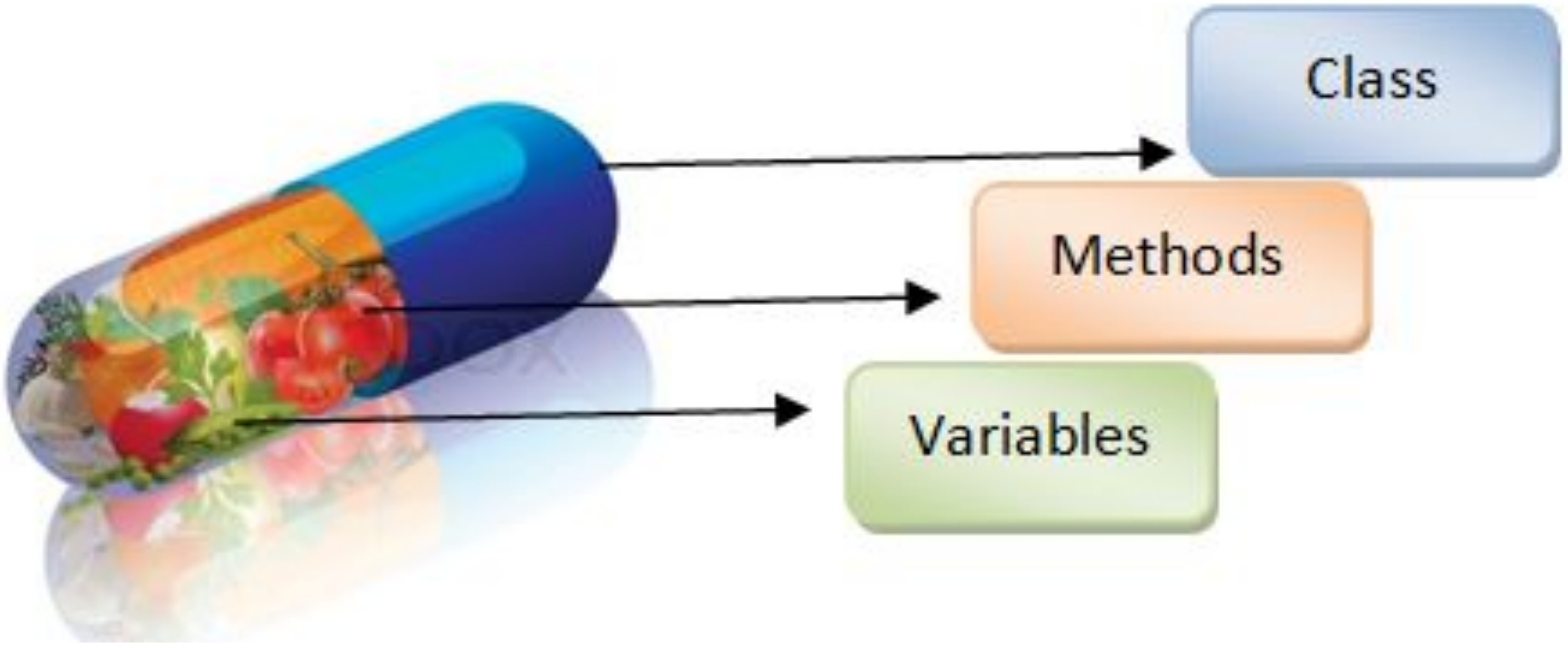
- box b1;
- box b2;
- box b3;

