Curso de programación en C#
Febrero de 2020

Serie de ejercicios

1. Programa que permite calcular la suma de los primeros n números naturales, siendo n un número natural ingresado por el usuario.

2. Realizar un programa en el que se le pida al usuario dos números del 1 al 9, num1 y num2. Después va a imprimir todos los números naturales del 1 al 100, sin embargo, cuando un número sea múltiplo de num1 o num2 o contenga alguno de estos números, va a imprimir 'clap'.

3. Escribir un programa que pida al usuario una cadena de texto e imprima la misma cadena de texto, pero antes de cada vocal, agregue una f.
   Ejemplo:
   "Mi nombre es Ana"
   "Mfi nfombrfe fes fAnfa"

4. Realizar un programa que imprima la serie de fibonacci hasta el elemento n que especifique el usuario.

5. Se debe simular una agenda telefónica. Cuando inicie el programa se debe desplegar un menú con las opciones:

   - Agregar contacto
   - Eliminar contacto
   - Mostrar contacto
   - Salir

   Los contactos van a ser almacenados en un diccionario (investiguen la colección Dictionary) en donde las llaves son los nombres de los contactos y sus valores van a ser los teléfonos.

   Hacer pruebas en en el método Main.

6. Se debe diseñar una clase CuentaBancaria que va a tener los métodos mostrarInformación, depósito y retiro, como atributos, cada objeto va a tener un saldo y un nombre. El método mostrar información va a imprimir el nombre de la cuenta y el saldo, el método depósito va a agregar una cantidad al saldo de la cuenta y va a mostrar información. El método retiro va a retirar dinero de la cuenta, pero antes tendrá que comprobar que se cuente con el dinero suficiente, terminando el retiro va a mostrar información. Cada vez que se cree un objeto, va a mostrar información de la cuenta. Se debe crear dos objetos y llamar sus métodos para probarlos.

7. Se debe diseñar una clase NumeroComplejo, que va a tener como atributos parteReal y parteImaginaria, se debe definir un método imprimir que va a imprimir el número complejo con formato. Se debe definir un método para la suma de dos objetos de la clase NumeroComplejo que retorne un nuevo número complejo que sea la suma de los otros dos. A continuación se debe crear dos números complejos y probar sus métodos.

8. Crear una clase llamada Auto que contenga los siguientes elementos:

   - Al menos 3 atributos de diferentes tipos.
   - Constructor por defecto y sobrecarga de constructor.
   - Métodos *getters* y *setters* para encapsulamiento.
   - 3 métodos que pueda realizar un automóvil.

   Instanciar 3 objetos y con ellos mandar a llamar a cada uno de los métodos realizados (Hacer uso también de los métodos set para poder cambiar alguno de sus atributos y de los métodos get para poder imprimir sus datos).

9. Haz una pequeña investigación sobre los métodos que se encuentran en la clase *Math* y para qué sirven. Crea una clase llamada Cículo y otra con el nombre de la figura geométrica de tu preferencia. Cada clase deberá tener sus respectivos atributos y métodos de tal manera que ocupes mínimo tres métodos de la clase *Math*.
   Nota: la investigación se puede incluir como comentario en su archivo.

10. Realiza una clase llamada "Persona"que tenga los atributos: nombre edad, estatura(metros) y peso(Kg). El usuario ingresará los datos dichos datos. Crea algunos métodos (mínimo 3) y mandalos llamar en una clase principal.

11. Elabora un programa que simule una calculadora de matrices (utilizando arreglos bidimensionales). El tamaño de la matriz (arreglo) deberá ser ingresado por el usuario, así como los valores contenidos en ella. En este caso sólo se soportarán matrices cuadradas (nxn). Las operaciones que deberá contender la calculadora son:

    - suma
    - resta
    - multiplicación

    Se puede utilizar cualquier tipo de dato (int, double, float). El programa debe contener un menú que me permita elegir entre las operaciones de la calculadora y cuando se hagan las operaciones, me debe mostrar las matrices que operan, así como el resultado. Estas matrices deben mostrarse en su respectivo formato, es decir, cuadradas. Sin importar lo que el usuario ingrese el programa no debe "morir", usen excepciones para controlar el ingreso de datos del usuario.

12. Una persona se dirige al *BecarioMart* a realizar sus compras semanales. Dicha persona no sabe cuántos productos va a comprar y tampoco sabe su valor. Para poder ayudarlo deberás de implementar dos Listas. Una contiene los nombres de los productos y otro contiene los precios

de los productos. Tanto el nombre como el precio, deberán ser ingresados por el usuario y en seguida, deberán de ser agregados a las listas. Al momento de pagar le aparece una lista de todos los productos que ha llevado con su respectivo precio, sin embargo, se da cuenta de que sólo tiene $500.00, por lo que, si ha excedido su cuenta, deberá elegir unos productos y dejar otros. Para lograr eso, debe poder acceder al índice del producto y así poder elegirlo para eliminarlo de su lista, hasta que finalmente la cuenta sea menor o igual a los $500.00.

