

## Qué son los paradigmas?

El concepto de paradigma se utiliza en la vida cotidiana como sinónimo de “marco teórico” o para hacer referencia a que algo se toma como “modelo a seguir”. En términos generales, se puede definir al término **paradigma** como la **forma de visualizar e interpretar los múltiples conceptos, esquemas o modelos del comportamiento en diversas disciplinas.**

En las ciencias sociales, **el paradigma se encuentra relacionado al concepto de cosmovisión.** El concepto se emplea para mencionar a todas aquellas experiencias, creencias, vivencias y valores que repercuten y condicionan el modo en que una persona ve la realidad y actúa en función de ellos. Esto quiere decir que **un paradigma es también la forma en que se entiende el mundo.**

## Paradigmas de programación.

**Un paradigma de programación provee y determina la visión y métodos de un programador en la construcción de un programa o subprograma. Diferentes paradigmas resultan en diferentes estilos de programación y en diferentes formas de pensar la solución de problemas (con la solución de múltiples “problemas” se construye una aplicación o producto de software).**

## Clasificación de Paradigmas de Programación

**Un paradigma de programación representa un enfoque particular o filosofía para diseñar soluciones. Los paradigmas difieren unos de otros, en los conceptos y la forma de abstraer los elementos involucrados en un problema, así como en los pasos que integran su solución del problema, en otras palabras, el cómputo. Además, los paradigmas dependen del contexto y el tiempo en el que surgen, ya que nacen en base a una necesidad generalizada de la comunidad de desarrolladores de software, para resolver cierto tipo de problemas de la vida real. Un paradigma de programación está delimitado en el tiempo en cuanto a aceptación y uso, porque**

nuevos paradigmas aportan nuevas o mejores soluciones que la sustituyen parcial o totalmente. De esta forma podemos encontrar los siguientes tipos de paradigmas: imperativo, declarativo, estructurado, funcional, logico.

**Paradigma Orientado a Objetos:** la programación orientada a objetos intenta simular el mundo real a través del significado de objetos que contiene características y funciones. Esta basado en la idea de encapsular estado y operaciones en objetos. En general, la programación se resuelve comunicando dichos objetos a través de mensajes. Su principal ventaja es la reutilización de código y su facilidad para pensar soluciones a determinados problemas.

## Qué es una clase?

Los conceptos de clase y objeto están estrechamente relacionados; sin embargo, existen diferencias entre ambos: mientras que un objeto es una entidad concreta que existe en el espacio y en el tiempo, una clase representa solo una abstracción, la "esencia" de un objeto. Se puede definir una clase como un grupo, conjunto o tipo marcado por atributos comunes, una división, distinción o clasificación de grupos basada en la calidad, grado de competencia o condición. En el contexto del paradigma Orientado a Objetos se define una clase de la siguiente forma: *"Una clase describe un grupo de objetos que comparten una estructura y un comportamiento comunes"*.

Un objeto es una instancia de una clase. Los objetos que comparten estructura y comportamiento similares pueden agruparse en una clase. En el paradigma orientado a objetos, los objetos pertenecen siempre a una clase, de la que toman su estructura y comportamiento.

### Interfaz e implementación

La programación es en gran medida un asunto de contratos: las diversas funciones de un problema mayor se descomponen en problemas más pequeños mediante subcontratos a diferentes elementos del diseño. En ningún sitio es más evidente esta idea que en el diseño de clases.

Una clase sirve como una especie de contrato que vincula a una abstracción y todos sus clientes. Esta visión de la programación como un contrato lleva a distinguir entre la visión

externa y la visión interna de una clase. La interfaz de una clase proporciona su visión externa y enfatiza la abstracción a la vez que oculta su estructura y los secretos de su comportamiento. Esta interfaz se compone principalmente de las declaraciones de todas las operaciones aplicables a instancias de la misma clase.

La implementación de una clase se compone principalmente de la implementación de todas las operaciones definidas en la interfaz de la misma. Lleva a una forma concreta las declaraciones abstractas de como debe ser la implementación.

Se puede dividir la interfaz de una clase en tres partes:

- **Pública:** Una declaración accesible a todos los clientes.
- **Protegida:** Una declaración accesible solo a la propia clase, sus subclases y sus clases amigas.
- **Privada:** Una declaración accesible sólo a la propia clase.

## Qué es un Objeto?

Desde el punto de vista de los humanos un objeto es cualquiera de estas cosas:

- Una cosa tangible y/o visible.
- Algo que puede comprenderse intelectualmente.
- Algo hacia lo que se dirige un pensamiento o una acción.

En software, el término objeto se aplicó formalmente por primera vez en el lenguaje Simula, por primera vez, donde los objetos existían para simular algún aspecto de la realidad. Los objetos del mundo real no son el único tipo de objeto de interés en el desarrollo de software. Otros tipos importantes de objetos son invenciones del proceso de diseño cuyas colaboraciones con objetos semejantes llevan a desempeñar algún comportamiento de nivel superior.

Podemos decir que un **objeto**: "representa un elemento, unidad o entidad individual e identificable, ya sea real o abstracta, con un rol bien definido en el dominio del problema".

De esta forma en el dominio de un problema relacionado con vehículos podríamos identificar como ejemplos de objetos a cada una de las ruedas, las puertas, las luces y demás elementos que lo conforman. Los casos anteriores son representaciones de cosas tangibles y reales, aunque también pueden serlo entidades abstractas como la marca, el modelo, el tipo de

combustible que usa, entre otras. Además, un objeto puede estar formado por otros objetos que ya mencionamos.