Encapsulation

Encapsulation

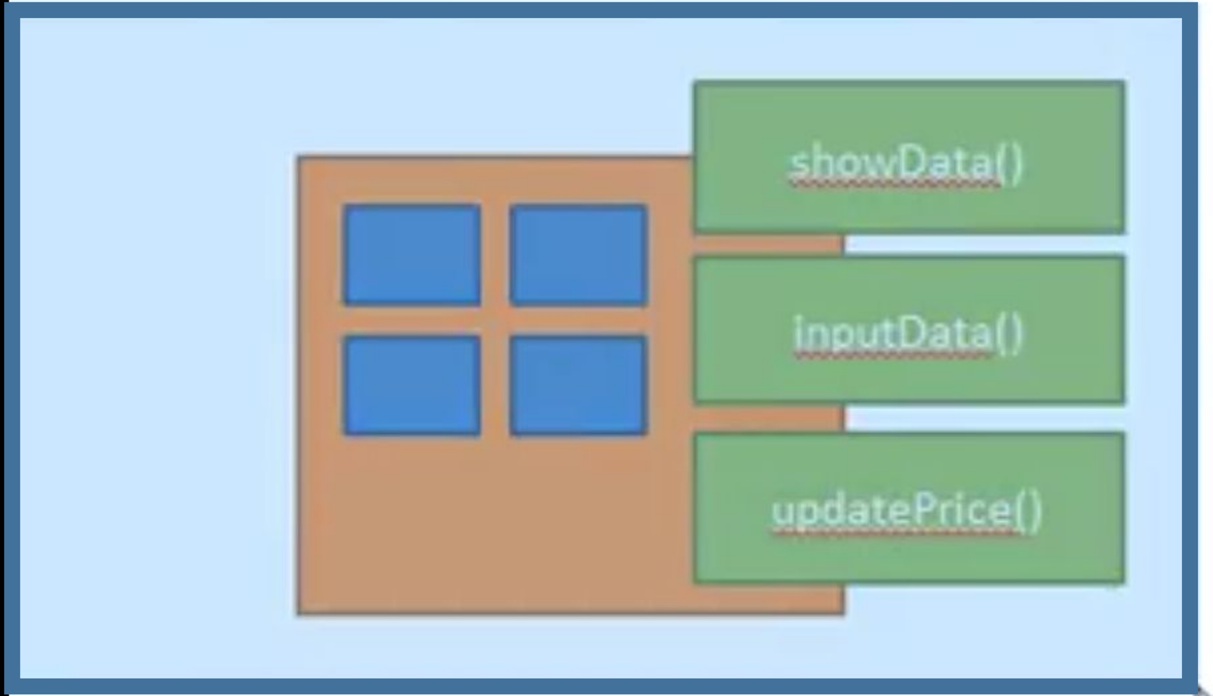
- The wrapping up of data and functions into a single unit (called class) is known as encapsulation.
- The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it.
- These function provide the interface between the object's data and the program.
- This insulation of the data from direct access by the program is called **Data Hiding** or **Information Hiding**

Encapsulation



Encapsulation

- An act of combining properties and methods, related to the same object, is known as Encapsulation



