Universidad Tecnológica Nacional Facultad Regional Avellaneda										
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos										
Materia: Laboratorio de Programación II										
Apellido:					Fecha:		13/12/2018			
Nombre:					Docente(2):					
División:					Nota(2):					
Legajo:					Firma(2):					
Instancia(1):	**PP**		**RPP**		**SP**		**RSP**		**FIN**	X

(1) Las instancias validas son: 1er Parcial (**PP**), Recuperatorio 1er Parcial (**RPP**), 2do Parcial (**SP**), Recuperatorio 2do 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.
- **2 (dos) errores en el mismo tema anulan su puntaje.**
- **Errores de conceptos de POO anulan el punto.**
- **Cada tema vale 1 (un) punto (Herencia, Generics, Test Unitarios, etc.). La correcta documentación también será evaluada.**
- **Se deberán tener al menos el 60% bien de los temas a evaluar según la instancia para lograr la aprobación.**
- Colocar sus datos personales en el nombre del proyecto principal, colocando: Apellido.Nombre.AñoCursada. Ej: Pérez.Juan.2018. No sé corregirán proyectos que no sea identificable su autor.
- **Salvo que se indique lo contrario, TODAS** las clases deberán ir en una Biblioteca de Clases llamada Entidades.
- No se corregirán exámenes que no compilen.
- **Reutilizar** tanto código como crean necesario.

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 **100** MINUTOS.*

---

1. Modificar el nombre tanto a la carpeta y como al proyecto entregados por el descripto anteriormente.
2. Generar la excepción `FechaNacimientoInvalidaException`. La misma tendrá 3 constructores:
   a. `FechaNacimientoInvalidaException()`: Tendrá un mensaje por defecto.
   b. `FechaNacimientoInvalidaException(string mensaje)`
   c. `FechaNacimientoInvalidaException(string mensaje, Exception innerException)`
3. Generar un método de extensión dentro del proyecto ***Extra*** para la clase de sistema `DateTime` que retorne si la instancia de este tipo es menor a la fecha actual (miInstanciaDeDateTime <= Now).
4. **Botón** *"Punto 1"*:
   a. Partiendo de las clases Humano y Pez generar una clase base llamada Animal e implementar los 4 pilares de la P.O.O..
      i. Animal será abstracta y tendrá 3 constructores:
         1. `Animal()` que asignará como nacimiento la fecha actual
         2. `Animal(string nombre)` que asignará como nacimiento la fecha actual

3. `Animal(string nombre, DateTime nacimiento)`

ii. La propiedad `Nacimiento` validará la fecha de nacimiento con el método de extensión antes creado. De ser inválida lanzará la excepción `FechaNacimientoInvalidaException`.

iii. Una conversión explicita para devolver todos sus datos en formato String.

iv. `ExponerDatos` será virtual.

b. Agregar a cada una de las 3 clases una sobrecarga de == para comparar que todos los atributos de dos objetos sean iguales.

c. Generar un atributo de la clase FrmFinal del tipo Pila de Animales llamado `animales`.

d. Agregar los 5 animales a la misma. Tener en cuenta que el animal a03 debería lanzar una excepción y controlarla mostrando el mensaje por pantalla.

e. Luego desapilar todos los elementos e ir acumulando sus datos para ser mostrados en un MessageBox.

5. **Botón** *"Punto 2"*:

a. Si el hilo no está corriendo

i. Generar una nueva instancia del hilo. El hilo será parametrizado.

ii. Iniciará dicha instancia para el método Animar (descripto en el punto 3.c).

b. Si el hilo está activo, se lo frenará. También se deberá frenar el hilo al salir del formulario.

c. El método Animar ejecutará el método `CorrerBarra` y hará un *Sleep* del tiempo recibido como parámetro. Repetirá dichas acciones hasta que sea frenado manualmente.

6. **Botón** *"Punto 3"*:

a. Agregar una interfaz genérica IArchivos al proyecto con los métodos:

i. `void Guardar(string path, T datos)`

ii. `T Leer(string path)`

b. Implementar la interfaz en Humano y Pez.

i. Humano se serializará en Binario.

ii. Pez se serializará en XML.

c. Guardar un Humano y un Pez en el escritorio con los nombres de archivo “Humano.bin” y “Pez.xml” respectivamente.

7. Agregar un proyecto de Prueba Unitaria. Dentro de la misma clase:

a. Generar un test unitario llamado TestBinario donde se serializará y deserializará un objeto, evaluando si el objeto guardado y el leído son iguales por intermedio de la sobrecarga del ==.

b. Generar un test unitario llamado TestXML donde se serializará y deserializará un objeto, evaluando si el objeto guardado y el leído son iguales por intermedio de la sobrecarga del ==.

8. **Botón** *"Punto 4"*:

a. Agrega a la clase Humano:

i. Un delegado con el formato `void MiDelegado(string mensaje)`.

ii. Un evento para dicho delegado llamado `miEvento`.

iii. Método `void LanzarEvento()` que invoque al evento creado en el punto anterior con el mensaje `"Mensaje para mostrar"`.

b. Enlazar el evento con un manejador en el formulario y mostrar el mensaje recibido con un MessageBox.

Segundo Parcial

---

9. Crear una clase AnimalDao que implemente IArchivos. Reutilizar código y seguir las buenas prácticas para lograr que:

a. **Guardar**: guardará en la base de datos el nombre, obtenido de los objetos del tipo Humano en una lista de animales recibida como parámetro. También recibirá como parámetro el nombre de la tabla (argumento *path*) en la cual se guardará esa información.

b. **Leer**: consultará los nombres guardados y retornará una lista de animales con los mismos.

10. Agregar un botón con el texto "Punto 5" y el nombre btnPunto5:

a. Se creará una lista con 3 animales, y se la guardará en la base de datos.

b. Luego se leerá de la base de datos todos los animales guardados y se mostrará por pantalla un MessageBox con todos sus nombres.