DAY-69
#100DAYSRTL

"System Verilog:- OOPs Introduction"

"**Introduction**":-

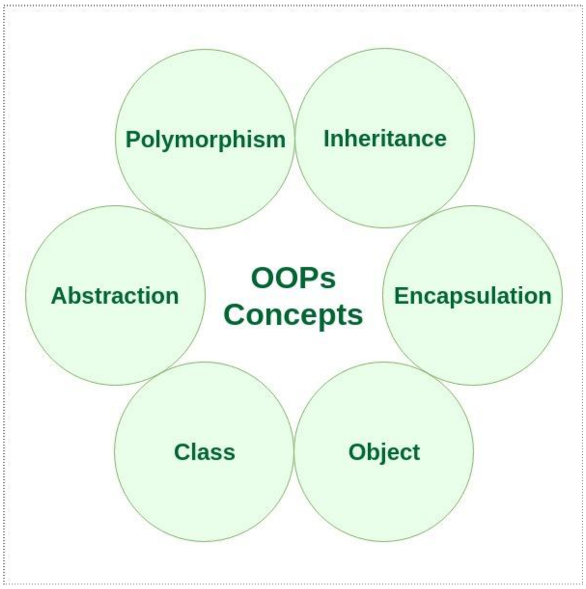
In System Verilog, a language used for complex design testing, they added Object-Oriented Programming (OOP) concepts. This means they included a way to build components that can inherit characteristics from other components and have more flexible behaviors. In traditional Verilog, the components, called modules, couldn't do this, making it harder to reuse them efficiently for complex tests.

"**OOPs**":-

- ✓ System Verilog, primarily an HDL for hardware design, incorporates certain Object-Oriented Programming (OOP) features but not all.
- ✓ OOP, inspired by real-world objects, simplifies coding by resembling real objects. Key OOP features include Inheritance, Polymorphism, Encapsulation, and Data Abstraction, found in languages like C++, C#, and Java.
- ✓ In System Verilog, supported OOP elements include Single and Multi-level Inheritance, Function Overriding, Virtual Classes, and Virtual Functions.
- ✓ These features allow defining blueprints called "classes" that specify an object's properties and behaviors.

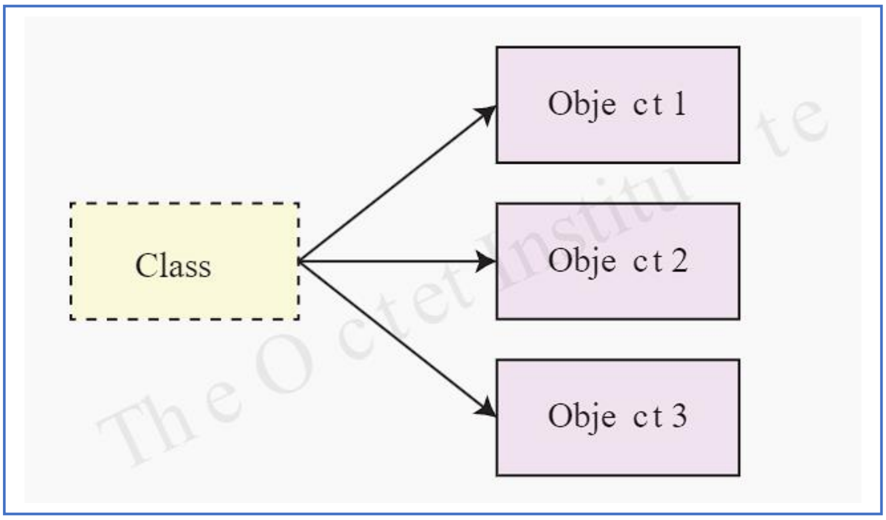
For example, a car, an object created from a car blueprint (class), shares common attributes like wheels, brakes, and steering. Yet, individual cars have unique features like color and configuration.

Exploring these OOP features through real-life examples, such as cars and their varying properties, helps us understand how classes define objects and their distinct characteristics.