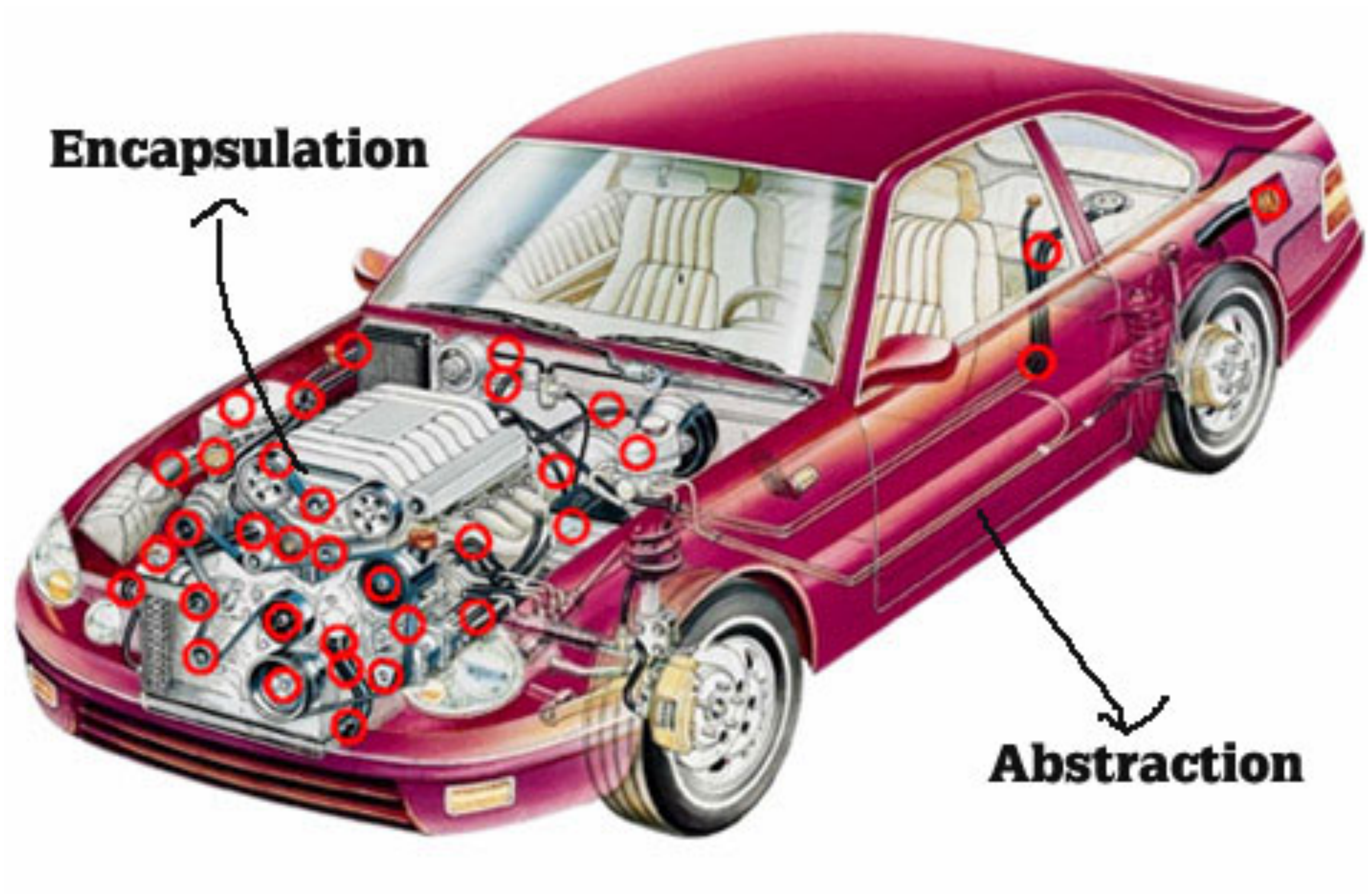
Why Encapsulation?

- Object becomes equipped with sufficient information set and set of operations.
- Any system can be assumed as a collection of objects.
- These objects are capable to interact with each other using various methods

