**PRACTICA 2: 21 DE AGOSTO DEL 2024**

**UNIVERSIDAD DE ORIENTE**

**Catedra:** Programación Orientada a Eventos. Sección A

**Instructor:** Williams Ernesto Rodríguez Ortiz

**Tema:** Programación Orientada a Objetos en C#

**REQUISITOS.**

* Visual Studio

**OBJETIVO.**

Aprender sobre la Programación Orientada a Objetos para la resolución de problemas.

**1. ¿Qué es la Programación Orientada a Objetos (POO)?**

La POO es un paradigma de programación que organiza el software en objetos, que son instancias de clases. Una clase es una plantilla que define las características y comportamientos (propiedades y métodos) de los objetos. Este enfoque permite estructurar el código de manera más natural y modular, lo que facilita el desarrollo, mantenimiento y escalabilidad del software.

**2. ¿En qué ayuda la POO?**

* **Modularidad**: El código se organiza en objetos, lo que facilita la reutilización y el mantenimiento.
* **Reutilización**: Las clases pueden reutilizarse en diferentes programas o proyectos, lo que ahorra tiempo y esfuerzo.
* **Facilidad de Mantenimiento**: El código es más fácil de mantener y actualizar debido a su estructura modular.
* **Escalabilidad**: Es más fácil añadir nuevas funcionalidades sin afectar a las ya existentes.
* **Claridad y Naturalidad**: La POO se asemeja a cómo percibimos el mundo real, lo que facilita la comprensión y la comunicación entre desarrolladores.

**Herencia**:

La herencia es uno de los pilares clave de la POO.

* Permite crear nuevas clases (clases derivadas) a partir de clases existentes (clases base).
* Las clases derivadas heredan los atributos y métodos de las clases base.
* La herencia facilita la reutilización de código y la organización lógica de las clases.

**Encapsulamiento**:

Consiste en ocultar los detalles internos de un objeto y exponer solo la interfaz necesaria.

* Los atributos de un objeto se declaran como privados o protegidos, y se accede a ellos a través de métodos públicos (getters y setters).
* El encapsulamiento mejora la modularidad, la seguridad y el mantenimiento del código.
* Ejemplo: la clase Cuenta tiene un atributo saldo privado, y se accede a él mediante los métodos públicos Depositar() y Retirar().

**Abstracción**:

La abstracción es el proceso de identificar las características y comportamientos esenciales de un objeto, ocultando los detalles irrelevantes.

* La abstracción permite definir conceptos generales que luego pueden ser especializados.

**Polimorfismo**:

El polimorfismo es la capacidad de un objeto de tomar múltiples formas.

* El polimorfismo se logra a través de la sobrecarga de métodos y la redefinición (overriding) de métodos.
* La sobrecarga de métodos permite que un método tenga múltiples implementaciones, diferenciadas por sus parámetros.
* La redefinición de métodos permite que las clases derivadas proporcionen su propia implementación de un método definido en la clase base.

**Ejercicios Prácticos:**

* **Ejercicio de Encapsulamiento:**
  + Crear una clase Cuenta que permita realizar operaciones de depósito y retiro de dinero. La clase Cuenta debe tener un atributo saldo privado, y métodos públicos RealizarDepositar() y RealizarRetiro() para manipular el saldo.
* **Ejercicio de Herencia:** 
  + Crear una jerarquía de clases que represente diferentes tipos de animales. La clase base Animal tendrá atributos como Nombre y Edad, y un método Comer(). La clase derivada Perro heredará de Animal y tendrá un método adicional Ladrar().
* **Ejercicio de Polimorfismo:**
  + Crear una jerarquía de clases que represente diferentes tipos de computadoras: Computadora, ComputadoraPortatil y ComputadoraEscritorio. Cada clase debe tener un método Encender() que muestre un mensaje diferente. En un formulario, crear una lista de objetos Computadora y llamar al método Encender() de cada uno, demostrando el polimorfismo.

**NOTA**: Deberá presentar la solución de estos ejercicios a más tardar el día 27 de agosto a las 11:59 pm