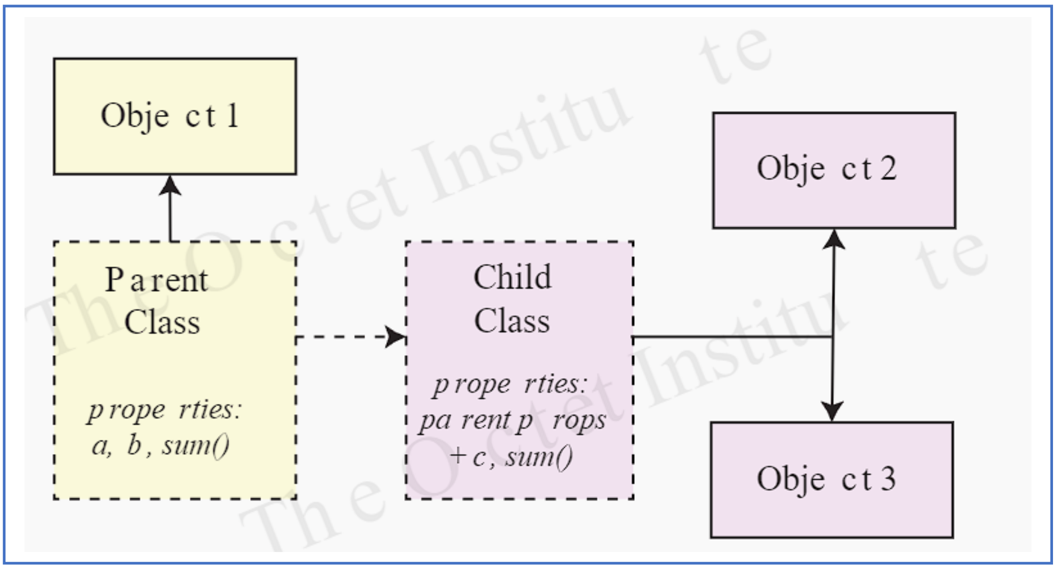
"**Class and Objects**":-

- ✓ Class is a blueprint that defines the properties and behavior of an object. In OOP classes are the entity that encapsulates all the data and methods.
- ✓ Objects are the unique entity created from class. Objects are dynamic in nature, i.e., it is created dynamically during runtime.
- ✓ Parent class can be known as base class or super class. Similarly, child class is known as derived class and sub class.



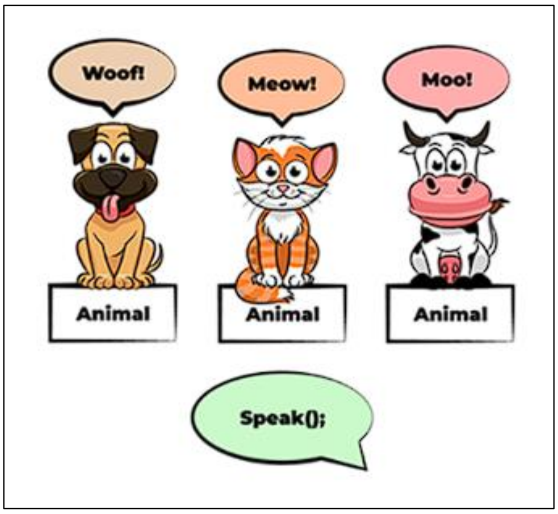
“**<u>Inheritance</u>**”:-

This is the property of OOP by virtue of which a class can inherit properties and behaviour of another class called parent or base class. The child or derived class can add more property or behaviour on the base class.



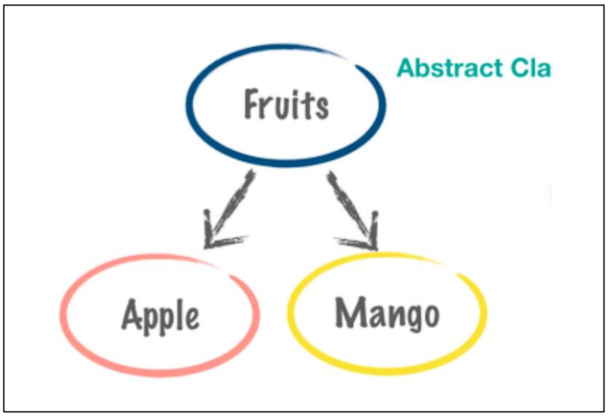
“**<u>Polymorphism</u>**”:-

Poly means many and morph means form. Thus, polymorphism is a concept in which the same method can act differently in a child class or when inputs are different. OOPs provide us with two ways to enable polymorphism – function overloading and function overriding.



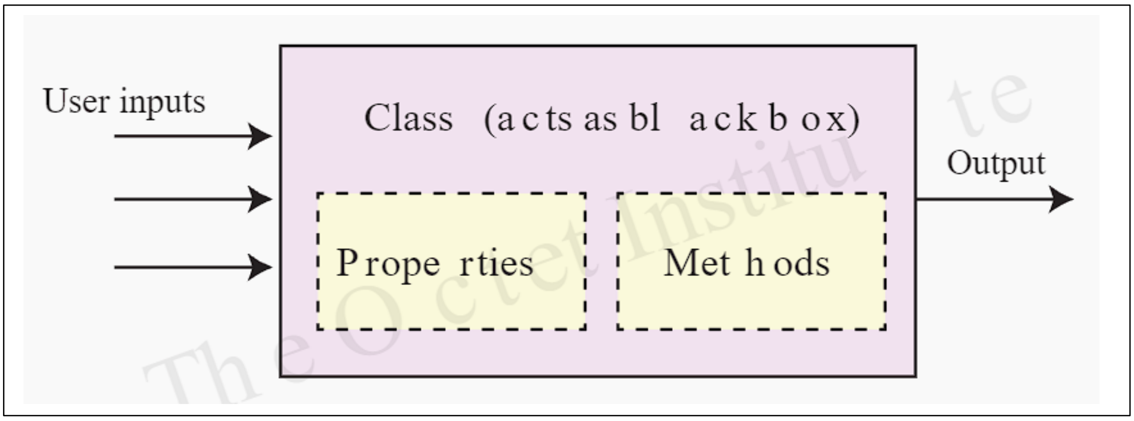
“**Abstraction**”:-

Data Abstraction means hiding unnecessary and representing only what is necessary for the user basically that particular use case.



“**Encapsulation**”:-

This is a property by which we can bundle all the data and methods into one unit. This also helps in black boxing a unit where users can focus more on using the unit without knowing the underlying process or complexity.


User inputs
Class (acts as black box)
Properties
Methods
Output