Abstraction

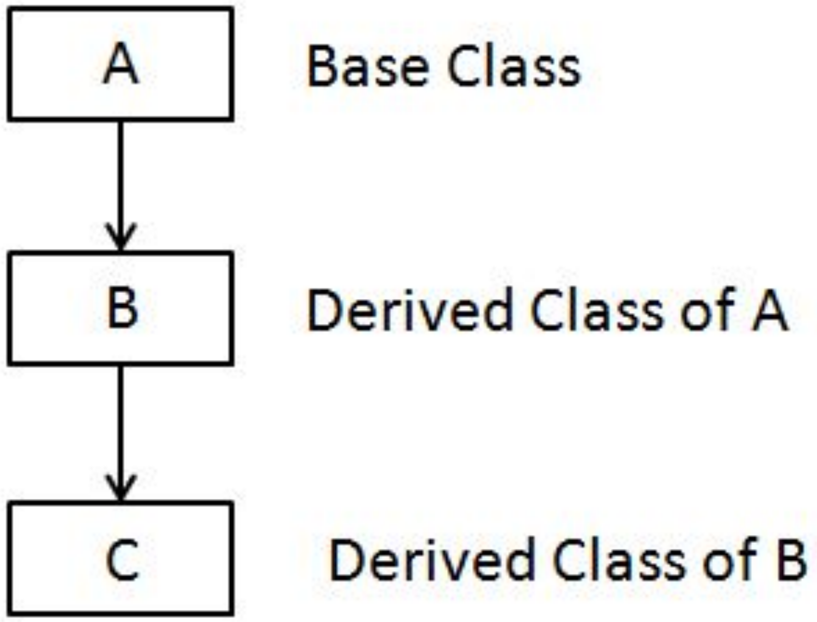
Abstraction

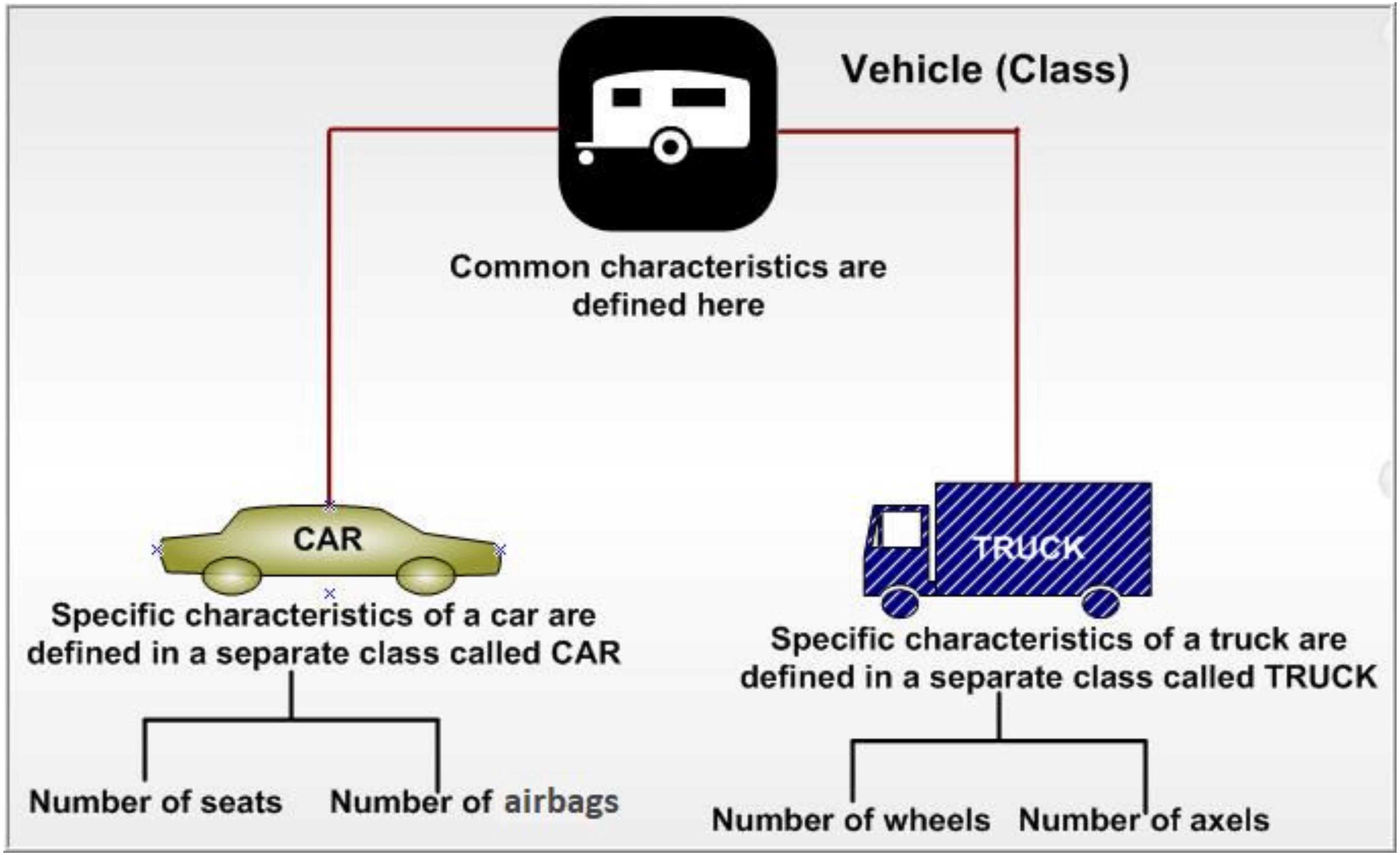
- It refers to the act of representing essential features without including the background details or explanations.
- Can change internal implementation of class independently without affecting the user.
- Helps to increase security of an application or program as only important details are provided to the user.
- Eg. Car, Fan – Switch, Mobile Phone etc