El programa debe preguntar al usuario si desea agregar un producto a su lista, si es así deberá ingresar tanto el nombre como el precio a las Listas. En caso de que ya no quiera agregar más productos, le aparecerá en pantalla la lista de todos los productos que ha llevado (nombre y precio) así como la suma de los precios. Cuando aparezca la lista, deberá tener dos opciones:

- Comprar.
- Dejar productos.

Para la primera opción deberás de tener validaciones para ver si el dinero que tiene es suficiente. Si el dinero es suficiente, deberá imprimir un mensaje que diga "Gracias por su compra!", en caso contrario debe mandarlo a la opción "Dejar productos". Si el usuario elige la opción 2 deberá poder ver los índices de los productos en las listas para así poder decidir cuál eliminar. Una vez eliminados los productos, debes imprimir de nuevo la lista de productos a comprar con su precio y darle a elegir de nuevo entre las dos opciones.

13. Se tendrá una clase llamada *Carro* que tendrá de atributos peso y altura junto con un atributo llamado " encendido" de tipo booleano inicilizado en *false*. En el constructor deberás indicarle los valores peso y altura.La misma clase contará con los métodos encender y apagar, que cambiarán el estado del atributo " encendido" y desplegarás un mensaje indicando si se encendió o se apagó.

    - Crear métodos get para poder obtener su peso y altura.
    - Se contará con un método llamado " Estado"para verificar si el auto se encuentra encendido o no.
    - El último método de la clase, es toString() el cual nos servirá para imprimir un mensaje, este mensaje deberá indicar el peso y altura del Carro.

    Crear dos clases más que hereden de Carro, con nombres como CarroBWM, CarroVW, etc. Esta clase tendrá un atributo llamado modelo. Implementa su constructor y también implementa su método toString() para que pueda imprimir todos sus atributos. En una clase de prueba(en el método Main), crea objetos de las dos clases hijas y manda a llamar todos sus métodos. Al final deberá verse una impresión como la siguiente, para los dos objetos que hayas creado:

    >El carro está apagado
    >El carro está encendido
    >Está encendido
    >Tengo turbo
    >El peso es: 100,000.
    >La altura es: 1.90.

>El modelo es: BMW

14. Implementa una interfaz con los métodos que tú desees. Crearás 3 clases diferentes que puedan hacer uso de esta interfaz, así como una clase de prueba donde podrás probar su uso. Incluir en comentarios cuál es el objetivo de la interfaz.

15. Crea una calculadora que haga operaciones entre dos números. Las operaciones que deberá soportar serán suma, resta, multiplicación y división. El programa debe de contener un menú que me permita elegir cualquier opción y la última de ellas debe ser para salir. Si se termina una operación debo de volver al menú de inicio. No debe haber forma de salirse del programa a menos que sea con la opción salir. Considerar todas las excepciones posibles e implementarlas para evitar errores en tiempo de ejecución. Considera: División entre cero, números demasiado grandes, cadenas en vez de números, entre otras que a ti se te puedan ocurrir.

16. Realiza un programa que haga un claro ejemplo de Polimorfismo, estas clases pueden ser de tu elección. Recuerda que polimorfismo no necesariamente es hacer una interfaz o hacer una clase abstracta.

17. Realiza un programa de temática libre en donde puedas utilizar los 4 pilares de la Programación Orientada a Objetos.

18. Realizar una investigación del manejo de archivos en C# y hacer un programa libre que implique el manejo de archivos.

Notas:

- TODOS LOS PROGRAMAS DEBEN ESTAR DOCUMENTADOS PARA HACER MÁS FÁCIL SU COMPRENSIÓN.
- Mandar al correo el link del repositorio donde se encuentra la serie.
- La serie se envía antes de que comience la clase del martes.
- Recuerden que estos ejercicios son para que practiquen sus conceptos. Si tienen alguna duda recuerden que Google es su mejor amigo y pueden encontrar un montón de información ahí.
- Cualquier duda contáctenme pero antes intenten resolver esa duda por su cuenta recuerden que una de las características de un buen programador es que sabe investigar y aprender por su cuenta.
- La serie es individual, no me molesta que se ayuden o se pregunten entre ustedes, después de todo los proyectos de la vida real siempre son en equipo, pero si encuentro dos series iguales o detecto algún tipo de plagio les bajaré puntos sobre calificación final, ayúdense entre ustedes pero no se copien las soluciones.