

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Desarrollo de Interfaces				CURSO: 2º
	PROTOCOLO:	Ejercicios repaso	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

Ejercicios repaso

Evaluación continua

Ejercicio 1 (BASE)

Se desea realizar un gestor de cuerpos celestes en consola. Para ello se pide generar el siguiente esquema de clases:

a) **Clase Astro** con propiedades privadas: nombre (String) y radio.

Al guardar el nombre lo hará en mayúsculas y eliminando posibles espacios en los extremos del nombre.

Habrà una sobrecarga del get con un parámetro tipo carácter que devuelve el nombre con sus letras separadas por ese caracter. Por ejemplo si se llama getNombre('_') a un astro denominado "SOL" devuelve "S_O_L".

Al guardar el radio se comprueba que sea positivo, si no fuera así lanza la excepción RadioNegativoException creada por ti.

Sobreescribe Equals de forma que se considera que dos astros son iguales si son iguales sus nombres y además son de la misma clase.

Sobreescribe ToString de forma que devuelva el nombre según getNombre y el radio con 2 decimales.

b) **Clase Planeta** que hereda de Astro y dispone de una propiedad privada booleana denominada gaseoso con set y get.


Otra propiedad pública que será una colección de Astros que serán los satélites.

Tendrá dos constructores, uno que inicialice los miembros nombre, radio y gaseoso con parámetros y la colección de satélites vacía.

El otro constructor sin parámetros que inicializa a "" nombre, 0 radio y false la propiedad gaseoso llamando al primer constructor.

c) En el programa principal (Estará en la clase **Program**) se crea una colección de Astros y el siguiente menú:

- Añade Planeta: Pregunta si es gaseoso y pide el nombre y radio. También pregunta cantidad de lunas y se introducirán sus nombres y radios.
- Añade otro astro: Pide su nombre y radio.
- Mostrar datos: Muestra toda la colección detectando si es Astro o Planeta para mostrar todos sus datos. En el caso de Astro simplemente llama a ToString() y lo muestra, en el caso de Planetas muestra todos los datos y en particular

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Desarrollo de Interfaces				CURSO: 2º
	PROTOCOLO:	Ejercicios repaso	AVAL:		DATA:	
	UNIDAD COMPETENCIA					

llamando a `getNombre` con parámetro `.`

- Elimina repetidos. Busca Astros/Planetas iguales y elimina todas las apariciones menos la primera.
- Salir. Se mantiene en el menú hasta que se selecciona esta opción.

Ejercicio 2 (BASE):

En un instituto se desea realizar una serie de estadísticos con las notas de los alumnos. Para ello, y como fase inicial de un futuro proyecto, se pide la realización de un programa de simulación de notas de dicho instituto. Para rellenarla se usarán notas aleatorias entre 0 y 10 sin decimales.

Se debe crear a continuación un menú (estará en la clase *Usuario* como se explica más abajo) con las opciones:

- Visualizar tabla completa
- Calcular la media de notas de toda la tabla.
- Media de un alumno
- Media de una asignatura
- Visualizar notas de un alumno
- Visualizar notas de una asignatura
- Nota máxima y mínima de un alumno
- Tabla solo de aprobados

Los nombres de los alumnos estarán en un vector y los nombres de las asignaturas en otro.


Debes estructurar el programa en al menos tres clases:

- **Clase Aula** donde se encuentra el array bidimensional de notas privado, un vector de string para los nombres de alumnos y otro para los nombres de asignaturas. Estarán aquí todos los métodos de proceso de datos (obtención de medias, devolución de una fila, etc.).

En concreto haz una función única que coloque el máximo y el mínimo de un alumno en parámetros por referencia. No devolverá nada.

Tendrá un constructor al cual se le pasan dos vectores: de nombres de alumnos y otro de nombres de asignaturas. El primero establecerá el número de filas de la tabla y el segundo el de columnas, dicha tabla se rellenará con notas aleatorias entre 1 y 10.

La función que devuelva la tabla de aprobados hazla que devuelva un hashtable usando como clave el nombre del alumno y como valor un vector con las notas del mismo.

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Desarrollo de Interfaces				CURSO: 2º
	PROTOCOLO:	Ejercicios repaso	AVAL:		DATA:	
	UNIDAD COMPETENCIA					

Las medias se devuelven siempre como doubles.

No debe tener interfaz de usuario.


- **Clase Usuario** donde tendrás un objeto tipo Aula y se organizará todo el interfaz de usuario para la aplicación. El menú principal estará en una función de esta clase.

- Una tercera clase con el main que simplemente cree un objeto del tipo Usuario y ejecute su método principal que contendrá el menú.

