**POLIMORFISMO**

El polimorfismo se refiere a la posibilidad de definir múltiples clases con funcionalidad diferente, pero con métodos o propiedades denominados de forma idéntica.

El polimorfismo es un concepto clave en la programación orientada a objetos que permite a los objetos de diferentes clases ser tratados como objetos de una clase común.

En otras palabras, el polimorfismo permite a los desarrolladores utilizar una única interfaz para representar diferentes tipos de datos y objetos. En C#, el polimorfismo se logra a través de la herencia, las interfaces y la sobrecarga de métodos y operadores.  
  
  
**Tipos de polimorfismo en C#**

En C#, hay dos tipos principales de polimorfismo:

**Polimorfismo estático:** También conocido como sobrecarga, el polimorfismo estático ocurre en tiempo de compilación.

**Polimorfismo dinámico:** También conocido como sobrescritura, el polimorfismo dinámico ocurre en tiempo de ejecución.

**Polimorfismo estático**

**Ejercicio1 - PolimorfismoEstatico\_SobreCargaMetodos**

Ejemplo de sobrecarga de métodos

Aquí hay un ejemplo de sobrecarga de métodos en una clase “Calculadora”:

En este ejemplo, la clase “Calculadora” tiene dos versiones del método “Suma”. Uno acepta dos argumentos enteros y el otro acepta dos argumentos double. El compilador selecciona la versión correcta del método en función de los tipos de argumentos proporcionados.

**Ejercicio2 - PolimorfismoEstatico\_SobreCargaOperadores**

Ejemplo de sobrecarga de operadores

Aquí hay un ejemplo de sobrecarga del operador “+” para una clase “Vector”:

En este ejemplo, la clase **“Vector”** sobrecarga el operador **“+”** para permitir la suma de dos objetos **“Vector”**. El operador **“+”** se define como un método estático con la palabra clave **“operator”** seguida del símbolo del operador que se sobrecarga.

* **operator**: Palabra clave que indica que se está sobrecargando un operador.
* +: El operador específico que se está sobrecargando.
* (Vector a, Vector b): Los parámetros que el operador toma. En este caso, dos objetos de tipo Vector.

La sobrecarga de operadores permite definir cómo los operadores estándar (como +, -, \*, etc.) se comportan para tipos definidos por el usuario, proporcionando una sintaxis más intuitiva y natural para realizar operaciones con esos tipos.

**Ejercicio3 - PolimorfismoEstatico\_SobreCargaConstructores**

Algunos también consideran polimorfismo a la sobrecarga de métodos o constructores, que básicamente es que un método o constructor se puede declarar varias veces con el mismo nombre, pero con diferentes parámetros, ya sea por cantidad de parámetros o tipo de dato.

*Crear instancias de la clase Casa utilizando diferentes constructores*

**Polimorfismo dinámico (sobrescritura) en C#**

La sobrescritura es una forma de polimorfismo dinámico en la que una clase derivada redefine un método o propiedad de su clase base.

La sobrescritura permite que la clase derivada proporcione una implementación específica de un método o propiedad mientras mantiene la misma firma que la versión de la clase base.

En tiempo de ejecución, el tipo de objeto determina qué versión del método o propiedad se debe llamar.

Uso de las palabras clave virtual, override y ‘base’

En C#, las palabras clave virtual, override y ‘base’ se utilizan para controlar y gestionar la sobrescritura de métodos y propiedades:

* virtual: Se utiliza en la clase base para indicar que un método o propiedad puede ser sobrescrito en una clase derivada.
* override: Se utiliza en la clase derivada para indicar que un método o propiedad está sobrescribiendo un método o propiedad virtual de la clase base.
* base: Se utiliza en la clase derivada para llamar a la versión del método o propiedad de la clase base que está siendo sobrescrita.

**CON CLASES**

**Ejercicio4 - PolimorfismoDinamico\_ClaseAbstract**

* Polimorfismo usando clases
* Se tiene una clase Empleado y otra clase Alumno, las cuales heredan de una clase abstracta Persona que tiene el metodo Saludar. En las clases hijas sobreescribiremos el metodo Saludar devolviendo el nombre de la persona y si es un empleado o un alumno.
* En vez de tener un arreglo de empleados y otro de alumnos por separado, usaremos el polimorfismo con un solo arreglo de personas, al cual agregaremos un alumno y un empleado para luego iterar y llamar al metodo saludar.

