

Universidad Tecnológica Nacional Facultad Regional Avellaneda



Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos

Materia: Laboratorio de Programación II

Apellido:		Fecha:	01/08/2019
Nombre:		Docente ⁽²⁾ :	F. Dávila – D. Boullon – M. Cerizza
División:		Nota ⁽²⁾ :	
Legajo:		Firma ⁽²⁾ :	
Instancia ⁽¹⁾ :	<div style="display: flex; justify-content: space-around;"> PP RPP SP RSP FIN </div>		X


(1) Las instancias validas son: 1^{er} Parcial (**PP**), Recuperatorio 1^{er} Parcial (**RPP**), 2^{do} Parcial (**SP**), Recuperatorio 2^{do} Parcial (**RSP**), Final (**FIN**). Marque con una cruz.

(2) Campos a ser completados por el docente.

IMPORTANTE:

- Guardar el proyecto en el **disco D:**. Ante un corte de energía o problema con el archivo de corrección, el alumno será responsable de que el proyecto sea recuperable.
- Colocar sus datos personales en el nombre del proyecto principal, colocando:
"Apellido.Nombre.AñoCursada_Final_20190801"
Ej: Pérez.Juan.2018_Final_20190801.
No sé corregirán proyectos que no sea identificable su autor.
- **Reutilizar código** siempre que se pueda. Será evaluado.
- **Cuando se indique, realizar las modificaciones necesarias a las clases para cumplir con las consignas.**
- Todos los **diagramas son orientativos**. No necesariamente coinciden con el resultado esperado.

Al finalizar, colocar la carpeta de la Solución completa en un archivo ZIP que deberá tener como nombre

Apellido.Nombre.AñoCursada.zip y dejar este último en el Escritorio de la máquina. Luego presionar el botón  de la barra superior, **colocar un mensaje** y presionar *Aceptar*. **Aguardar a que el profesor indique que el examen fue copiado de forma correcta.** Luego retirarse del aula.

TIEMPO MÁXIMO PARA RESOLVER EL EXAMEN 120 MINUTOS.

A. Requisitos mínimos y obligatorios:

1. Usar la solución entregada. Modificar el nombre tanto a la carpeta y como a la solución como se indicó anteriormente.
2. Todos los atributos/campos deben ser privados e inicializados en el constructor excepto que se indique lo contrario.
3. Respetar las reglas de estilo de la cátedra, buenas prácticas y escribir un código prolijo.
4. Crear la base de datos utilizando el siguiente script:

```

CREATE DATABASE Final
CREATE TABLE Final.dbo.Usuarios (
    UsuarioID int PRIMARY KEY IDENTITY(1,1),
    Cuenta nvarchar(50),
    Clave nvarchar(50)
);

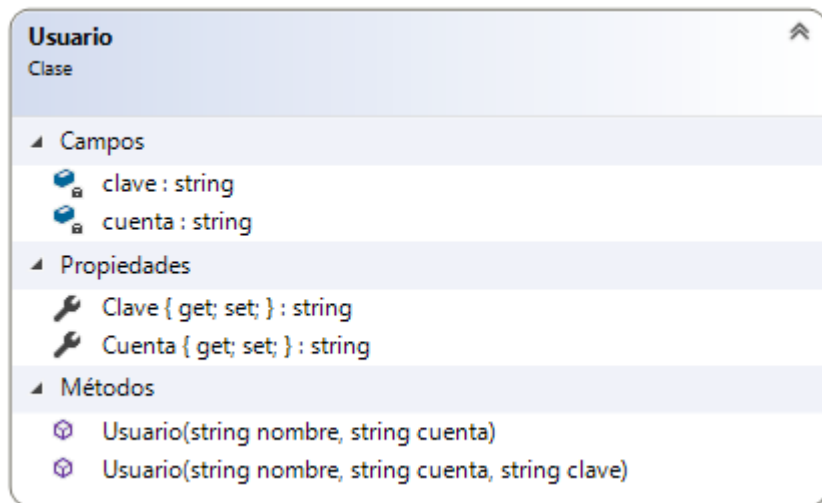
INSERT INTO Final.dbo.Usuarios (Cuenta, Clave) VALUES ('Admin', 'Admin');

```

B. Clase Persona:

1. La clase debe poder heredarse y heredar, pero NO debe poder generarse una instancia de la misma. El constructor deberá ser público.
2. Crear un método “MostrarDatos” que podrá ser opcionalmente invalidado/sobrescrito en las clases derivadas.
 - a. Retornará el nombre de la persona.

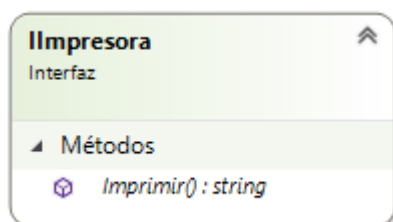
C. Clase Usuario:



1. Hereda de Persona. Inicializar campos heredados.
2. Crear los constructores.
 - a. La sobrecarga que no recibe clave inicializará la misma con el valor “EXTERNO”.
 - b. Reutilizar código.
3. Sus constructores sólo deben ser accesibles por clases derivadas.

D. Interfaz Impresora:

1. Crear la interfaz:



E. Clase Impresora:

1. El método "GenerarArchivo" guardará el texto recibido como argumento en un archivo de texto.
 - a. La ruta será la indicada en el campo "ruta".
 - b. Se anexará o sobrescribirá el archivo dependiendo del valor del campo "anexaTexto".
2. Crear una sobrecarga del método generar archivo.
 - a. Debe recibir cualquier objeto que implemente Impresora. NO usar genéricos.
 - b. Reutilizar código. El método "Imprimir" del objeto devuelve el texto a guardar.
3. Asegurarse de cerrar las conexiones ante una excepción (si están abiertas).