Encapsulation
Abstraction

Inheritance

Inheritance in Object Oriented Programming can be described as a process of creating new classes from existing classes. New classes **inherit** some of the properties and behavior of the existing classes. An existing class that is "parent" of a new class is called a base class. New class that inherits properties of the base class is called a **derived class**



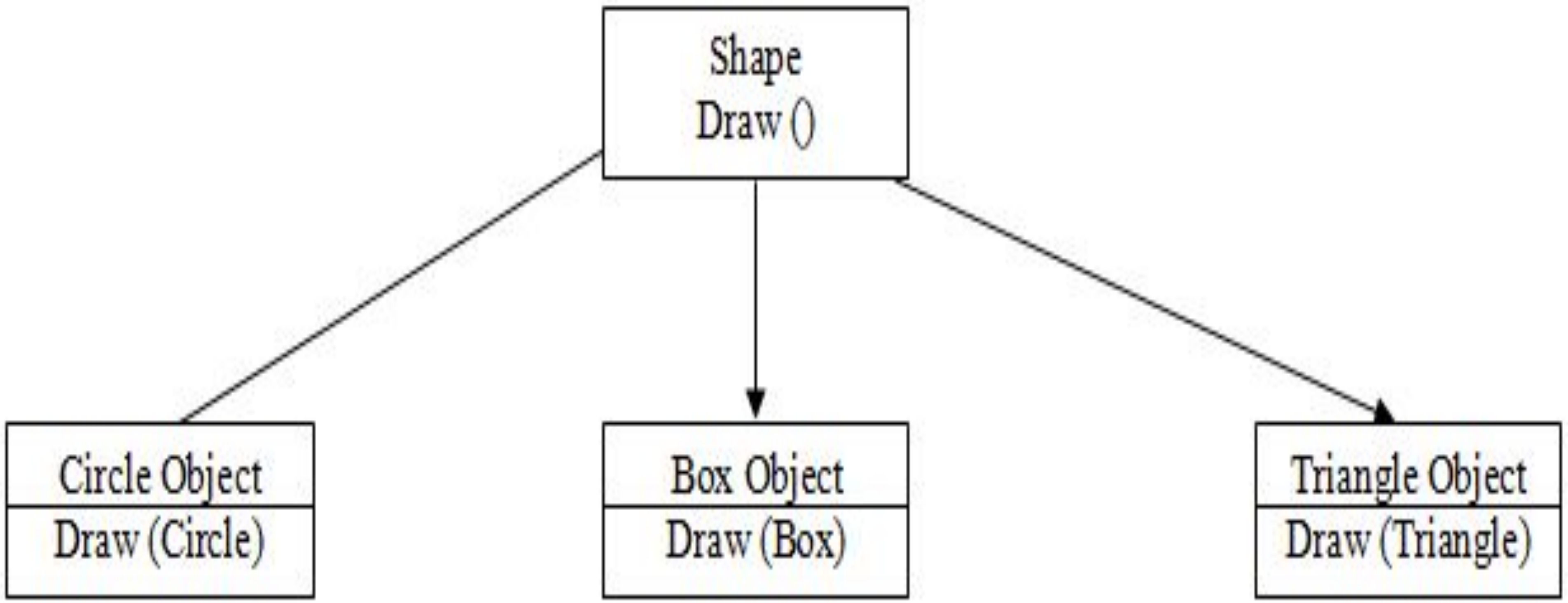

Vehicle (Class)
Common characteristics are defined here
CAR
Specific characteristics of a car are defined in a separate class called CAR
Number of seats
Number of airbags
TRUCK
Specific characteristics of a truck are defined in a separate class called TRUCK
Number of wheels
Number of axels