*"Un objeto tiene un estado, comportamiento e identidad: la estructura y comportamiento de objetos similares están definidos en su clase común; los términos instancia y objeto son intercambiables".*

**Estado:** el estado de un objeto abarca todas las propiedades (normalmente estáticas) del mismo más los valores actuales (normalmente dinámicos) de cada una de esas propiedades.

**Comportamiento:** el comportamiento es cómo actúa y reacciona un objeto, en términos de sus cambios de estado y paso de mensajes.

**Identidad:** la identidad es aquella propiedad de un objeto que lo distingue de todos los demás objetos.

## Relaciones entre clases:

Las clases, al igual que los objetos, no existen aisladamente. Para un dominio de problema específico, las abstracciones suelen estar relacionadas por vías muy diversas y formando una estructura de clases del diseño.

**Asociación:** una asociación denota una dependencia semántica y puede establecer o no, la dirección de esta dependencia, conocida como navegabilidad. Si se establece la dirección de la dependencia o navegabilidad, debe nombrarse el rol que desempeña la clase en relación con la otra. Esto es suficiente durante el análisis de un problema, donde solo es necesario identificar esas dependencias.

**Multiplicidad:** existen tres tipos habituales de multiplicidad en una asociación:

- Uno a uno.
- Uno a muchos.
- Muchos a muchos.

**Herencia:** la herencia es una herramienta que permite definir nuevas clases en base a otras clases. La herencia es una relación entre clases, en la que una clase comparte la estructura y/o el comportamiento definidos en una (herencia simple) o más clases (herencia múltiple). La clase de la que otras heredan se denomina "superclase", o "clase padre". La clase que hereda de

una o mas clases se denomina “subclase” o “clase hija”. La herencia define una jerarquía de “tipos” entre clases, en la que una subclase hereda de una o mas superclases.

**Polimorfismo:** es un concepto aplicado en el contexto de la teoria de tipos, como en las relaciones de herencia o generalizacion, donde el nombre de un metodo puede implicar comportamientos diferentes, conforme este especificado en clases diferentes, en tanto y en cuanto estas clases esten relacionadas por una superclase común. Un objeto puede responder de diversas formas a un mismo comportamiento, denotado por este nombre, dependiendo de la clase a la que pertenezca.

El polimorfismo es la propiedad que potencia el uso de la herencia y para poder utilizarla, las clases deben especificar los comportamientos con los mismos protocolos. La herencia sin polimorfismo es posible, pero no es muy util.

## El Modelo de Objetos

El modelo de objetos es el marco de referencia conceptual para evaluar si se respeta el paradigma orientado a objetos. Hay cuatro elementos fundamentales en el modelo de objetos:

- Abstraccion
- Encapsulamiento
- Modularidad
- Jerarquía







**Abstracción:** una abstraccion denota las características esenciales de un objeto, que lo distingue de todos los demas y proporciona asi fronteras conceptuales nitidamente definidas respecto a la perspectiva del observador.

**Encapsulamiento:** es el proceso de almacenar en un mismo comportamiento los elementos de una abstracción, que constituyen su estructura y su comportamiento; sirve para separar la interfaz contractual de una abstracción y su implantación.

**Modularidad:** es la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de modulos cohesivos y debilmente acoplados.

**Jerarquía:** la jerarquia es una clasificación u ordenación de abstracciones.

### Material recomendado:

-  Curso C#. POO. ¿Qué es la POO? Vídeo 27
-  Curso C#. POO II. Creación de clases e instancias. Vídeo 28
-  Curso C#. POO III. Encapsulación y convenciones. Vídeo 29
-  Curso C#. POO IV. Constructores. Vídeo 30
-  Curso C#. POO V. Getters y Setters. Vídeo 31
-  Curso C#. Herencia V. Polimorfismo. Vídeo 45






### Windows Form:

Windows Forms is a UI framework for building Windows desktop apps. It provides one of the most productive ways to create desktop apps based on the visual designer provided in Visual Studio. Functionality such as drag-and-drop placement of visual controls makes it easy to build desktop apps.

With Windows Forms, you develop graphically rich apps that are easy to deploy, update, and work while offline or while connected to the internet. Windows Forms apps can access the local hardware and file system of the computer where the app is running.

[Desktop Guide \(Windows Forms .NET\)](#)

### Tutoriales:

-  **Windows Forms C# TUTORIAL**  ( Visual Studio 2019 )
-  **COMO Crear CRUD en Windows Forms DESDE 0 ) Windows Forms C# TUTORIA...**
-   **COMO Llamar Un Procedimiento Almacenado en C# | Windows Forms Entity Fra...**

### Problematica Semanal

1. Agregar validaciones para que no nos deje ingresar una edad que no sea un número.
2. Utilizar la lista que se encuentra en Empresa para llenar dgvEmpleados.
3. Agregar un Label que muestre un mensaje de error cuando queremos agregar un empleado de forma erronea y que se desaparezca cuando ingresamos a algun textBox.

7

Pueden encontrar una resolución en la rama res-clase-4-ejercicio-1.

<https://github.com/JoaquinLeimeter/UAI-POO/tree/res-clase-4-ejercicio-1/4-ejercicio-WF/ejercicio1>