F. Clase BusinessException:

1. Modificar la clase:
 - a. Es una excepción propia de la aplicación.
 - b. El constructor recibirá el mensaje de la excepción e inicializará el campo en la clase base.
2. Implementará Impresora.
3. El método creado retornará un mensaje **usando String.Format()** con el siguiente formato: "{fecha y hora actual} – {mensaje de la excepción}".

G. Clase Serializador:

1. Debe ser una clase no-estática que NO se pueda heredar.
2. Clase genérica. Recibirá un solo tipo genérico.
3. Restringir el tipo genérico, deberá tener un constructor sin parámetros.
4. Crear el método "Serializar" para serializar a XML el tipo genérico. Usar el campo "ruta".
5. Crear el método "Deserializar" para deserializar de XML al tipo genérico. Usar el campo "ruta".
6. Modificar las clases Persona, Usuario, Empleado y Externo para que sean serializables a XML todos sus datos. Se deben realizar los cambios en todas, sino no funcionará (por la relación de herencia).
7. Asegurarse de cerrar las conexiones ante una excepción (si están abiertas).

H. Clase UsuarioDAO:

1. Clase estática.
2. Modificar el método "Registrar" para insertar un nuevo usuario a la base de datos.
3. Crear un método "Autenticar" que recibirá un nombre de cuenta y su clave, y verificará en la base de datos que la combinación exista. Para esto usar un bloque if-else y el método "Read" de SqlDataReader, si leyó algún registro es porque existe.
 - a. Si no existe lanzará un BusinessException con el mensaje "Nombre de la cuenta o clave incorrectos. Por favor, reintentar."
 - b. Si existe retornará true.
4. Asegurarse de cerrar las conexiones ante una excepción (si están abiertas).
5. Estructurar la clase de manera de reutilizar código.

I. Clase Empleado:

1. Hereda de Usuario y TIENE clave (usar sobrecarga correspondiente).
2. Tiene un enumerado "Area". Crear el enumerado, recibirlo e inicializarlo en el constructor.



3. Invalidar el método "MostrarDatos" de Persona.
 - a. Debe retornar un string que indique que es un empleado, junto con el nombre y el área. Reutilizar código. Ejemplo: "EMPLEADO – Juan perez – RRHH".

J. Clase Externo:

1. Hereda de Usuario y NO tiene clave (usar sobrecarga correspondiente).
2. Invalidar el método "MostrarDatos" de Persona.
 - a. Debe retornar un string que indique que es externo con el nombre. Reutilizar código. Ejemplo: "EXTERNO – Maria Suarez".

K. Clase Extension:

1. Crear un método de extensión "ListarUsuarios" de la clase String que reciba una lista de Usuarios (List<Usuario>).
2. "ListarUsuarios" recorrerá la lista concatenando el texto devuelto por el método "MostrarDatos".
 - a. **Utilizar StringBuilder.**
 - b. Hacer un salto de línea por cada elemento de la lista, para esto usar el método de StringBuilder que corresponda, NO "\n".

L. Clase Empresa:

1. Sobrescribir el método "ToString". Utilizar el método de extensión del punto K pasándole como argumento el campo "usuarios".
 - a. Ayuda: Usar un string vacío "String.Empty" como -instancia de String-.
2. Sobrecargar el operador "==". Devolverá true si ya existe un objeto Usuario en la lista con el mismo valor en el campo cuenta.
3. El método "ActualizarEnParalelo" inicializará un nuevo hilo que correrá el método "Actualizar".
4. Crear un tipo delegado "Accion" que no retorne nada y no reciba nada.
5. Crear un método "OnChange" de tipo "Accion".
 - a. Lanzarlo verificando previamente que tenga manejadores asociados en los métodos "Actualizar" y "Registrar" en el punto que está indicado como comentario.

M. PrincipalForm:

1. Al final del método "PrincipalForm_Load" asociar el evento "OnChange" de Empresa al manejador "ActualizarLista".

N. AuthForm:

1. Agregar un botón con el texto "Ingresar" llamado "btnIngresar".
2. Al hacer click en dicho botón usar el método "Autenticar" de la clase UsuarioDAO para verificar los datos ingresados.
 - a. En la base de datos ya existe la cuenta "Admin" con la clave "Admin".
3. Atrapar las excepciones de tipo "Business Exception":
 - a. Usar el método "GenerarArchivo" de la clase Impresora para anexar el mensaje del punto F3 en un archivo de logs "Log.txt". Se valorará usar la sobrecarga que NO recibe un string. El archivo debe guardarse en el escritorio.
4. Atrapar todas las excepciones restantes:
 - a. Mostrar el mensaje de la misma en un Message Box con título (caption) e icono de error, y un solo botón de aceptar(OK).
5. Si se pudo autenticar:
 - a. Abrir el "PrincipalForm" de forma no-modal.
 - b. Usar la propiedad "Visible" del "AuthForm" y ocultarlo.

O. Test Unitario:

1. Crear un nuevo proyecto de test unitario.
 - a. Que tanto el proyecto como la clase y el método tengan un nombre apropiado y descriptivo de lo que hacen.
2. Verificar en un método que al tratar de registrar un Usuario (método Registrar de Empresa) con la misma "cuenta" se lance una excepción de tipo BusinessException.
 - a. Variar el resto de los datos para asegurarse que sólo se lance por la cuenta.