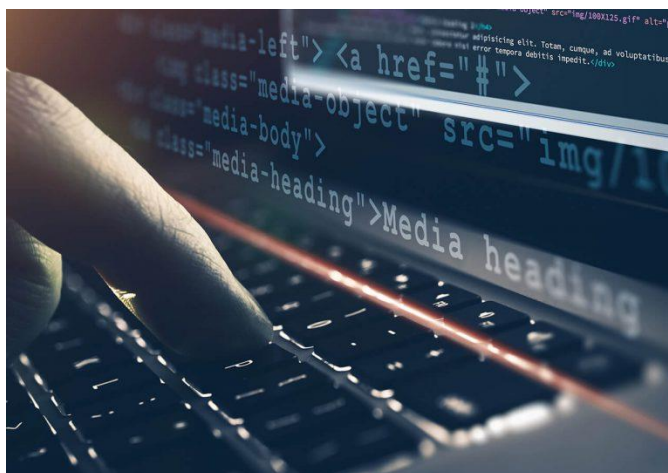




ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA



PROGRAMACIÓN WEB **AVANZADA**



Prueba 1er parcial

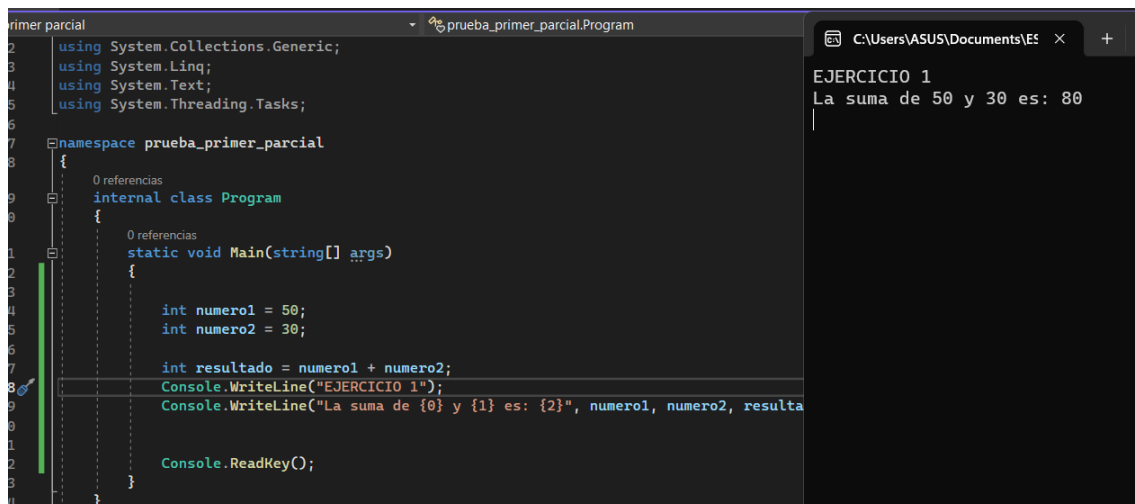
NOMBRE: MASAPANTA JEFFERSON

NRC: 14956

FECHA: 13 – 12 - 2023

Ejercicio 1: Variables y Operadores en C#

1. Declarar dos variables, numero1 y numero2, e inicialízalas con valores numéricos.
2. Calcula la suma de estas dos variables y almacena el resultado en una tercera variable llamada resultado.
3. Imprime en la consola el valor de resultado.



```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace prueba_primer_parcial
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int numero1 = 50;
            int numero2 = 30;

            int resultado = numero1 + numero2;
            Console.WriteLine("EJERCICIO 1");
            Console.WriteLine("La suma de {0} y {1} es: {2}", numero1, numero2, resultado);

            Console.ReadKey();
        }
    }
}
```

EJERCICIO 1
La suma de 50 y 30 es: 80

Ejercicio 2: Estructuras de Control en C#

1. Declara una variable edad e inicialízala con un valor numérico.
2. Utiliza una estructura if para determinar si la persona es mayor de edad (mayor o igual a 18).
3. Imprime en la consola un mensaje indicando si la persona es mayor de edad o no.



```
int edad = 30;
Console.WriteLine("EJERCICIO 2");
if (edad >= 18)
{
    Console.WriteLine("La persona es mayor de edad.");
}
else
{
    Console.WriteLine("La persona no es mayor de edad.");
}
Console.ReadKey();
```

EJERCICIO 1
La suma de 50 y 30 es: 80
EJERCICIO 2
La persona es mayor de edad.

Ejercicio 3: Programación Orientada a Objetos - Clases y Objetos

1. Crea una clase llamada Estudiante con propiedades como Nombre, Edad y Calificación.
2. Crea un objeto de tipo Estudiante llamado estudiante1 e inicializa sus propiedades con valores ficticios.
3. Imprime en la consola la información del estudiante.

<pre>Estudiante estudianteespe = new Estudiante { Nombre = "Jefferson Masapanta", Edad = 30, Calificación = 19.25 }; Console.WriteLine("EJERCICIO 3"); Console.WriteLine("Información del Estudiante:"); Console.WriteLine(\$"Nombre: {estudianteespe.Nombre}"); Console.WriteLine(\$"Edad: {estudianteespe.Edad} años"); Console.WriteLine(\$"Calificación: {estudianteespe.Calificación}");</pre>	<pre>EJERCICIO 1 La suma de 50 y 30 es: 80 EJERCICIO 2 La persona es mayor de edad. EJERCICIO 3 Información del Estudiante: Nombre: Jefferson Masapanta Edad: 30 años Calificación: 19,25</pre>
--	---

Ejercicio 4: Programación Orientada a Objetos - Métodos

1. Amplía la clase Estudiante con un método llamado MostrarInformacion que imprima en la consola los detalles del estudiante.
2. Llama al método MostrarInformacion para el objeto estudiante1 y observa la salida.

