1. Abstracción:
    - Definición: La abstracción se refiere a la capacidad de enfocarse en las características esenciales de un objeto mientras se ignoran los detalles irrelevantes. Es como crear un modelo simplificado de algo del mundo real.
    - Ejemplo: Un automóvil. Al pensar en un automóvil desde una perspectiva abstracta, nos enfocamos en sus características principales como el motor, las ruedas, el volante y su función de transporte. Ignoramos detalles como el color, la marca, el modelo o los componentes internos del motor.
2. Encapsulamiento:
    - Definición: El encapsulamiento es la agrupación de datos (atributos) y métodos (comportamientos) relacionados dentro de una unidad única (clase o objeto). Oculta los detalles internos de implementación y solo expone una interfaz pública para interactuar con el objeto.
    - Ejemplo: Una cuenta bancaria. La información de la cuenta como el número de cuenta, el saldo y el nombre del titular está encapsulada dentro de la clase "CuentaBancaria". Los métodos para depositar, retirar y consultar el saldo están disponibles a través de la interfaz pública, mientras que los detalles de cómo se almacena y procesa la información de la cuenta se ocultan.
3. Herencia:
    - Definición: La herencia es la capacidad de una clase (clase hija) de heredar atributos y métodos de otra clase (clase padre). Permite la reutilización de código y la creación de jerarquías de clases que reflejan relaciones entre objetos del mundo real.
    - Ejemplo: Animales. La clase "Animal" puede tener atributos como nombre, especie y edad. Las clases "Perro" y "Gato" pueden heredar de "Animal" y agregar atributos específicos como raza y pelaje. Los métodos para comer, dormir y moverse pueden definirse en "Animal" y heredarse por las clases hijas, o redefinirse específicamente para cada animal.
4. Polimorfismo:
    - Definición: El polimorfismo es la capacidad de que diferentes objetos respondan al mismo mensaje de manera diferente. Permite que las clases hijas sobrescriban o implementen métodos heredados de la clase padre, proporcionando comportamientos específicos para cada tipo de objeto.
    - Ejemplo: Formas geométricas. Un método "calcularÁrea()" puede definirse en una clase base "Forma". Las clases derivadas como "Triángulo", "Círculo" y "Rectángulo" pueden sobrescribir este método para implementar el cálculo del área específico para cada forma.
5. Clases y Objetos:
    - Clases: Son las plantillas o blueprints que definen las características (atributos y métodos) de un tipo de objeto.
    - Objetos: Son instancias individuales de una clase, creadas a partir de la plantilla. Cada objeto tiene sus propios valores de atributos y puede utilizar los métodos definidos en la clase.

Ejemplo: Clase "Vehículo" con atributos como marca, modelo y color. Un objeto "miAuto" de la clase "Vehículo" tendría valores específicos para esos atributos y podría usar métodos como encender, apagar y conducir.

6. Métodos y Atributos:

- Métodos: Son las acciones o comportamientos que un objeto puede realizar. Se definen dentro de la clase y se invocan desde los objetos.
- Atributos: Son las características o propiedades que describen el estado de un objeto. Se almacenan dentro del objeto y se puede acceder a ellos o modificarlos utilizando métodos.

Ejemplo: En la clase "CuentaBancaria", el método "depositar(cantidad)" aumenta el saldo de la cuenta, mientras que el atributo "saldo" almacena la cantidad actual de dinero en la cuenta.

7. Modularidad:
   - Definición: La modularidad se refiere a la organización del código en módulos independientes y bien definidos. Cada módulo encapsula una unidad funcional específica y tiene interfaces claras para interactuar con otros módulos.
   - Ejemplo: Un sistema de gestión de pedidos en línea puede dividirse en módulos para gestión de usuarios, administración de productos, procesamiento de pedidos y pagos. Cada módulo se desarrolla y mantiene de forma independiente, pero interactúa con los demás a través de interfaces definidas.
8. Reusabilidad:
   - Definición: La reusabilidad se refiere a la capacidad de utilizar código existente en diferentes partes de un programa o en diferentes proyectos. Esto reduce el tiempo de desarrollo y mejora la calidad del código.
   - Ejemplo: Las clases y métodos bien diseñados pueden reutilizarse en diferentes aplicaciones. Por ejemplo, una clase "ConexiónBD" que maneja la conexión a una base de datos puede usarse en varios proyectos que requieren acceso a datos.