GUÍA DE TRABAJOS PRÁCTICOS

TRABAJO PRACTICO NÚMERO 1

Nos solicitan crear un programa que maneje una lista de alumnos.
Un alumno posee:

Características

Legajo	int	lectura/escritura
Nombre	string	lectura/escritura
Apellido	string	lectura/escritura
Fecha_Nacimiento	date	solo escritura
Fecha_Ingreso	date	solo escritura
Edad	int	solo lectura
Activo	boolean	lectura escritura
Cant_Materia_Aprobadas	int	solo escritura

Métodos

Antigüedad	int	Retorna la cantidad de años/meses/días que hace que el alumno asiste a la institución. El método posee un parámetro que indica en que unidad (años, meses, días) se desea el retorno. Para el cálculo se considera solo año/mes cumplido, que hace que el alumno acude a la institución. Par determinar el número de años se considera la fecha actual y la fecha de ingreso.
Materias_No_Aprobadas	int	Retorna la cantidad de materias no aprobadas a la fecha sabiendo que la carrera posee 36 Materias y considerando la cantidad de materias aprobadas.
Edad_De_Ingreso	int	Retorna la edad a la que el alumno ingresó a la institución considerando la fecha de nacimiento y la fecha de ingreso.

Constructores:

1. Constructor sin parámetros
2. Constructor con todos los parámetros que permiten inicializar las propiedades.

Finalizadores:

1. Que cuando el objeto queda liberado muestre una leyenda indicando el Legajo, Nombre y Apellido del Alumno.

Nos solicitan que la GUI (interfaz gráfica del usuario) permita visualizar la lista de los alumnos ingresados en una lista del tipo DataGridView.

La GUI debe tener botones para:

Agregar un alumno a la lista.
Borrar el alumno seleccionado de la lista.
Modificar el alumno seleccionado de la lista.

La GUI debe tener cajas de texto para ver del alumno seleccionado en la grilla. Estas cajas de texto se deben actualizar cuando el usuario de desplaza por las filas de la grilla con el cursor o les hace click

Antigüedad
Materias_No_Aprobadas
Edad_De_Ingreso

Recursos a considerar para la resolución:

1. **Tema** Listas de programación 1. **Sugerencia**: Usar **List(of…)**

2. **Tema** Clases, Instancias, Propiedades, Métodos, Constructores, Finalizadores de la Unidad I de Programación Orientada a Objetos. **Sugerencia**: Ver el Orientador y la bibliografía recomendada.

3. **Tema** DatagridView. Grilla vista el programación 1. **Sugerencia**: https://docs.microsoft.com/es-es/dotnet/framework/winforms/controls/datagridview-control-windows-forms

TRABAJO PRACTICO NÚMERO 2

Nos solicitan crear un programa que maneje una lista de personas y los autos que estas poseen. Nos indican que una persona puede ser dueño de más de un auto. Pero que los autos poseen como máximo un titular.

Una persona posee:

Características

DNI	string	lectura/escritura
Nombre	string	lectura/escritura
Apellido	string	lectura/escritura

Métodos

Lista_de_autos	List<auto>	Retorna la lista de autos que a persona es dueño.
Cantidad_de_Autos	int	Retorna la cantidad de autos que posee la persona

Constructores: Un constructor con todos los parámetros que permiten inicializar las propiedades.

Finalizador: Que cuando el objeto queda liberado muestre una leyenda indicando el DNI de la Persona.

Un auto posee:

Características

Patente	string	lectura/escritura
Marca	string	lectura/escritura
Modelo	string	lectura/escritura
Año	string	lectura/escritura
Precio	decimal	lectura/escritura

Métodos

Dueño Persona Retorna el dueño del auto.

Constructores: Un constructor con todos los parámetros que permiten inicializar propiedades.

Finalizador: Que cuando el objeto queda liberado muestre una leyenda indicando la Patente del Auto.

Nos solicitan que la GUI (interfaz gráfica del usuario) permita visualizar en grillas DataGridView:

1. La lista de las presonas. (Grilla 1)
2. La lista de los autos. (Grilla 2)
3. Los autos de la persona seleccionado en la grilla 1. (Grilla 3)
4. Una grilla (Grilla 4) con los siguientes datos y en el siguiente orden de columnas, para cada auto de la grilla 2. Marca, Año, Modelo, Patente, DNI del dueño, “Apellido, Nombre”del dueño en una misma columna. (Grilla 4)

La GUI debe tener botones para:

a. Agregar personas y autos.
b. Borrar personas y autos.
c. Modificar personas y autos.
d. Asignarle a la persona selecionada en la grilla 1 el auto seleccionado en la grilla 2

La GUI debe tener un label donde se observe el valor correspondiente a la suma de los precios de los autos de la persona seleccionada en la grilla 1.

