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

1) **(1 pto)** Realizar un método de extensión, de una clase sellada persona con un atributo único y que el método muestre el largo del atributo único

2) **(1 pto)** Crear un objeto de tipo Queue <Double>. Apilarle la siguiente secuencia de números: 5, 2, 4. Realizar un algoritmo que permita tener la secuencia ordenada de manera inversa en la misma colección, es decir: 5,4, 2. De ser necesario, utilizar sólo colecciones de tipo Stack<Double> ó Queue<Double>.

3) **(2 ptos)** Crear dos objetos de tipo Deposito, cada uno de estos objetos contiene un Array de la clase Producto (de hasta 3 elementos). La clase Producto tiene dos atributos: Nombre y Stock.
Se debe poder sumar los Array de los dos depósitos (con la sobrecarga de un operador en la clase Deposito) y guardar el valor que retorna en un Array de Productos, recordar que si un producto está en los dos Arrays, se debe sumar el stock y no agregar dos veces al mismo producto.

4) **(1 pto)** Sabiendo que estas líneas de código son correctas, que los atributos de las clases son **públicos** y además que todas las clases poseen un **solo constructor**, realice los **constructores**, de cada una de las clases, sabiendo que **ProdVendido** hereda de **ProdExport**, que **ProdExport** hereda de **ProdImpuesto** y que éste último hereda de **Producto**.

```
Producto pro = new Producto("Pala", 22);
ProdImpuesto pI = new ProdImpuesto(pro.Nombre, pro.Stock, 600.33);
ProdExport pEX = new ProdExport(pI, "Argentina");
ProdVendido p = new ProdVendido (pEX, "Cliente Juan");
```

5) **(2 ptos)** Crear la clase Galpon, que contenga una lista genérica de tipo T, con una propiedad ***"Cantidad"*** que sólo permita asignar un valor (entero) al atributo ***"_cantidad"*** y un evento (diseñarlo para que reciba un Object y un EventArgs, su retorno será void). Si el valor que se intenta asignar es cero, se deberá lanzar una excepción de tipo ArgumentException informando de lo acontecido. Si el valor es par (utilizar lo hecho en el punto 1), dejar asignarlo. Si el valor es impar (utilizar lo hecho en el punto 1) disparar el evento ***EsImpar*** cuyo manejador tendrá que escribir en un archivo de texto (log.txt) la fecha (hh:mm:ss y el valor) y asignarlo.

6) **(1 pto)** Realizar una estructura try-catch (en el Main) para la escritura de la propiedad del punto anterior que, al capturar la excepción, muestre el mensaje.

7) **(2 ptos)** Realizar el burbujeo de una excepción **propia**, comenzando en un método de instancia, pasando por un método de estático y capturado por última vez en el Main.

8) **(2 ptos)** Crear la siguiente interface: `public interface IGuardarXML { bool SerializarXML(); }`
Implementarla en la clase Galpon.
Agregar a una instancia de tipo Galpon<Deposito> (que contenga al menos un objeto de tipo Producto, otro de tipo ProdImpuesto, otro de tipo ProdExport y otro de tipo ProdVendido) y generar una serializacion XML del galpón.
Modificando lo que crea conveniente para poder serializar todos los atributos de todos los objetos intervinientes, guardando en el archivo ***archivo.xml***.

9) **(2ptos)** hacer un botón que dispare un hilo que luego de 10 segundos cambie el color del formulario, por un color que recibe el hilo por parámetro.

Nota: dos errores en el mismo punto lo anulan, errores de conceptos de POO anulan el punto.

TIEMPO MAXIMO PARA RESOLVER EL EXAMEN 60 MINUTOS.