Classes:

- Là bản thiết kế cho một loại đối tượng cụ thể, giống như bản vẽ blueprint của ngôi nhà. Nó xác định các thuộc tính (properties) và phương thức (methods) chung cho tất cả các đối tượng thuộc lớp đó.
- Ví dụ: Class `Car` có thể có các thuộc tính như `color`, `model`, `speed` và các phương thức như `startEngine`, `accelerate`, `brake`.

Methods:

- Là những khối mã thực hiện các hành động cho một đối tượng. Chúng xác định cách thức đối tượng tương tác với thế giới xung quanh.
- Các phương thức có thể truy cập và cập nhật các thuộc tính của đối tượng.
- Ví dụ: Phương thức `startEngine` của Class `Car` sẽ khởi động động cơ xe.

Objects:

- Là một phiên bản cụ thể của một lớp.
- Mỗi đối tượng có một bộ giá trị riêng cho các thuộc tính của lớp.
- Ví dụ: Có thể có nhiều đối tượng `Car` khác nhau, mỗi chiếc có màu sắc, kiểu dáng và tốc độ khác nhau.

Encapsulation:

- Là việc đóng gói dữ liệu (thuộc tính) và hành vi (phương thức) của một đối tượng vào một đơn vị thống nhất.
- Nó giúp kiểm soát việc truy cập vào dữ liệu của đối tượng, đảm bảo tính an toàn và bảo mật.
- Ví dụ: Thuộc tính `fuelLevel` của Class `Car` có thể được đặt thành private, nghĩa là chỉ các phương thức bên trong Class `Car` mới có thể thay đổi giá trị của nó.

Information Hiding:

- Là một phần của encapsulation, tập trung vào việc ẩn các chi tiết bên trong của một đối tượng, chỉ cung cấp cho người dùng giao diện cần thiết để tương tác với đối tượng.
- Nó đơn giản hóa việc sử dụng đối tượng và ngăn ngừa việc sửa đổi nội bộ không mong muốn.
- Ví dụ: Thay vì cho người dùng truy cập trực tiếp vào thuộc tính `engineTemperature` của Class `Car`, ta có thể cung cấp phương thức `getEngineTemperature` để đọc giá trị an toàn.

Polymorphism:

- Là khả năng của một đối tượng thể hiện nhiều hành vi khác nhau tùy theo tình huống.

- Nó cho phép xử lý các đối tượng thuộc các lớp khác nhau theo cùng một cách, làm cho mã code linh hoạt hơn.
- Ví dụ: Phương thức `print` có thể được sử dụng cho nhiều loại đối tượng khác nhau, in ra nội dung phù hợp với từng loại.

Inheritance:

- Là cơ chế cho phép một lớp (con) kế thừa các thuộc tính và phương thức của một lớp khác (cha).
- Nó giúp tái sử dụng mã code và tránh lặp lại code thừa, tăng tính hiệu quả và bảo trì code.
- Ví dụ: Class `Truck` có thể kế thừa từ Class `Car`, thừa hưởng các thuộc tính và phương thức chung như `color`, `model`, `startEngine`, `accelerate`, `brake`, đồng thời thêm các thuộc tính và phương thức riêng mới như `cargoCapacity`, `unloadCargo`.

Abstraction:

- Là việc tập trung vào các đặc điểm và hành vi thiết yếu của một đối tượng, bỏ qua các chi tiết bên trong phức tạp.
- Nó giúp đơn giản hóa giao diện của đối tượng, khiến code dễ hiểu và sử dụng hơn.
- Ví dụ: Người dùng xe chỉ cần biết cách điều khiển các tính năng chính như khởi động, tăng tốc, phanh mà không cần hiểu cơ chế bên trong của động cơ hay hệ thống điện.