EJERCICIOS PRÁCTICOS CURSO .NET CON C#

Conceptos Básicos

1) Ingresar 5 números por consola, guardándolos en una variable escalar. Luego calcular y mostrar: el valor máximo, el valor mínimo y el promedio.

2) Ingresar un número y mostrar: el cuadrado y el cubo del mismo. Se debe validar que el número sea mayor que cero, caso contrario, mostrar el mensaje: ***"ERROR. ¡Reingresar número!"***.
Nota: *Utilizar el método '**Pow**' de la clase **Math*** para realizar la operación.

3) Mostrar por pantalla todos los **números primos** que haya hasta el número que ingrese el usuario por consola.

4) Escribir un programa que determine si un año es bisiesto.
Un año es bisiesto si es múltiplo de 4. Los años múltiplos de 100 no son bisiestos, salvo si ellos también son múltiplos de 400. Por ejemplo: el año 2000 es bisiesto pero 1900 no.
Nota: *Utilizar estructuras repetitivas, selectivas y la función módulo (%).*

5) Hacer un programa que pida por pantalla la fecha de nacimiento de una persona (día, mes y año) y calcule el número de días vividos por esa persona hasta la fecha actual (tomar la fecha del sistema con ***DateTime.Now***).
Nota*: Utilizar estructuras selectivas. Tener en cuenta los años bisiestos.*

6) Por teclado se ingresa el valor hora, el nombre, la antigüedad (en años) y la cantidad de horas trabajadas en el mes de ***N*** empleados de una fábrica.
Se pide calcular el importe a cobrar teniendo en cuenta que el total (que resulta de multiplicar el valor hora por la cantidad de horas trabajadas), hay que sumarle la cantidad de años trabajados multiplicados por $ 150, y al total de todas esas operaciones restarle el 13% en concepto de descuentos.
Mostrar el recibo correspondiente con el nombre, la antigüedad, el valor hora, el total a cobrar en bruto, el total de descuentos y el valor neto a cobrar de todos los empleados ingresados.
Nota: *Utilizar estructuras repetitivas*

7) Escribir un programa que imprima por pantalla una pirámide como la siguiente:

```
*
***
*****
*******
*********
```

El usuario indicará cuál será la altura de la pirámide ingresando un número entero positivo. Para el ejemplo anterior la altura ingresada fue de 5.
Nota: *Utilizar estructuras repetitivas y selectivas.*

Métodos Estáticos

8) Ingresar 10 números enteros que pueden estar dentro de un rango de entre -100 y 100.

Para ello realizar una clase llamada **Validacion** que posea un método estático llamado **Validar**, que posea la siguiente firma: ***bool Validar(int valor, int min, int max)***:
a. valor: dato a validar
b. min y max: rango en el cual deberá estar la variable valor.
Terminado el ingreso mostrar el valor mínimo, el valor máximo y el promedio.
Nota: *Utilizar variables escalares, NO utilizar vectores*

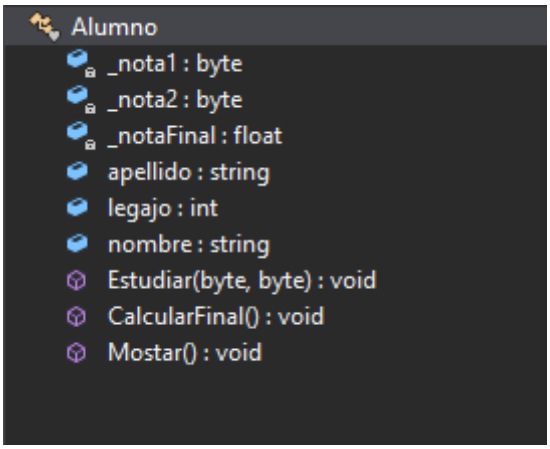
9) Realizar un programa que sume números enteros hasta que el usuario lo determine, por medio de un mensaje "**¿Continuar? (S/N)**".
En el método estático ***ValidaS_N(char c)*** de la clase **ValidarRespuesta**, se validará el ingreso de opciones. El método devolverá un valor de tipo booleano, *TRUE* si se ingresó una 'S' y *FALSE* si se ingresó cualquier otro valor.

10) Desarrollar una clase llamada **Conversor**, que posea dos métodos de clase (**estáticos**):
string DecimalBinario(double). Convierte un número de decimal a binario.
double BinarioDecimal(string). Convierte un número binario a decimal.

