Object-Oriented Programming is a programming paradigm that organizes code into objects that contain both data and behavior. Here's an explanation of the key concepts:

**Object**

An object is a concrete instance that contains both data (attributes/properties) and code (methods/functions). Think of it like a real-world object - a car has properties (color, model, speed) and behaviors (accelerate, brake, turn).

**Class**

A class is a blueprint or template for creating objects. It defines what attributes and methods the objects of that type will have. For example, a Car class defines the structure that all car objects will follow.

Inheritance

This allows a class to inherit attributes and methods from another class. It enables code reuse and establishes a relationship between parent (base) and child (derived) classes. For example, an ElectricCar class could inherit from a Car class, adding specific features while keeping the basic car functionality.

**Polymorphism**

This concept allows objects of different classes to respond to the same method call in different ways. It comes in two main forms:

**Method overriding:** A child class can provide a different implementation of a method defined in its parent class

**Method overloading:** Multiple methods can have the same name but different parameters

**Abstraction**

This principle involves hiding complex implementation details and showing only the necessary features of an object. It helps manage complexity by letting you focus on what an object does rather than how it does it.

**Encapsulation**

This involves bundling data and the methods that operate on that data within a single unit (class), and restricting access to internal details. It helps protect data integrity and reduces system complexity.