Ejercicio 3:

Crear un formulario con las siguientes características:

- Habrá un botón salir que termina la aplicación. También se puede salir con la tecla ESC
- Antes de salir se pedirá confirmación al usuario.
- Habrá 3 TextBox y un botón Color. En las 3 textbox se puede meter números RGB (0-255) y al pulsar el botón se cambia el color del fondo. También si se pulsa ENTER.
- Otro TextBox en el que se escribe el Path de una imagen, esta se cargará al pulsar otro botón asociado de fondo en el actual.
- Cambiar el icono que viene por defecto
- El formulario será normal pero su tamaño no se podrá modificar. Tampoco tendrá controles de maximización ni minimización. Aparecerá siempre centrado en la pantalla y no se verá en la barra de tareas. El título será Colores e Imágenes.
- El puntero del ratón será una mano señalando.
- Cada vez que el ratón pase por encima de algún botón este cambiará de color, restaurando el color original cuando el ratón salga del mismo.
- El orden de tabulación debe ser adecuado.
- Cada vez que el ratón esté dentro del formulario se debe indicar la coordenada en el título (p. ej: x:100, y:212). También debe hacerlo aunque se esté encima de algún componente. Cuando el ratón sale del formulario se restablece el título.
- Si se pulsa la tecla T, entonces se restaura el título del formulario.

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Desarrollo de Interfaces				CURSO: 2º
	PROTOCOLO:	Ejercicios repaso	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

Ejercicio 4:

Realiza un programa con al menos 2 textbox, un botón y 4 radiobutton que permitan seleccionar entre suma, resta, multiplicación y división. Al pulsar en un radiobutton se selecciona la función a ejecutar mediante una tabla hash con clave el campo Text del radiobutton y valor el delegado. Al pulsar el botón ejecuta dicha función con los operandos que haya en los textbox.

Habrás también una etiqueta entre los dos TextBox donde aparece el símbolo de operación que cambiará al cambiar los RadioButton. Las funciones de operación realízalas como funciones lambda.


En la barra de título se verá el tiempo de uso en minutos y segundos (formato mm:ss).

Debe respetarse el orden de tabulación y pedirá confirmación al salir.

Ejercicio 5:

Coloca 2 ListBox (la primera permite multiselección, la segunda no), 4 botones, 1 TextBox, 2 label

- Cada botón es una acción (añadir, quitar, traspasar (2 botones))
- Se añade el texto que hay en la TextBox a la lista 1. Al darle tanto al botón Añadir como a la tecla Enter.
- Se eliminan los elementos seleccionados (comprobar si hay alguno) de la lista 1
- Se traspasan los elementos seleccionados de una lista a la otra con los botones. Deben quedar en el mismo orden (uso de Insert en posición 0 en vez de add)
- En una label indica cuántos elementos hay en la lista principal y en la otra los índices de los seleccionados.
- El título del formulario debe ser animado, apareciendo una letra del título empezando por el final (efecto scroll) cada 200 ms y cuando se complete, volviendo a empezar. Cada 400 ms debe cambiar también el icono del formulario. Se debe hacer sólo con un timer.
- Meter ToolTip para los distintos componentes. En el caso de la lista secundaria, el ToolTip debe mostrar la cantidad de elementos que hay.
- Orden adecuado de tabulación saltándose el botón Quitar.
- Habrá un menú Lista 1 con las opciones de Añadir, traspasar y quitar. Otro Lista 2 solo con la opción traspasar. Debe tener navegación con ALT y atajos de teclado.

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Desarrollo de Interfaces				CURSO: 2º
	PROTOCOLO:	Ejercicios repaso	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

Ejercicio 6:

Realizar un formulario que simule el teclado de un móvil mediante un TextBox y 12 botones (dígitos, * y #).

Nada más arrancar se pedirá mediante un formulario modal un pin de 4 dígitos. Si se mete mal 3 veces, la aplicación se cierra, si no da paso al programa principal.

Los botones deben ser creados e inicializados en tiempo de ejecución en el evento Load del Formulario. Las pulsaciones de los botones escriben su contenido en el TextBox. Además cuando el ratón pase por encima de cada uno, este cambiará de color resaltándolo y volviendo al color original al salir. Si se aprieta, cambiará a un tercer color que ya no se restaura. Debe existir también un botón de Reset (este creado en tiempo de diseño) que borra el TextBox y deja todos los botones del color original.

Tendrá un pequeño menú con dos encabezados: Archivo y Acerca de... En Archivo habrá las opciones Grabar número, Ver números, Reset, separador y Salir.

