CODES
Introducción a objetos - Práctica

codes
CONSULTORIA
Y DESARROLLO DE SISTEMAS

CMMI

ÍNDICE

I. EJERCICIOS

Generalidades:

- Probar todo el código que uno escribe

- Utilizar la consola de C# para visualizar la ejecucion del programa

- Cuando no se especifíca que debe hacer un método, mostrar un mensaje en consola indicando que el método se ejecutó. Ej en el ejercicio 6 en el método cargar() se puede mostrar un mensaje por consola que diga "Cargando..."

1. EJERCICIO 1

Crear una clase Complejo que permita trabajar con números complejos *a*+*b*i donde *a* es la parte real y *b* la parte imaginaria. Representadas con números de tipo *float*

Crear constructores con y sin parámetros. ¿Que parámetros se usarian?

Incluir los siguientes métodos:

- Complejo *sumar(Comblejo z):* Suma otro complejo y retorna dicha suma
- Complejo *multiplicar*: multiplica la parte real e imaginaria por una constante real.
- bool esIgualA(Complejo z): Retorna true si y solo si el complejo es igual al complejo z
-*void print()*: Muestra en la consola un complejo con el formato "a+bi".

Que ocurre al enviar el mensaje ToString a un complejo?
Implementar el metodo ToString

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/how-to-override-the-tostring-method

2. EJERCICIO 1 – SOLUCIÓN PROPUESTA

```
public class Complejo
    {
        private int _parteReal;
        private int _parteImaginaria;

        public Complejo()
        {
            this._parteImaginaria = 0;
            this._parteReal = 0;
        }

        public Complejo(int parteReal, int parteImaginaria)
        {
            this._parteReal = parteReal;
            this._parteImaginaria = parteImaginaria;
        }

        public int ParteReal
        {
            get { return this._parteReal; }
            set { this._parteImaginaria = value; }
        }

        public int ParteImaginaria
        {
            get { return this._parteImaginaria; }
            set { this._parteImaginaria = value; }
        }
```

Continúa en próxima diapositiva…

3. EJERCICIO 1 – SOLUCIÓN PROPUESTA

```
    public Complejo Sumar(Complejo sumando)
    {
        Complejo resultado = new Complejo();
        resultado.ParteReal = this._parteReal + sumando.ParteReal;
        resultado.ParteImaginaria = this._parteImaginaria + sumando.ParteImaginaria;

        return resultado;
    }

    public Complejo Restar(Complejo sustraendo)
    {
        Complejo resultado = new Complejo();
        resultado.ParteReal = this._parteReal - sustraendo.ParteReal;
        resultado.ParteImaginaria = this._parteImaginaria - sustraendo.ParteImaginaria;

        return resultado;
    }

    //(a + bi ) (c + di ) = (ac - bd) + (ad + bc)i
    public Complejo Multiplicar(Complejo multiplicador)
    {
        Complejo resultado = new Complejo();

        resultado.ParteReal = this.ParteReal * multiplicador.ParteReal
                    - this.ParteImaginaria * multiplicador.ParteImaginaria;
        resultado.ParteImaginaria = this.ParteReal * multiplicador.ParteImaginaria
                    + this.ParteImaginaria * multiplicador.ParteReal;

        return resultado;
    }

}
```

NUMEROS RACIONALES

Crear una clase Racional que permita trabajar con números racionales (fracciones).

Recordar que un número racional se puede escribir como a/b con a y b reales y b distinto de 0.

Crear un constructor parametrizado. ¿Es válido el denominador 0? Como se puede evitar construir instancias con denomidador 0?

(INVESTIGAR EXCEPTIONS https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/exceptions/)

Incluir los siguientes métodos:
Sumar: recibe otro racional y retorna un racional que representa la suma.
Multiplicación: recibe otro racional y representa la multiplicación de ambos.

COCHE

Crear las siguientes clases (cada una en su archivo):

- Motor: con métodos para arrancar el motor y apagarlo. Se desea saber también cuando el motor esta prendido o apagado.

- Rueda: con métodos para inflar la rueda y desinflarla. Se desea también saber si la rueda esta inflada o desinflada.

- Ventana: con métodos para abrirla y cerrarla. Se desea saber si la ventana está abierta o cerrada

- Puerta: con una ventana y métodos para abrir la puerta y cerrar la puerta.

- Coche: con un motor, cuatro ruedas y dos puertas. Implementar un método print que muestre en consola el estado de todos sus componentes. (Motor: Encendido, Rueda1: Inflada, Rueda2: Desinflada, etc)

LISTA AL REVES

Escribir un programa que pida diez palabras y las muestre en orden inverso (del último leído hasta el primero)
Utilizar el metodo ReadLine y el objeto List<string>

https://docs.microsoft.com/en-us/dotnet/api/system.console.readline?view=net-5.0

https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=net-5.0

- Puede utilizarse el objeto Stack?

Stack:
https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.stack-1?view=net-5.0

CATALOGO DE LIBROS

Desarrollar un catálogo de Libros. De los libros nos interesa su nombre y año de publicación. Las funcionalidades serán:

- Conocer el número de libros que hay en el catálogo,
- Agregar un nuevo libro
- Eliminar un libro a partir de su nombre o parte de él
- Obtener un libro por su nombre o parte de él.

También incluirá los siguientes métodos:

- Buscar un libro a partir de una parte de su título (sin distinguir entre mayúsculas y minúsculas);
- Buscar un libro según la cantidad de caracteres en su título
- Buscar un libro según su año de publicación
- Si quisiéramos agregar un método para buscar según algún otro criterio, en que cambiaría el mismo con respecto a los 3 métodos anteriores? Se puede implementar de otra forma?

Ver método Where:
https://docs.microsoft.com/en-us/dotnet/api/system.linq.enumerable.where?view=net-5.0

¿MULTIHERENCIA?

Se desea modelar un sistema para algunos artefactos electricos y recargables. Implementar el punto 1 y luego hacer las modificaciones para el punto 2.
Detallar comentando el código cuales fueron los cambios realizados.

1) Se dispone de **Lamparas** y **Linternas.** Como es de esperarse, un **Enchufe** permite conectar una lampara. Por otro lado las linternas se pueden **cargar** mediante un **Cargador Portatil**. Se sabe que luego ingresarán mas artefactos eléctricos y artefactos recargables que se podrán conectar a enchufes y otros que se podrán cargar con cargadores portatiles pero aún no se sabe cuales serán.

2) Llega sin previo aviso un artefacto llamado **Farol Led** que se puede conectar a un enchufe para usarlo como si fuera un velador de mesa de luz como también se puede cargar con un cargador para encenderlo sin cable gracias a su bateria interna.

FIN

Gracias

codes
CONSULTORIA
Y DESARROLLO DE SISTEMAS

CMMI