Polymorphism

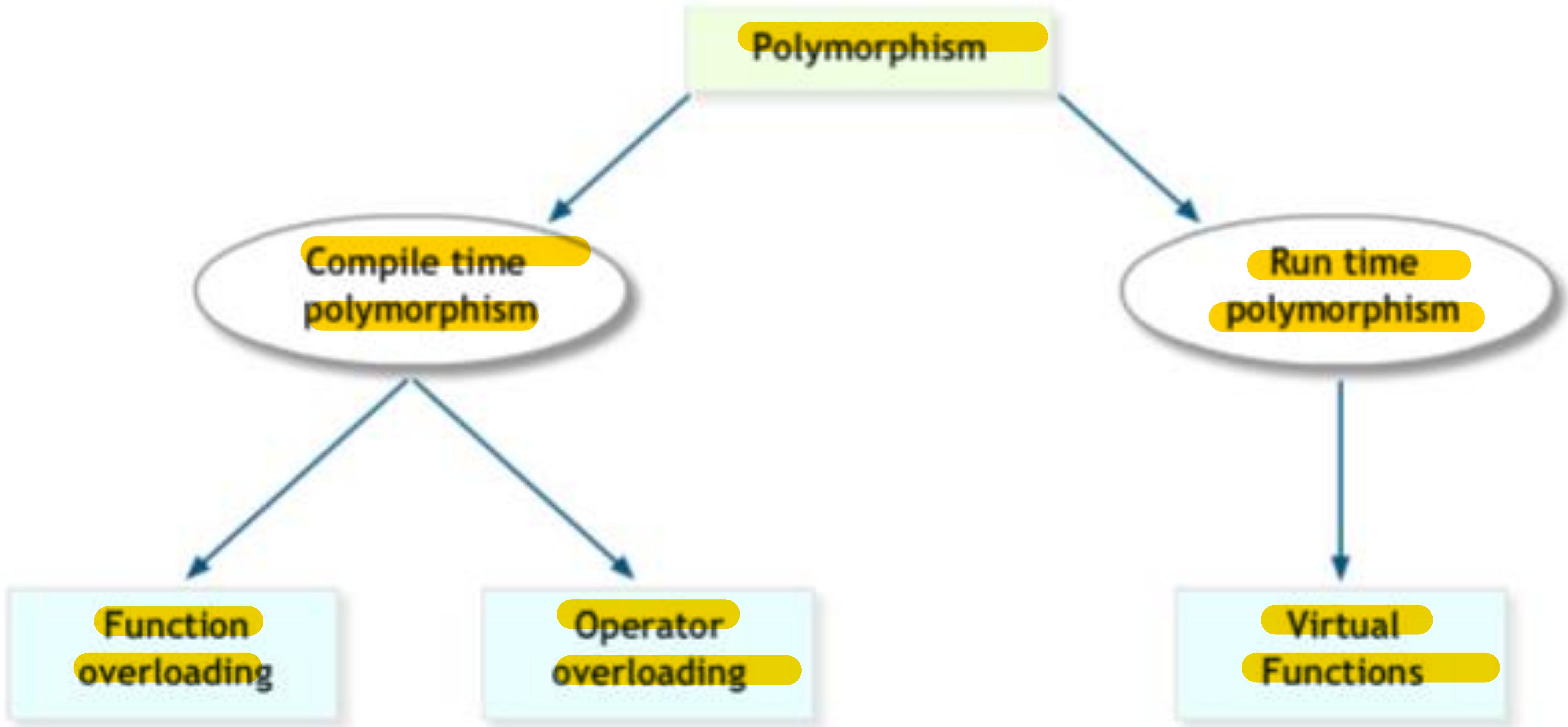
Polymorphism

- Poly means **many** and morphs means **forms.** So polymorphism is the ability to take more than one form.
- An operation may exhibit different behaviours in different instances. The behaviour depends upon the type of data used in the operation.