Grabar número añade a un archivo de texto seleccionado mediante SaveDialog (txt o todos los archivos) el número que haya en el TextBox (la única comprobación es que haya algo).

Leer números saca en un formulario secundario con un TextBox multilínea que lo ocupa siempre todo la lista de números que haya en el archivo. Será modal y al cerrarlo (con la X) se pregunta si se desea grabar solo si ha sido modificado.

Reset hará lo mismo que el botón y salir saldrá del programa sin pedir confirmación.


Acerca de reutiliza el formulario de leer números pero pondrá información del Autor. No será editable.

Ejercicio 7

Toma el ejemplo de LabelTextBox y añade lo siguiente:

- Haz que el evento KeyUp del LabelTextbox sea lanzado cuando suceda un evento KeyUp de txt.
- Crea un evento en LabelTextbox denominado TxtChanged el cual sea lanzado cuando suceda el evento TextChanged del textbox interno.
- Haz una nueva propiedad de LabelTextbox denominada PswChr que enlace con la propiedad PasswordChar del Textbox interno.
- Cambia la funcionalidad de recolocación para que en lugar de que txt cambie de tamaño al cambiar la label, se mantenga siempre del mismo tamaño y cambie el tamaño del LabelTextBox aumentando Separacion.

En un formulario haz pruebas de todo lo anterior.

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Desarrollo de Interfaces				CURSO: 2º
	PROTOCOLO:	Ejercicios repaso	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

Ejercicio 8

Toma el ejemplo EtiquetaAviso visto en teoría y realízalo de forma que se le añadan las siguientes características **(se recomienda hacerlo desde cero sin mirar el código de los apuntes para que el repaso sea completo)**:

- Que pueda tener de forma opcional un fondo de gradiente entre dos colores. Establece los colores inicial y final del gradiente así como una booleana que indique si hay o no gradiente como nuevas propiedades.
- El enumerado eMarca tendrá una constante más denominada Imagen de forma que si se selecciona se colocará en la posición de la marca la imagen indicada en la propiedad imagenMarca.
- Crea un evento denominado ClickEnMarca que será lanzado cuando el usuario pulsa el ratón pero solo en la zona donde está la marca (salvo que sea Nada).

Ejercicio 9

Realiza el juego del ahorcado para que los dibujos del cliente los hagas con el GDI+ en lugar de con PictureBox. Para ello debes de crear un componente DibujoAhorcado que disponga de una propiedad entera denominada errores. Dependiendo del número que contenga dibujará más o menos partes del ahorcado. El componente debe disponer de un evento CambiaError que se lanza cada vez que cambia el número de errores y otro denominado Ahorcado que se lanza cuando se completa el dibujo.


Ejercicio 10 (Valida apartados por separado)

Como bien es sabido, en un alto porcentaje tanto a frikis como a geeks les cuesta establecer relaciones sociales por lo que se ha visto la posibilidad de crear una agencia matrimonial para fans extremos de diversos temas de forma que se le pueda poner en contacto una vez se estudien sus costumbres y se vea la compatibilidad entre ellas. Este es el objetivo de la agencia matrimonial Frikilove Inc. que ha visto, además, desbordada su capacidad de gestión por la gran cantidad de solicitantes (incluso han abierto su propio club de fans). Para controlar esta sobreproductividad han encargado a los aventajados alumnos del Colegio Vivas un software de gestión de frikis que, en una primera versión simple, debe cumplir las siguientes especificaciones:

a) Se establecerá un nuevo componente denominado ValidateTextBox que herede de UserControl y con estas características:

Dispondrá de un TextBox colocado en la posición 10,10 del nuevo componente. De esta forma le quedará un margen para la posibilidad de que quede rodeado por un recuadro de color. Siempre será de línea simple (No Multiline)

El alto del componente siempre será del tamaño del textbox interno +20 pixels y el textbox será del ancho del componente -20 pixels.

	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Desarrollo de Interfaces				CURSO: 2º
	PROTOCOLO:	Ejercicios repaso	AVAL:		DATA:	
	UNIDAD	COMPETENCIA				

Habrás propiedades en el nuevo componente para acceder al Text en el textbox interno.

Tendrá una propiedad pública (denominada tipo) que será un enumerado (denominado eTipo) con los valores:

Numérico (Sólo son válidos los dígitos)

Textual (No admitirá nada que no sea una letra o un espacio. Puedes usar la función Char.IsLetter)

El componente estará rodeado por un recuadro rojo (pintado mediante el GDI+) desde la posición x=5, y=5 hasta el ancho-5 y el alto-5 del componente. Se recomienda usar el DrawRectangle con la sobrecarga que simplemente necesita un Pen, la posición inicial del rectángulo y el ancho y el alto. Para el Pen usa la sobrecarga que sólo hay que darle color.