11) Realizar un programa que permita realizar operaciones matemáticas simples (suma, resta, multiplicación y división). Para ello se le debe pedir al usuario que ingrese dos números y la operación que desea realizar (pulsando el caracter +, -, * ó /).
El usuario decidirá cuándo finalizar el programa.
Crear una clase llamada **Calculadora** que posea tres métodos estáticos (de clase):
a. **Calcular** (público): Recibirá tres parámetros, el primer número, el segundo número y la operación matemática. El método devolverá el resultado de la operación.
b. **Validar** (privado): Recibirá como parámetro el segundo número. Este método se debe utilizar sólo cuando la operación elegida sea la DIVISIÓN. Este método devolverá TRUE si el número es distinto de CERO.
c. **Mostrar** (público): Este método recibe como parámetro el resultado de la operación y lo muestra por pantalla. No posee valor de retorno.

<u>Objetos</u>

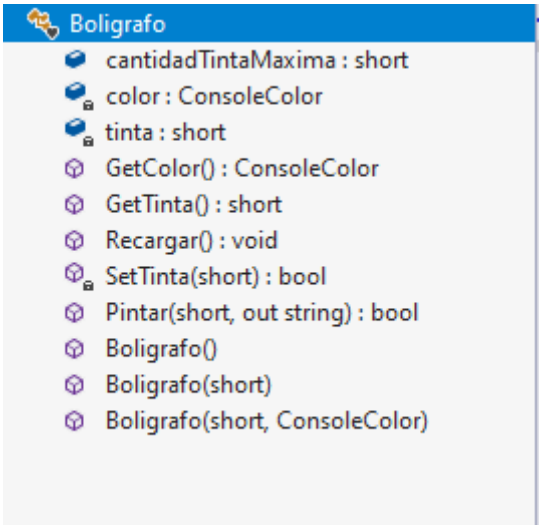
12) Crear la clase Alumno de acuerdo al siguiente diagrama:



a. Se pide crear 3 instancias de la clase Alumno (3 objetos) en la función Main. Colocarle nombre, apellido y legajo a cada uno de ellos.

b. Sólo se podrá ingresar las notas (nota1 y nota2) a través del método **Estudiar**.
c. El método **CalcularFinal** deberá colocar la nota del final sólo si las notas 1 y 2 son mayores o iguales a 4, caso contrario la inicializará con -1. Para darle un valor a la nota final utilice el método de instancia **Next** de la clase **Random**.
d. El método Mostrar, expondrá en la consola todos los datos de los alumnos. La nota final se mostrará sólo si es distinto de -1, caso contrario se mostrará la leyenda "Alumno desaprobado".

13) Crear la clase **Bolígrafo** a partir del siguiente diagrama:



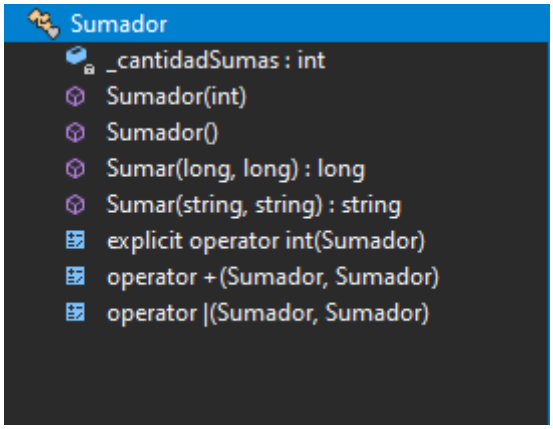
a. La cantidad máxima de tinta para todos los bolígrafos será de 100. Generar una constante en el Boligrafo llamada cantidadTintaMaxima donde se guardará dicho valor.
b. Generar los métodos ***GetColor y GetTinta*** para los correspondientes atributos (sólo retornarán el valor del mismo).
c. Generar el método privado SetTinta que valide el nivel de tinta y asigne en el atributo.