También nos solicitan que cada valor ingresado sea validado de manera que no se introduzcan datos inconsistentes.

EJERCICIOS VARIOS

1. Desarrollar un programa que posea una estructura de clases donde se puedan observar dos métodos y el constructor sobrecargados.

2. Desarrollar un programa que posea una estructura de clases donde se pueda observar una propiedad de solo lectura, una propiedad de solo escritura, una propiedad de escritura-lectura, una propiedad de predeterminada y una propiedad con argumentos.

3. Desarrollar un programa que posea una clase que aplique un destructor que finalice el ciclo de vida de un objeto que se instanció en el constructor.

4. Desarrollar un programa que posea una clase que tenga propiedades, métodos y sucesos. Los sucesos deben tener suscripciones manuales realizadas por el programador a funciones de la clase donde se encuentra instanciado el objeto que posee los sucesos.

5. Desarrollar un programa que posea una clase que contenga al menos dos campos, tres métodos, un constructor y dos sucesos estáticos. Comente que característica le otorga esta característica a cada miembro.

6. Desarrollar un programa donde se observe claramente el uso de los miembros de base en un entorno de herencia. Enfatice las particularidades al usarlo en los constructores y finalizadores. Demuestre en el mismo programa el uso de this. Establezca las diferencias y en qué caso se justifica utilizar cada uno.

7. Desarrollar un programa que posea al menos una clase abstracta, un método virtual, una clase sellada y una clase anidada.

8. Desarrollar un programa que posea una estructura de clases donde se puedan observar dos métodos y el constructor sobrecargados.

9. Desarrollar un programa que posea una estructura de clases donde se pueda observar una propiedad de solo lectura, una propiedad de solo escritura, una propiedad de escritura-lectura, una propiedad de predeterminada y una propiedad con argumentos.

10. Desarrollar un programa que posea una clase que aplique un destructor que finalice el ciclo de vida de un objeto que se instanció en el constructor.

11. Desarrollar un programa que posea una clase que tenga propiedades, métodos y eventos. Los eventos deben tener suscripciones manuales realizadas por el programador a funciones de la clase donde se encuentra instanciado el objeto que posee los sucesos.

12. Desarrollar un programa que posea una clase que contenga al menos dos campos, tres métodos, un constructor y dos eventos estáticos. Comente que característica le otorga esta característica a cada miembro.

13. Desarrollar un programa donde se observe claramente el uso de los miembros de base en un entorno de herencia. Enfatice las particularidades al usarlo en los constructores y finalizadores. Demuestre en el mismo programa el uso de this. Establezca las diferencias y en qué caso se justifica utilizar cada uno.

14. Desarrollar un programa que posea al menos una clase abstracta, un método virtual, una clase sellada y una clase anidada.

15. Trabajo práctico integrador 2:

    NOTA: Aplicar las buenas prácticas de la POO

    Una universidad nos encarga un programa para administrar y conocer que materias se encuentra cursando un alumno, las materias que aprobó y las que no.
    Para ello nos indican que de los alumnos desean mantener el legajo, su nombre y su apellido. El legajo de los alumnos es numérico.
    De una materia el código y su denominación. En caso que el alumno la haya cursado también la nota que obtuvo.
    En la interfaz del usuario se debe poder ver en una grilla a todos los alumnos que se dan de alta.
    Al seleccionar un alumno se debe poder observar en otras tres grillas: las materias que el alumno está cursando, las materias aprobadas y las materias desaprobadas.
    El usuario podrá ingresar la cantidad de alumnos que desee, así como desde la misma interfaz, agregarle materias a cursar.
    Se podrá seleccionar una materia de las que tiene asignadas para cursar, para indicar que ha sido calificada con una nota de 1 a 10. Para considerar que una materia está aprobada la nota debe ser de 4 o más puntos. Valide que estos valores se respeten.
    También nos informan que hay dos tipos de materias: materias básicas y materias especializadas. Un alumno no puede cursar más de una materia especializada a la vez. Un alumno puede cursar varias veces una materia desaprobada. Un alumno no puede cursar nueva- mente una materia aprobada.
    Las materias especializadas al colocarles la nota de cursada si esta está entre 6 y 9 puntos se le adiciona un punto en virtud de la dificultad de las mismas. Si se le asigna un 10 esta queda como máxima nota.

Desean que al seleccionar un alumno de la grilla podamos conocer el promedio de materias aprobadas y el promedio general (incluyendo las materias no aprobadas).
Si el alumno arroja un promedio general igual o superior a 9 puntos se desencadenará un evento que indique lo ocurrido.

