Es un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representan una instancia de alguna clase, y cuyas clases son, todas ellas miembros de una jerarquía de clases unidas mediante relaciones de herencia.

Ejemplo: Diríamos que el coche es el elemento principal que tiene una serie de características, como podrían ser el color, el modelo o la marca. Además tiene una serie de funcionalidades asociadas, como pueden ser ponerse en marcha, parar o aparcar.

La relación de herencia es una relación entre clases que comparten su estructura y el comportamiento. Se denomina herencia simple: Cuando una clase comparte la estructura y comportamiento de una sola clase. Se denomina herencia múltiple: Cuando una clase comparte la estructura y comportamiento de varias clases

Los atributos son las características que tiene un objeto en un auto puede ser:

Marca, modelo, placa

Y los métodos son las funciones que tiene el objeto

Arrancar, frenar, poner cambios, encender la radio etc

Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

**4 Principios de la Programación Orientada a Objetos**

**Encapsulación**

La encapsulación contiene toda la información importante de un objeto dentro del mismo y solo expone la información seleccionada al mundo exterior.

Esta propiedad permite asegurar que la información de un objeto esté oculta para el mundo exterior, agrupando en una Clase las características o atributos que cuentan con un acceso privado, y los comportamientos o métodos que presentan un acceso público.

**Abstracción**

La abstracción es cuando el usuario interactúa solo con los atributos y métodos seleccionados de un objeto, utilizando herramientas simplificadas de alto nivel para acceder a un objeto complejo.

En la programación orientada a objetos, los programas suelen ser muy grandes y los objetos se comunican mucho entre sí. El concepto de abstracción facilita el mantenimiento de un código de gran tamaño, donde a lo largo del tiempo pueden surgir diferentes cambios.

**La herencia**

La herencia define relaciones jerárquicas entre clases, de forma que atributos y métodos comunes puedan ser reutilizados. Las clases principales extienden atributos y comportamientos a las clases secundarias. A través de la definición en una clase de los atributos y comportamientos básicos, se pueden crear clases secundarias, ampliando así la funcionalidad de la clase principal y agregando atributos y comportamientos adicionales.

**El polimorfismo**

El polimorfismo consiste en**diseñar objetos para compartir comportamientos**, lo que nos permite procesar objetos de diferentes maneras. Es la capacidad de presentar la misma interfaz para diferentes formas subyacentes o tipos de datos. Al utilizar la herencia, los objetos pueden anular los comportamientos principales compartidos, con comportamientos secundarios específicos. El polimorfismo permite que el mismo método ejecute diferentes comportamientos de dos formas: anulación de método y sobrecarga de método.

**RELACION ENTRE CLASES**

**Agregación / Composición (todo-parte//tiene-un // parte-de).**

Esta relación se presenta entre una clase TODO y una clase PARTE que es componente de TODO. La implementación de este tipo de relación se consigue definiendo como atributo un objeto de la otra clase que es parte-de.

Los objetos de la clase TODO son objetos contenedores. Un objeto contenedor es aquel que contiene otros objetos.

En la agregación, las clases contienen objetos, y no otras clases.

**Asociación**

Especifica una relación semántica entre objetos no relacionados. Este tipo de relaciones permiten crear asociaciones que capturen los participantes en una relación semántica.

Es una relación entre clases. Implica una dependencia semántica. Son relaciones del tipo «pertenece\_a» o «está\_asociado\_con». Se da cuando una clase usa a otra clase para realizar algo.

Multiplicidad de la Relación

Indica el número de instancias de una clase que se asocian con las instancias de la otra clase.

Tipos de multiplicidad:

uno\_a\_uno

uno\_a\_muchos

muchos\_a\_muchos

La implementación en código suele conseguirse mediante un puntero desde la clase a la clase asociada. Un puntero (referencia), por cada instancia de esa relación.

**Generalización / Especialización: HERENCIA.**

ando una clase comparte la estructura y comportamiento de una sola clase.

Se denomina herencia múltiple: Cuando una clase comparte la estructura y comportamiento de varias clases.

Para que un lenguaje de programación pueda ser considerado orientado a objetos, debe implementar el mecanismo de herencia.

La relación de herencia entre dos clases es una relación binaria entre dos clases que nos dice:

**La clase subclase o derivada hereda de la clase base o superclase.**