1. **tinta** será el valor a agregar de tinta, pudiendo ser positivo (cargar tinta) o negativo (gastar tinta)
2. Se deberá validar que el nivel de tinta quede entre los valores válidos mayor o igual a 0 y menor o igual a cantidadTintaMaxima.

d. Recargar() colocará a tinta en su nivel máximo de tinta. **Reutilizar código**.
e. En el Main, crear un bolígrafo de tinta azul (***ConsoleColor.Blue***) y una cantidad inicial de tinta de 100 y otro de tinta roja (***ConsoleColor.Red***) y 50 de tinta.
f. El método Pintar(int, out string) restará la tinta gastada (**reutilizar código** SetTinta), sin poder quedar el nivel en negativo, avisando si pudo pintar (nivel de tinta mayor a 0).
También informará mediante el out string tantos "*" como haya podido "gastar" del nivel de tinta. O sea, si nivel de tinta es 10 y gasto es 2 valdrá "**" y si nivel de tinta es 3 y gasto es 10 "***".
g. Utilizar todos los métodos en el Main.
h. Al utilizar Pintar, si corresponde, se deberá dibujar por pantalla con el color de dicho bolígrafo.
Nota: *Crear el constructor que crea conveniente. La clase Boligrafo y el Program deben estar en namespaces distintos.*

Sobrecarga de Operadores

14) Realizar una aplicación de consola. Agregar la clase Sumador.



a. Crear dos constructores:

- Sumador(int) inicializa cantidadSumas en el valor recibido por parámetros.
- Sumador() inicializa cantidadSumas en 0. Reutilizará al primer constructor.

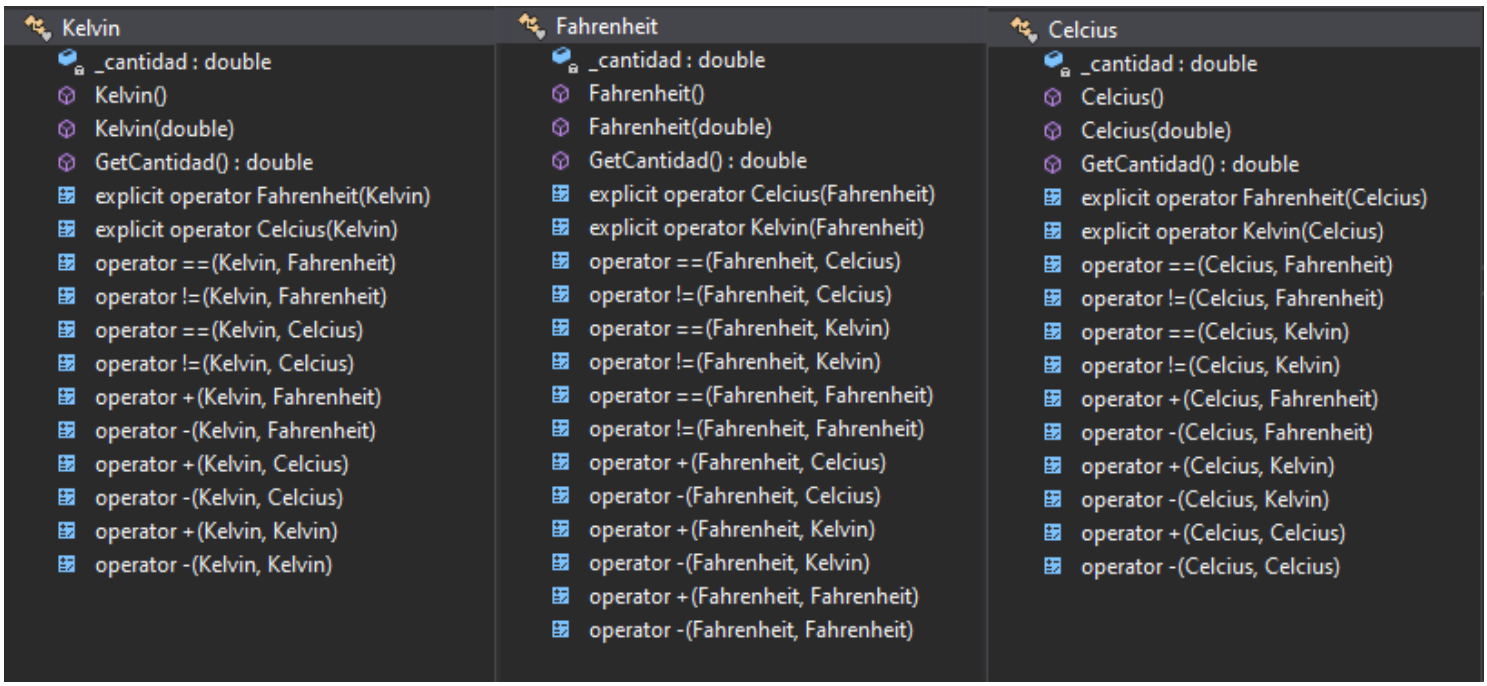
b. El método Sumar incrementará cantidadSumas en 1 y adicionará sus parámetros con la siguiente lógica:

- En el caso de Sumar(long, long) sumará los valores numéricos
- En el de Sumar(string, string) concatenará las cadenas de texto.