16. Trabajo práctico integrador 3:

NOTA: Aplicar las buenas prácticas de la POO

Nos solicitan crear un programa que permita administrar los inversores que compran y venden acciones para la empresa Invest S.A. Los inversores son personas que se las identifica por un legajo que se genera aleatoriamente en el sistema al momento de darlos de alta. Además, se solicita que el usuario ingrese el apellido, nombre y dni. Los inversores invierten en acciones de empresas. Cada inversor puede tener inversiones en distintas acciones y de cada una de ellas poseer distintas cantidades. Las acciones se identifican por un código compuesto por tres partes divididas por guiones. La primera parte son caracteres alfabéticos en mayúscula que identifican a la empresa. La segunda cuatro números que son de validación. La tercera una combinación de cuatro caracteres para un control interno, el primero y tercero son letras y el segundo y el curto son números (BGAL-4148-J4T3). Además, las acciones poseen una denominación, la cotización actual y la cantidad emitida, la cual nunca podrá ser superada por las compras de los inversores. Como se mencionó anteriormente los inversores pueden comprar y vender acciones. La inversión de un inversor se calcula por cuanto dinero posee considerando las acciones que posee y su cotización actual. Cada vez que una acción cambia su cotización, el inversor debe enterarse por medio de un evento que se desencadena en la acción que cambió su cotización. El evento lleva en su argumento personalizado la cotización actual. También al realizar la ingeniería de requerimientos descubrimos que hay dos tipos de inversores. Un grupo está representado por los inversores comunes y otro por los inversores Premium. Los primeros cada vez que realizan una operación de compra o venta se les cobra una comisión del 1% sobre el total. Los Premium también pagan el 1% por compras y ventas hasta 20.000 pesos. Superada esa base sobre el resto abonan la mitad del porcentaje consignado anteriormente.
Nos solicitan que la GUI (interfaz gráfica del usuario) permita visualizar en grillas DataGridView:

1. La lista de todos los inversores (todos sus datos). (Grilla 1)

2. Las acciones que posea el inversor seleccionado en la Grilla 1 (todos los datos de las acción + cuantas acciones posee el inversor + el valor total de la inversión [total acciones * cotización]). (Grilla 2)

3. Todas las acciones (todos sus datos) en la que el inversor puede invertir.(Grilla 3)

La GUI debe tener botones para:

a. Agregar inversores y acciones.

b. Borrar inversores y acciones.

c. Modificar inversores y acciones.

d. Que un inversor pueda comprar acciones.

e. Que un inversor pueda vender acciones.

Nos solicitan:

Validar los datos ingresados.

Utilizar Try Catch y generar las excepciones personalizadas necesarias para informar sobre problemas de validación, imposibilidad de compras por no haber acciones disponibles, borrados o modificaciones de datos inexistentes etc.

Colocar constructores y destructores a las clases.

Utilizar en cada clase la Intefaz IDisposable.

Aplicar Herencia donde corresponda.

Aplicar Agregación y composición donde corresponda.

Aplicar Asociación donde corresponda.

Clonar donde corresponda.

Que los Accionistas se puedan ordenar de manera ascendente y descendente por (Legajo, Nombre, Apellido, DNI). Se deben utilizar interfaces.

Que las Acciones se puedan ordenar de manera ascendente y descendente por (Legajo, Nombre, Apellido, DNI). Se deben utilizar interfaces.

Que se pueda obtener el código de la acción seleccionada en la grilla 3 iterándola acción con un foreach de manera que en cada iteración retorne cada una de las partes que lo componen sin los guiones. Se debe utilizar interfaces.

La GUI debe tener un label donde se observe el total invertido por el inversor seleccionado en la grilla 1.
La GUI debe tener 4 labels donde se observe el total ganado por la empresa en concepto de comisiones cobradas por compras y ventas. La información que se observa en cada uno de ellos es:

a) Label1: El total recaudado por operaciones de los clientes comunes.

b) Label2: El total recaudado en concepto de comisiones por operaciones delos clientes premium por los ingresos correspondientes hasta 20.000.

c) Label3: El total recaudado en concepto de comisiones por las operaciones de los clientes premium por los ingresos correspondientes que superan los20.000.

d) Label4: el total general percibido en concepto de comisiones.

Recursos a considerar para la resolución:

1. Tema Listas de programación 1. Sugerencia: Usar List(of...)
2. Tema Clases de vista.
3. Tema Clases, Instancias, Propiedades, Métodos, Constructores, Finalizadores, Eventos e Interfaces. Sugerencia: Ver los Orientador y la bibliografía recomendada.
4. Tema Try ... Catch

5. Tema DatagridView. https://docs.microsoft.com/es-es/dotnet/framework/winforms/controls/datagridview-control-windows-form