El rectángulo pasará a verde si el contenido es correcto según el enumerado. Para ello realiza una función denominada comprobar() y llámala cuando lo consideres necesario.

Debe haber acceso al evento TextChange del textBox desde el nuevo componente. Aunque no se use dicho evento no debe saltar excepción. Crea para ello un evento del componente denominado CambiaTexto.

b)En la aplicación existirán 2 formularios: El principal y el de introducción de datos que se describen más abajo.

Se tendrá una colección de estructuras que disponen de los siguientes campos:

Nombre (Cadena): Guardará tanto nombre como apellidos

Edad (integer)

Afición principal: Será un enumerado (denominado eAficion) con posibles valores: Manga, SciFi, RPG, Fantasía, Terror, Tecnología.

Sexo: Enumerado Hombre/Mujer (denominado eSexo)

Sexo opuesto: También el enumerado Hombre/Mujer. Indicará el sexo con el que desea buscar una relación matrimonial.

Foto: Cadena que indica la posición de la foto (se te entregarán varias imágenes que puedes utilizar).

La estructura se denominará sFiki y la colección frikis.

El formulario principal sirve para visualizar datos y de control general de la aplicación. Para ello se dispone de una lista (ListBox) donde aparecen los nombres de los clientes. Dicha lista permite selección múltiple.

Se dispone junto a la lista de los siguientes botones:

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Desarrollo de Interfaces				CURSO: 2º
	PROTOCOLO:	Ejercicios repaso	AVAL:		DATA:	
	UNIDAD COMPETENCIA					

Borrar: borra todos los elementos seleccionados previa confirmación con MessageBox. Los borra de la lista y de la colección.

Nuevo: Se describe más abajo como funcionará la inserción de nuevo clientes.

Para nuevo se usa un formulario de entrada de datos (será modal) en el cual se usarán 2 ValidateTextBox: uno para Nombre de tipo Textual y otro para Edad de tipo Numerico . Se usará un ComboBox para la afición principal y dos grupos de RadioButton para sexo y sexo opuesto. La foto se plantea con un TextBox con un botón al lado que abre un OpenFileDialog. De esta forma se puede escribir la trayectoria y el nombre de la foto o seleccionarlo mediante el botón. Dicho OpenFileDialog debe permitir escoger entre todos los archivos y archivos de imagen jpg en el filtro.

Tendrá además los botones Aceptar y Cancelar que devolverán las respuestas de diálogo de forma automática al ser pulsados. No habrá nada de código en este formulario salvo el necesario para rellenar el combobox y el botón para seleccionar la foto mediante el OpenFileDialog.

Al darle a aceptar se guardará en la colección.

Además, junto a la lista mostrarán todos los datos y la foto del primer cliente seleccionado mediante una etiqueta y un PictureBox de forma que la foto se amolde al tamaño del mismo pero sin distorsionarse. También aparecerá una segunda lista con las parejas válidas, es decir, que tengan la misma afición preferida y que su sexo coincida con el sexo opuesto del cliente (y viceversa).

Se vera también en la parte inferior las fotos de las parejas válidas. Para ello se crea en un panel tantos PictureBox de tamaño 80x80 como sean necesarios. En varias filas y 3 columnas

Existirá un menú con las siguientes opciones:

Clientes: Que a su vez tendrá los ítems: Nuevo, Borrar, separador y salir. Todos tienen atajo.

Acerca de: MessageBox con nombre del programa y del autor:

El menú tendrá acceso a todas sus opciones mediante ALT+Tecla seleccionada. Además si alguna opción ya existe asociada a otro botón compartirá código con el mismo (es decir, usará el mismo método asociado a los eventos click de ambos componentes)

Otras cosas que debe hacer el programa.

Si se pasa por encima de una foto el ToolTip será el campo Nombre de la estructura.

El título estará animado de forma que aparezcan las letras una a una desde el principio cada 200 ms. Cuando hayan aparecido todas vuelve a empezar.

Nota final: si te diera tiempo a realizar todos estos ejercicios, realiza el examen de evaluación como repaso final. Si aún así te sobra tiempo habla con el profesor para que te indique otras tareas de repaso.

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM			
	MÓDULO	Desarrollo de Interfaces				CURSO:	2º
	PROTOCOLO:	Ejercicios repaso	AVAL:		DATA:		
	UNIDAD		COMPETENCIA				