Antes de continuar, agregar un objeto del tipo Sumador en el Main y probar el código.
c. Generar una conversión explícita que retorne cantidadSumas.
d. Sobrecargar el operador + (suma) para que puedan sumar cantidadSumas y retornen un long con dicho valor.
e. Sobrecargar el operador | (pipe) para que retorne True si ambos sumadores tienen igual cantidadSumas.
Agregar un segundo objeto del tipo Sumador en el Main y probar el código.

15) Crear tres clases: **Fahrenheit**, **Celsius** y **Kelvin**. Teniendo en cuenta que:
F = C * (9/5) + 32
C = (F-32) * 5/9
F = K * 9/5 – 459.67
K = (F + 459.67) * 5/9

a. Se debe lograr que los objetos de estas clases se puedan sumar, restar y comparar entre sí.
b. Sobrecargar los operadores **explicit** y/o **implicit** para lograr compatibilidad entre los distintos tipos de datos.
c. Colocar dentro del Main el código necesario para probar todos los métodos.
d. Los comparadores == compararan cantidades.
e. Reutilizar el código. Sólo realizar las conversiones dentro de los operadores para dicho uso.
f. El método `GetCantidad` es estático.



16) Generar un nuevo proyecto del tipo Console Application. Construir tres clases dentro de un namespace llamado Billetes: **Pesos**, **Euro** y **Dolar**.

a. Se debe lograr que los objetos de estas clases se puedan sumar, restar y comparar entre sí con total normalidad como si fueran tipos numéricos, teniendo presente que 1 Euro equivale a 1,10 dólares y 1 dólar equivale a 55,93 pesos.

b. El atributo cotizRespectoDolar y el método GetCotizacion son estáticos.

c. Sobrecargar los operadores **explicit** y/o **implicit** para lograr compatibilidad entre los distintos tipos de datos.

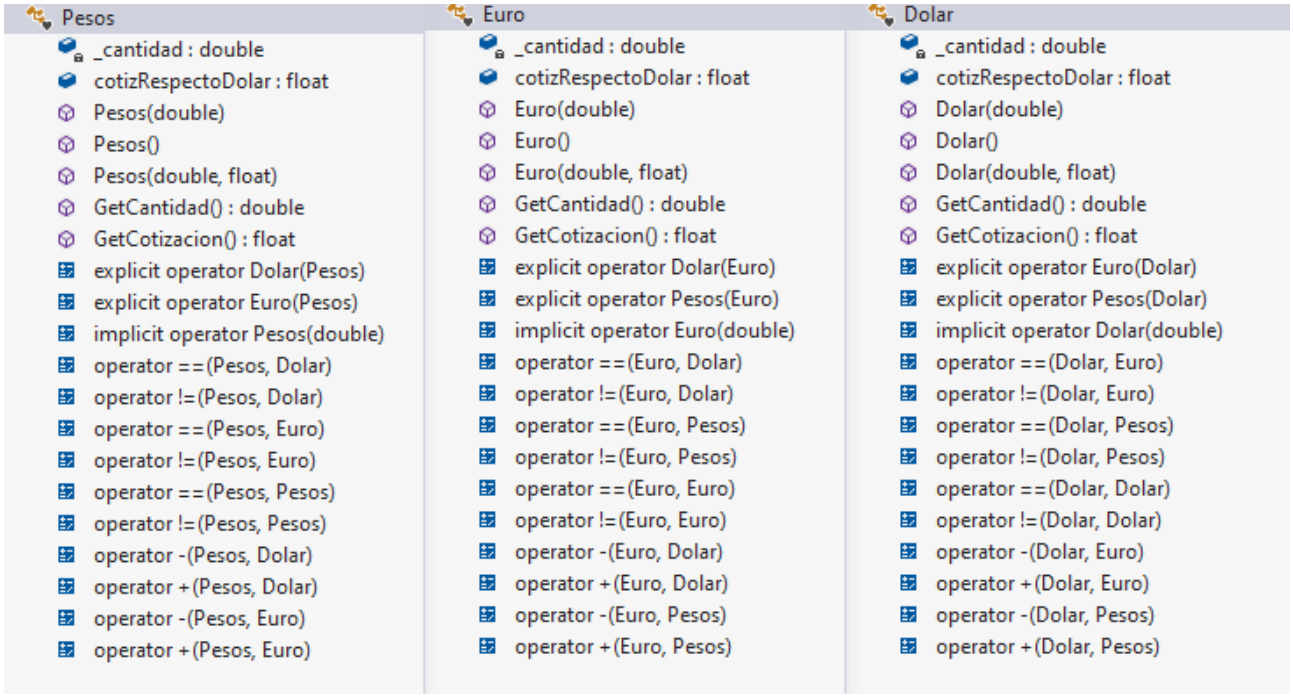
d. Colocar dentro del Main el código necesario para probar todos los métodos.