<pre>Console.WriteLine("EJERCICIO 4"); estudianteespe.MostrarInformacion(); EstudianteGraduado graduado1 = new EstudianteGraduado { Nombre = "Estefania Vargas", Edad = 22,</pre>	<pre>Nombre: Jefferson Masapanta Edad: 30 años Calificación: 19,25 EJERCICIO 4 Nombre: Jefferson Masapanta Edad: 30 años Calificación: 19,25 EJERCICIO 5</pre>
--	--

Ejercicio 5: Programación Orientada a Objetos - Herencia

1. Crea una nueva clase llamada EstudianteGraduado que herede de la clase Estudiante.
2. Añade una nueva propiedad a EstudianteGraduado llamada Titulo que almacene el título obtenido.
3. Crea un objeto de tipo EstudianteGraduado llamado graduado1 e inicializa sus

propiedades.

4. Utiliza el método `MostrarInformacion` de la clase base para mostrar la información del estudiante graduado.

<pre>EstudianteGraduado graduadoespe = new EstudianteGraduado { Nombre = "Estefania Vargas", Edad = 22, Calificacion = 19.35, Titulo = "Ingeniera en Sistemas" }; Console.WriteLine("EJERCICIO 5"); Console.WriteLine("Información del Estudiante Graduado:"); graduadoespe.MostrarInformacion();</pre>	<pre>EJERCICIO 4 Nombre: Jefferson Masapanta Edad: 30 años Calificación: 19,25 EJERCICIO 5 Información del Estudiante Graduado: Nombre: Estefania Vargas Edad: 22 años Calificación: 19,35 Título obtenido: Ingeniera en Sistemas</pre>
--	---

Preguntas de Reflexión:

1. ¿Cuál es la diferencia entre una variable y una propiedad en C#?

Un variable es un espacio identificado por un nombre en el que se puede almacenar un valor o una referencia a un objeto. Su declaración implica la especificación de un tipo de datos y tiene la capacidad de cambiar su contenido durante la ejecución del programa.

En contraste, una propiedad representa una forma de acceder y manipular la lectura y escritura de un campo en una clase. Las propiedades suelen incluir métodos "get" para obtener el valor y "set" para asignar un nuevo valor. Estas ofrecen un nivel de encapsulación y control sobre el acceso a los datos.

2. Explica cómo funciona la estructura if y por qué es útil en programación.

La estructura "if" es una herramienta de control de flujo en programación que posibilita la ejecución de un conjunto de instrucciones solo si se cumple una condición específica. Su funcionamiento implica la evaluación de una expresión booleana. Cuando la condición es verdadera, se ejecuta el bloque de código asociado al "if"; en caso contrario, se omite esa porción de código. Su utilidad radica en la capacidad de permitir que un programa tome decisiones con base en condiciones. Esto posibilita la ejecución de acciones distintas según si una condición dada es verdadera o falsa, otorgando flexibilidad y control al flujo del programa

3. ¿Qué ventajas ofrece la programación orientada a objetos en comparación con otros paradigmas de programación?

La Programación Orientada a Objetos (POO) posibilita la reutilización eficiente de código al permitir la creación de clases y objetos que pueden ser empleados en diversas partes del programa, resultando en un ahorro considerable de tiempo y esfuerzo. La modularidad se alcanza mediante la encapsulación y la abstracción, facilitando la gestión y mantenimiento del código al dividirlo en unidades más manejables. La POO simplifica el modelado del mundo real, ya que los objetos en este paradigma representan entidades y sus interacciones, lo que facilita tanto la comprensión como el diseño de sistemas. La herencia y el polimorfismo son conceptos fundamentales que permiten la creación de jerarquías de clases y la implementación de interfaces, mejorando la estructura y la extensibilidad del código de manera significativa.

4. ¿Cuándo usarías la herencia en un diseño de clases?

La herencia se emplea al desear crear una nueva clase que comparta atributos y comportamientos de una clase ya existente, conocida como clase base o padre. Facilita la reutilización de código y la expansión de capacidades, ya que la clase derivada (hija) hereda las características de la clase base.

Se aplica cuando existe una relación de "es un" entre la clase base y la derivada. Por ejemplo, si se tiene una clase "Vehículo", podría haber clases derivadas como "Automóvil" y "Bicicleta".

5. ¿Por qué es importante la encapsulación en programación orientada a objetos?

La encapsulación representa la idea de ocultar los pormenores internos de una clase y ofrecer una interfaz pública para su interacción. Sirve para salvaguardar los datos de la clase al restringir el acceso y la modificación directa desde fuera de dicha clase. Facilita la modificación interna de la implementación sin impactar a otras partes del programa que hacen uso de la clase.

Contribuye a la modularidad y mantenimiento del código al garantizar que las modificaciones internas no afecten a otros componentes, siempre y cuando la interfaz pública permanezca inalterada.

Entregable:

Todo el trabajo se ha subido a un repositorio en GitHub. La URL del repositorio es

https://github.com/jsmasapanta/Masapanta_Jefferson_evaluacion_1_parcial.git