Redefinición de Operadores

Ejercicio 1

Para cierta implementación que no viene al caso, el departamento de diseño ha detectado la necesidad de implementar un nuevo tipo de números a los que ha denominado "números curiosos". Un número curioso se caracteriza por tres coordenadas reales (a, b, c) que verifican $a^2 + b^2 + c^2 = 1$, salvo en el caso del número "cero" cuyas coordenadas son (0, 0, 0).

Sobre los números curiosos interesa realizar las siguientes operaciones:

1. **Suma** (otro:CURIOSO) definida mediante la fórmula...

$$(a_1,b_1,c_1)+(a_2,b_2,c_2)=\left(\frac{a_1+a_2}{\sqrt{(a_1+a_2)^2+(b_1+b_2)^2+(c_1+c_2)^2}},\frac{b_1+b_2}{\sqrt{(a_1+a_2)^2+(b_1+b_2)^2+(c_1+c_2)^2}},\frac{c_1+c_2}{\sqrt{(a_1+a_2)^2+(b_1+b_2)^2+(c_1+c_2)^2}}\right)$$

$$cuando\ (a_1+a_2)^2+(b_1+b_2)^2+(c_1+c_2)^2\neq 0\ \ y\ \ (0,0,0)\ \ en\,caso\,contrário$$

2. **Resta** (otro:CURIOSO) definida mediante la fórmula...

$$(a_1,b_1,c_1)-(a_2,b_2,c_2)=\left(\frac{a_1-a_2}{\sqrt{(a_1-a_2)^2+(b_1-b_2)^2+(c_1-c_2)^2}},\frac{b_1-b_2}{\sqrt{(a_1-a_2)^2+(b_1-b_2)^2+(c_1-c_2)^2}},\frac{c_1-c_2}{\sqrt{(a_1-a_2)^2+(b_1-b_2)^2+(c_1-c_2)^2}}\right)$$

$$cuando\ (a_1-a_2)^2+(b_1-b_2)^2+(c_1-c_2)^2\neq 0\ \ y\ \ (0,0,0)\ \ en\,caso\,contrário$$

- Crea la clase **NumeroCuriosos** con los atributos, propiedades y métodos que tú creas necesarios para su correcta implementación y prueba.
- Redefine los operadores necesarios para poder realizar la suma y resta de dos números curiosos.

Ejercicio 2

A partir del ejercicio de Cuenta Bancaria de excepciones, vas a redefinir como mínimo dos operadores que serán:

- != → que a partir de dos cuentas te devolverá true sin son distintas y false en caso contrario.
- == → que a partir de dos cuentas, te devolverá false sin son distintas y true en caso contrario.
- Para ambos operadores utilizaremos un solo método estático Iguales que será el que nos compruebe si las cuentas son iguales.

Nota: Para ver si dos cuentas son iguales tendrás que ver además del titular y saldo, si los números de cuenta son iguales (redefiniendo también su operador == y !=).

Ejercicio 3

- Crea la clase **Centímetros** y la clase **Metros**. Ambas poseerán un campo double. Redefine los operadores - y + que permitan desde ambas clases sumar o restar centímetros con metros.
- También debes redefinir los operadores explícitos Metros y double que permitan el cambio de tipo de forma CAST, para ello utiliza la palabra explicit.
- Haz lo mismo para la clase Centimetros.
- Implementa el programa necesario para probar todos los operadores.

Ejercicio 4

- Intenta construir dos clases: La clase **Euro** y la clase **Peseta** (la peseta era la antigua moneda oficial de España antes de ser reemplazada por el Euro).
- Tienes que hacer que los objetos de estas clases se puedan **sumar**, **restar**, **comparar** (operador == y ¡=), **incrementar** y **decrementar** con total normalidad como si fueran tipos numéricos, teniendo presente que 1 Euro + 166.386 pesetas = 2 euros.
- Además, tienen que ser compatibles entre sí y también con el tipo double.

Ejercicio 5 Ampliación

Diseñar la clase **Fracción**, que representa el conjunto de los números racionales.

Un número racional se representa por un numerador, que es un número **entero** y un denominador, que es un número **natural**.

Esta clase debe ofrecer como mínimo los siguientes métodos públicos:

1. Constructor que recibe el numerado y el denominador y los simplifica.
2. Sobrescribe el método **ToString**(), para que devuelva una cadena con formato **"num/den"**.
3. Redefinición del operador de cast implicito y explicito, para que devulevan el valor real de la fracción como double.
4. Propiedades para acceder y modificar el numerador y numerador simplificando la franción en caso que se modifique.
5. En todos los casos cuando el denominador al construir, al usar las propiedades, etc. sea cero. Generaremos una excepción DivideByZeroException con un mensaje indicándolo.
6. Redefiniremos las operaciones aritméticas simples, cuyo resultado será otra fracción **<u>en su forma simplificada</u>**:

   <u>Suma y Resta</u>
   $$\frac{c.n}{c.d}=\frac{a.n*b.d+a.d*b.n}{a.d*b.d}$$
   <u>Multiplicación</u>
   $$\frac{c.n}{c.d}=\frac{a.n*b.n}{a.d*b.d}$$
   <u>División</u>
   $$\frac{c.n}{c.d}=\frac{a.n*b.d}{a.d*b.n}$$

7. Aplicación para probar el programa.

¿Cómo realizar la simplificación?

Para simplificar una fracción primero hay que hallar el máximo común divisor del numerador y del denominador.

Crearemos el método … private void simplifica()

Crearemos el método de clase privado private static int mcd(uint n, uint d) que se encargará de esta tarea y para ello, empleará el algoritmo de Euclides, cuyo funcionamiento se muestra en el siguiente ejemplo:

Sea n = -1260 y d = 231,

1. Tomaremos el valor absoluto de n = Math.Abs(n)
2. En la primera iteración, se halla el resto r de dividir el primero n entre d. Se asigna a n el divisor d, y se asigna a d el resto r.
3. En la segunda iteración, se halla el resto r de dividir n entre d. Se asigna a n el divisor d, y se asigna a d el resto r.
4. Se repite el proceso hasta que el resto r sea cero.
5. El máximo común divisor será el último valor de d.

   1260 = 231 * 5 + 105

   231 = 105 * 2 + 21

   105 = 21 * 5 + 0
6. El máximo común divisor es 21.

$$\frac{-1260}{231}=\frac{\frac{-1260}{21}}{\frac{231}{21}}=\frac{-60}{11}$$