e. Los constructores privados le darán una cotización respecto del dólar por defecto a las clases.

f. Los comparadores == compararan cantidades.

g. Para operar dos tipos de monedas, se deberá convertir todo a una y luego realizar lo pedido. Por ejemplo, si quiero sumar Dólar y Euro, deberé convertir el Euro a Dólar y luego sumarlos.

h. Reutilizar el código. Sólo realizar las conversiones dentro de los operadores para dicho uso.



Forms

17) Tomar el Ejercicio 15. Realizar un Formulario con el siguiente formato:

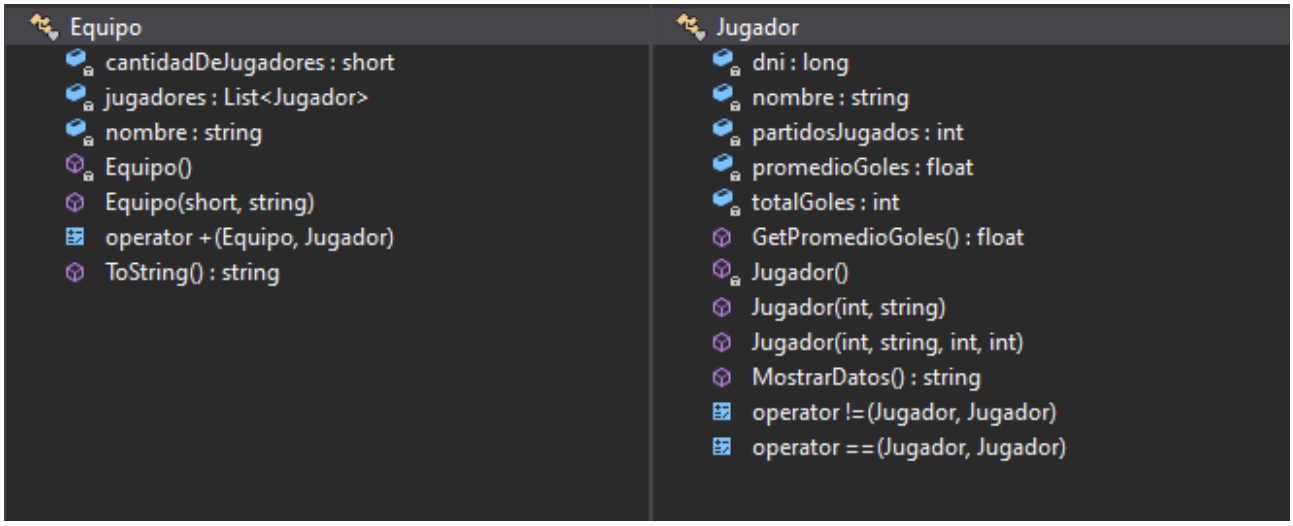
Arrays y Colecciones

18) Crear una aplicación de consola que cargue 20 números enteros (positivos y negativos) distintos de cero de forma aleatoria utilizando la clase Random.
a. Mostrar el vector tal como fue ingresado
b. Luego mostrar los positivos ordenados en forma decreciente.
c. Por último, mostrar los negativos ordenados en forma creciente.

19) Realizar el ejercicio anterior pero esta vez con las siguientes colecciones: ***Listas***.

Trabajo Práctico

Crear una Clase llamada **Jugador** y otra **Equipo** con la siguiente estructura:



Jugador:
a. Todos los datos estadísticos del Jugador se inicializarán en 0 dentro del constructor privado.
b. El promedio de gol sólo se calculará cuando invoquen al método GetPromedioGoles.
c. MostrarDatos retornará una cadena de string con todos los datos y estadística del Jugador.
d. Dos jugadores serán iguales si tienen el mismo DNI.
Equipo:
e. La lista de jugadores se inicializará sólo en el constructor privado de Equipo.
f. La sobrecarga del operador + agregará jugadores a la lista siempre y cuando este no exista aun en el equipo y la cantidad de jugadores no supere el límite establecido por el atributo cantidadDeJugadores.
Generar los métodos en el Main para probar el código.