Polymorphism



Types of Polymorphism

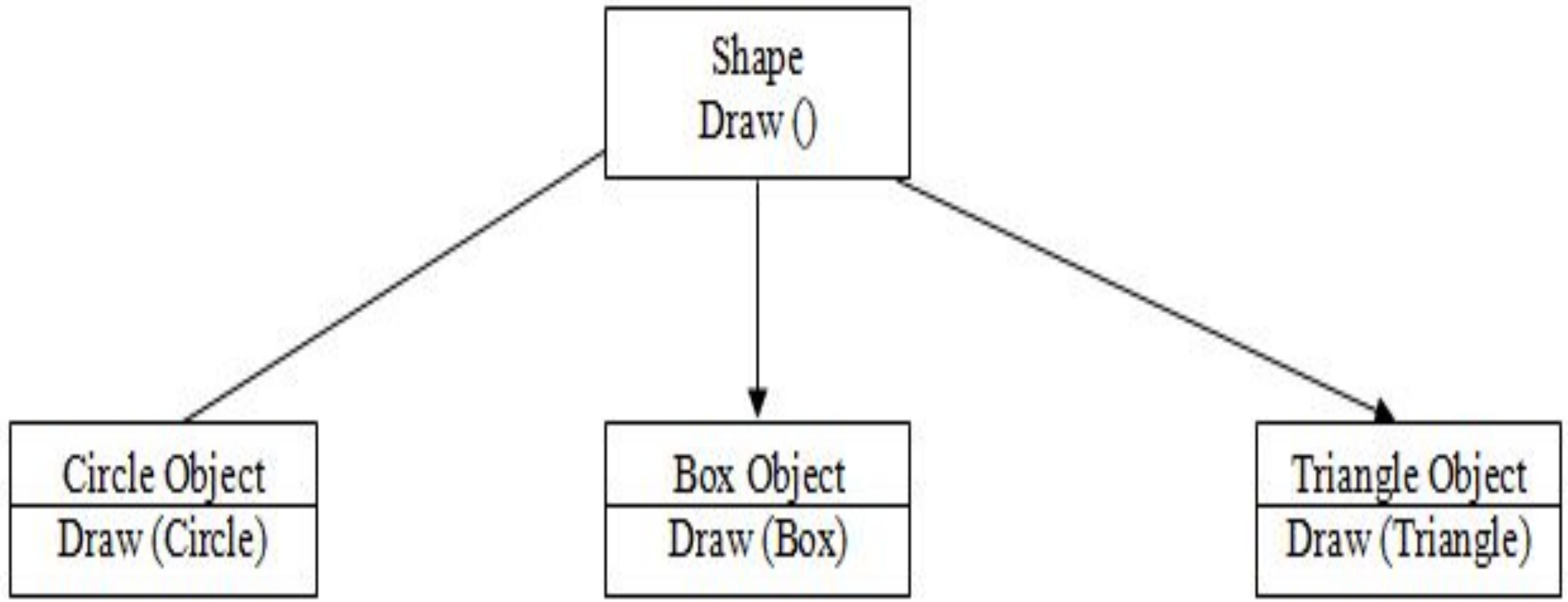


Dynamic Binding

Dynamic Binding

- Binding refers to the linking of a procedure call to the code to be executed in response to the call.
- Dynamic binding is also known as Late Binding, means that the code associated with a given procedure call is not known until the call at run-time.
- It is associated with Polymorphism & Inheritance.

Dynamic Binding


Shape
Draw ()
Circle Object
Draw (Circle)
Box Object
Draw (Box)
Triangle Object
Draw (Triangle)