Object-Oriented Basics

Object-oriented programming (OOP) is a style of programming that focuses on using objects to design and build applications. Contrary to procedure-oriented programming where programs are designed as blocks of statements to manipulate data, OOP organizes the program to combine data and functionality and wrap it inside something called an “Object”.

If you have never used an object-oriented programming language before, you will need to learn a few basic concepts before you can begin writing any code. This chapter will introduce some basic concepts of OOP:

- **Objects:** Objects represent a real-world entity and the basic building block of OOP. For example, an Online Shopping System will have objects such as shopping cart, customer, product item, etc.
- **Class:** Class is the prototype or blueprint of an object. It is a template definition of the attributes and methods of an object. For example, in the Online Shopping System, the Customer object will have attributes like shipping address, credit card, etc., and methods for placing an order, canceling an order, etc.

OOP basics

The four principles of object-oriented programming are encapsulation, abstraction, inheritance, and polymorphism.

- **Encapsulation:** Encapsulation is the mechanism of binding the data together and hiding it from the outside world. Encapsulation is achieved when each object keeps its state private so that other objects don’t have direct access to its state. Instead, they can access this state only through a set of public functions.
- **Abstraction:** Abstraction can be thought of as the natural extension of encapsulation. It means hiding all but the relevant data about an object in order to reduce the complexity of the system. In a large system, objects talk to each other, which makes it difficult to maintain a large code base; abstraction helps by hiding internal implementation details of objects and only revealing operations that are relevant to other objects.
- **Inheritance:** Inheritance is the mechanism of creating new classes from existing ones.

- **Polymorphism:** Polymorphism (from Greek, meaning “many forms”) is the ability of an object to take different forms and thus, depending upon the context, to respond to the same message in different ways. Take the example of a chess game; a chess piece can take many forms, like bishop, castle, or knight and all these pieces will respond differently to the ‘move’ message.