**Ejercicio5 - PolimorfismoDinamico\_Clase**

* Tenemos la superclase o clase padre Inmueble, con sus respectivos atributos y el método VENDER. La clase casa, departamento y terreno derivan (Heredan) de la clase Inmueble, por lo tanto, estas clases pueden usar los atributos y métodos protegidos o públicos de la clase Inmueble.

**Ejercicio6 - PolimorfismoDinamico\_ ClaseAbstract**

* Un ejemplo simple de polimorfismo sería tener una clase base llamada “Forma” y varias clases derivadas como “Círculo”, “Rectángulo” y “Triángulo”.
* Aunque cada una de estas clases derivadas tiene propiedades y métodos específicos, todas ellas pueden ser tratadas como objetos de la clase “Forma”. Esto permite a los desarrolladores escribir código más genérico y reutilizable.

**CON INTERFACES**

**Ejercicio7 - PolimorfismoDinamico\_ Interfaces**

* Polimorfismo usando interfaces  
    
  De la misma manera que en el ejemplo anterior, en vez de usar una clase padre Persona de la cual heredan las clases hijas Empleado y Alumno podemos definir una interfaz IPersona que defina el metodo Despedirse, y que las clases Empleado y Alumno implementen la interfaz IPersona.
* De la misma manera que en el ejemplo anterior, podemos instanciar un arreglo de tipo IPersona, agregar objetos que implementen la interfaz IPersona: un objeto alumno o un objeto empleado, para luego iterar y llamar al metodo Despedirse.

**Ejercicio8 - PolimorfismoDinamico\_ Interfaces**

Desarrolla un sistema de procesamiento de pagos para una tienda en línea que admite diferentes métodos de pago: tarjetas de crédito, PayPal y transferencias bancarias. Cada método de pago debe implementar una interfaz IPago que tenga un método ProcesarPago para procesar el pago y un método GenerarRecibo para generar un recibo de pago.

La tienda debe poder procesar una lista de pagos utilizando diferentes métodos de pago, mostrando los detalles de cada pago procesado y generando los recibos correspondientes. El sistema debe ser flexible para permitir la adición de nuevos métodos de pago en el futuro sin modificar el código existente.

**Ejercicio Final 1**

Desarrolla un sistema de gestión de vehículos para una empresa de alquiler de vehículos. La empresa tiene diferentes tipos de vehículos: coches, motos y camiones. Cada vehículo tiene una matrícula, un modelo y una tarifa diaria de alquiler. Los coches tienen un número de puertas, las motos tienen un tipo de motor (eléctrico o gasolina), y los camiones tienen una capacidad de carga en toneladas.

Crea una clase base Vehiculo y las clases derivadas Coche, Moto y Camion. Implementa un método polimórfico CalcularTarifa que, además de la tarifa diaria base, aplique diferentes recargos según el tipo de vehículo:

* Los coches tienen un recargo del 10% si tienen más de 4 puertas.
* Las motos tienen un recargo del 5% si son de gasolina.
* Los camiones tienen un recargo del 20% si la capacidad de carga es superior a 10 toneladas.

**Ejercicio Final 2: Sistema de Facturación Electrónica**

Implementa un sistema de facturación electrónica que gestione diferentes tipos de documentos fiscales: facturas, notas de crédito y notas de débito. Cada documento tiene un número, una fecha y un monto. Además, cada tipo de documento tiene características adicionales:

* Las facturas tienen un campo de Detalle.
* Las notas de crédito tienen un campo de MontoDescuento.
* Las notas de débito tienen un campo de MontoRecargo.

Crea una clase base DocumentoFiscal y las clases derivadas Factura, NotaCredito y NotaDebito. Implementa un método polimórfico CalcularTotal que calcule el monto total aplicando los descuentos o recargos correspondientes según el tipo de documento.