1. Abstracción: La abstracción es el proceso de representar objetos o conceptos de manera simplificada, enfocándose en los aspectos relevantes y omitiendo detalles irrelevantes. Esto permite modelar sistemas complejos de forma más manejable.

   Ejemplo practico:

```
public abstract class Vehiculo {
   public abstract void arrancar();
   public abstract void detener();
}

public class Auto extends Vehiculo {
      public void arrancar() {
       System.out.println("El auto arranca.");
   }

       public void detener() {
       System.out.println("El auto se detiene.");
   }
}
```

2. Herencia: La herencia permite crear clases que reutilizan, extienden y modifican el comportamiento definido en otras clases. Una clase derivada hereda todos los miembros de la clase base y puede agregar nuevos o sobrescribir los existentes.

```
public class Auto extends Vehiculo {
   private int velocidad;

   public Auto(int velocidad) {
      this.velocidad = velocidad;
   }


   public void arrancar() {
      System.out.println("El auto arranca.");
   }
```

```
    public void detener() {
        System.out.println("El auto se detiene.");
    }

    public void acelerar() {
        System.out.println("El auto acelera a " + velocidad + " km/h.");
    }
}
```

3. Encapsulamiento: El encapsulamiento implica reunir datos y métodos dentro de una estructura, ocultando la implementación del objeto y permitiendo el acceso solo a través de métodos definidos. Esto protege la integridad de los datos y facilita la modificación del código sin afectar a otros componentes.

```
public class Persona {
    private String nombre;
    private int edad;

    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
```

```
    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        if (edad >= 0) {
            this.edad = edad;
        }
    }
}
```

4. Polimorfismo: El polimorfismo permite que objetos de diferentes clases puedan ser tratados como si fueran de una clase común, siempre que respondan a los mismos mensajes (métodos). Esto facilita la creación de código flexible y extensible.