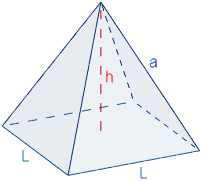
Parcial POO – Tema 1

1. Crear la clase ***Pirámide Cuadrada***, según lo siguiente ***(1 punto)***:
   1. Tiene como atributos ***lado*** de la base y la ***altura*** de tipo entero y de solo lectura.
   2. Tiene un atributo privado Apotema de tipo double que se calcula con la siguiente fórmula.
      1. ***Apotema***= donde ***h*** es la altura y ***L*** el lado de la base
   3. La clase tiene un ***constructor*** donde se pasan los valores de sus atributos, si los datos no son válidos, arrojar una excepción de tipo ***ArgumentException***.
   4. La clase debe poder informar su área y volumen.
      1. Área= AreaBase + AreaLateral.
         1. AreaBase= donde ***L*** es el lado de la base.
         2. AreaLateral=2\*L\*ap donde ***L*** es el lado de la base y ***ap*** es la apotema.
      2. Volumen= donde ***L*** es el lado de la base y ***h*** la altura.
   5. La clase debe tener un método para informar todos sus datos, cada dato en una línea diferente.
   6. Utilizar la clase en un proyecto de consola, hacer el ingreso de datos utilizando los métodos de MiDLL.
      1. *Lote de prueba:*
         1. *Lado de la base: 6*
         2. *Altura: 12*
         3. *Resultado:*
            1. *Volumen: 144*
            2. *Área: 184.43180252223578*
            3. *Apotema: 12.36931687685298*
2. Crear la clase estática ***ValidadorFecha*** que tiene un único método público ***Validar*** que recibe 3 parámetros de tipo entero, ***Dia***, ***Mes*** y ***Anio***, y debe informar si los mismos conforman una fecha válida o no. (***¡¡¡No utilizar el método TryParse del objeto DateTime!!!***). Se pueden utilizar las expresiones regulares, luego sí verificar con TryParseExact. ***(2 puntos)***:
   1. Debe tener un método privado para determinar si un año es bisiesto:
      1. Un año es bisiesto si es divisible por 4 y no son divisibles por 100, y los divisibles por 400.
   2. Usar la clase en un programa de consola para probar su funcionalidad.
3. Crear las clases ***Celsius*** y ***Fahrenheit*** con las siguientes características ***(2 puntos)***:
   1. Cada clase tiene un atributo privado ***grados*** de tipo ***float***.
   2. Las clases deben tener un método para informar dicho atributo.
   3. En el constructor se proporciona el valor del atributo ***grados***, que por defecto inicializa en ***5***.
   4. Hacer la sobrecarga ***implícita*** entre ***float*** y las temperaturas.
   5. Hacer la sobrecarga ***explícita*** entre las temperaturas.
   6. Sobrecargar los operadores de igualdad en las temperaturas. Los operadores de comparación == compararán los valores de los atributos de las clases.
   7. Se debe lograr que los objetos de estas clases se puedan sumar entre sí con total normalidad como si fueran tipos numéricos, teniendo en cuenta las siguientes equivalencias:
      1. F = C \* (9/5) + 32
      2. C = (F-32) \* 5/9
   8. ***Siempre que se pueda se debe reutilizar código***.
   9. Probar con el programa de consola, modificar el ingreso de datos utilizando los métodos de MiDLL.
4. Crear un programa que permita almacenar una cantidad determinada de números Armstrong ***(4 puntos)***.

**Número Armstrong:** Los números Armstrong son aquellos que cumplen una característica especial: la suma de las potencias n-ésimas de sus dígitos es igual al propio número.

***Clase NumeroArmstrong***

* 1. Tiene un atributo ***Valor*** de tipo ***entero*** de lectura y escritura a través de propiedades.
  2. ***Se provee el método para validar el número armstrong***:
  3. Sobrescribir los métodos ***heredados*** de ***Object***.
  4. Sobrecargar los operadores == y != para comparar 2 números por su ***Valor***

***Clase RepositorioNumerosArmstrong***

1. El atributo ***cantidad*** es de tipo ***entero*** y ***privado***.
2. Debe tener un ***método público*** para informar dicho atributo.
3. Otro atributo privado de tipo array de ***NumeroPerfecto***, que acepte nulos, donde se almacenarán los números perfectos ingresados.
4. Definir 2 constructores:
   1. Uno que tome un parámetro que inicializará el atributo cantidad.
   2. Otro que no reciba parámetros y por defecto inicializará el atributo cantidad en 5. En ambos casos se inicializa el array.
5. El repositorio debe definir métodos privados para:
   1. Establecer si está completo.
   2. Establecer si está vacío.
   3. Informar si un elemento ya existe.
6. Crear métodos para agregar y quitar un número perfecto del repositorio, los mismos devolverán una ***tupla<bool, string>***, indicando si se pudo realizar la operación y una descripción de la misma o del error.
7. Definir un método público que permita acceder a un elemento, el mismo puede devolver un ***nulo***; si el índice estuviere fuera de rango, ***arrojar una excepción***.
8. Definir un método que devuelva un ***string*** con todos los números que tuviere, en el caso de un elemento nulo, mostrar ***“Elemento Nulo”***; si el vector está vacío, mostrar el mensaje ***“No hay elementos almacenados todavía”***.
9. Definir un método que devolverá una ***tupla<bool, int>***, que informe si un número forma parte del repositorio y en ***qué posición se encuentra***, caso contrario informar ***falso*** y -1 como posición.
10. Crear una ***sobrecarga implícita*** del repositorio a un ***número entero***, devolviendo la ***suma*** de los números perfectos.
11. Utilizar las clases en un programa de consola con menú para realizar todas las operaciones pertinentes.
12. Testing ***(1 punto)***:
    1. Crear un proyecto de testing para probar todos los métodos de la clase del Ejercicio 1.
    2. Crear un proyecto de testing para probar al menos 4 métodos del repositorio del Ejercicio 4.

***Pautas para la resolución y entrega:***

1. Solución en Blanco: Nombre y Apellido del alumno.
2. Cada ejercicio en una carpeta con el número de ejercicio.
3. Cada proyecto en su capa correspondiente y con el nombre correspondiente al ejercicio que está resolviendo.
4. Subir la solución a su repositorio en Git.
5. Adjuntar el link al classrrom en tiempo y forma, no serán tenidos en cuenta los trabajos enviados por otros medios y formatos.
6. Utilizar MiDLL para la captura de datos por consola, ya sea en la carpeta compartida o compilada.

***Pautas para la corrección:***

1. Cada ejercicio entregado NO debe tener errores de compilación, para su corrección.
2. Errores en las fórmulas serán tenidos en cuenta y dicho punto se considerará mal resuelto.
3. Se podrá solicitar al alumno la defensa oral de su trabajo.