Segundo Parcial Laboratorio II

IMPORTANTE:

- **2 (dos) errores en el mismo tema anulan su puntaje.**
- La correcta documentación y reglas de estilo de la cátedra serán evaluadas.
- Colocar sus datos personales en el nombre de la carpeta principal y la solución: Apellido.Nombre.Div. Ej: Pérez.Juan.2D. No sé corregirán proyectos que no sea identificable su autor.
- No se corregirán exámenes que no compilen.
- **Reutilizar** tanto código como crean necesario.

El código completo deberá ser subido a un repositorio **privado** de github con el nombre "SP_LABORATORIO_II_A**221**_apellido_nombre" (deberán reemplazar la palabra apellido por su verdadero apellido y nombre por su nombre de pila) y deberá agregarse como colaboradores a los usuarios:

- german2017
- patocostello

Deberán realizar un commit en los siguientes horarios:

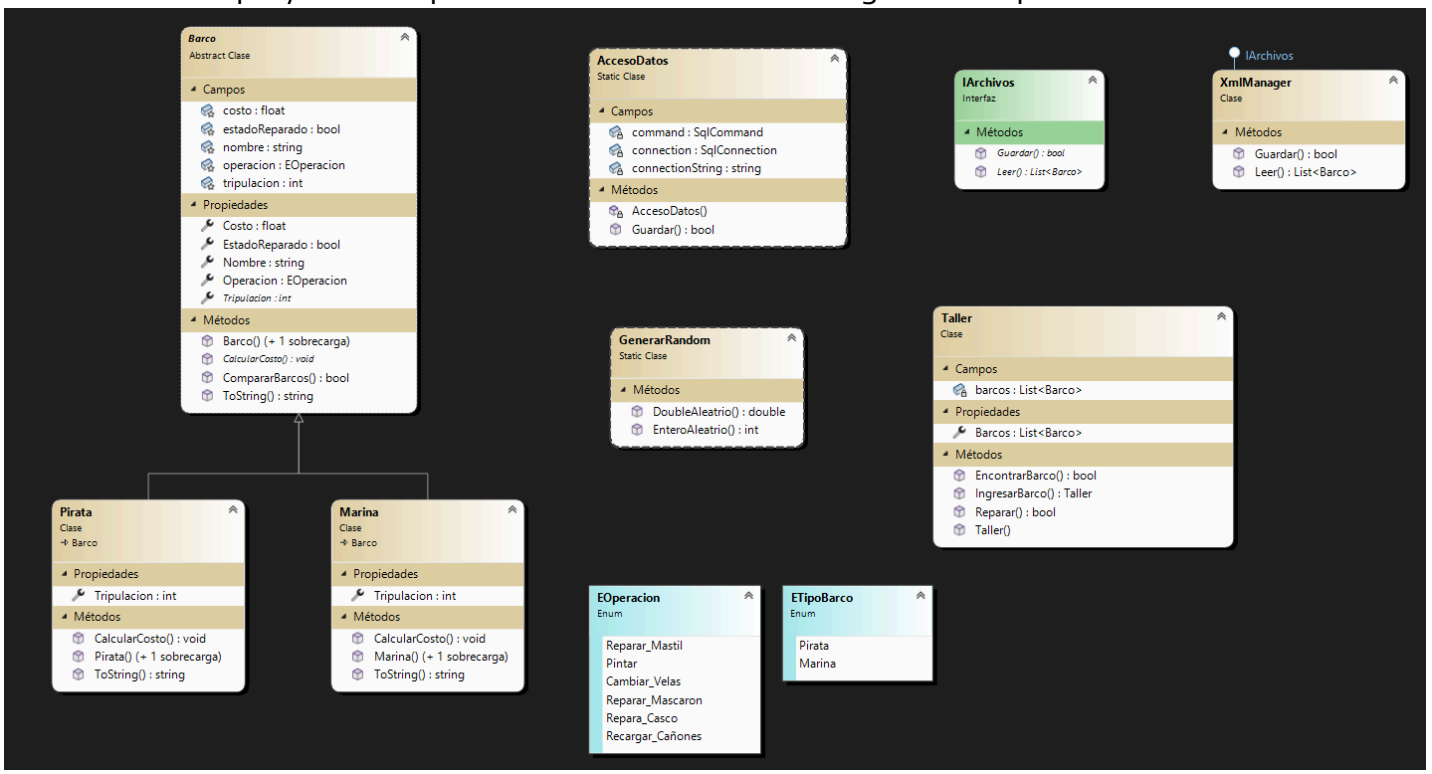
- 14:30: commit inicial
- 15:30
- 16:30
- 17:30 Commit final

Se debe evidenciar el avance del proyecto en cada commit

Primera parte:

Se desea desarrollar una aplicación para un taller de reparación de barcos. Para ello se debe:

1. Crear un proyecto de tipo biblioteca de clases con el siguiente esquema:



Clase Barco:

a. Clase Abstracta. Los miembros marcados *en cursiva deberán ser declarados como miembros abstractos.*
b. El constructor sin parámetros no hará nada.
c. CompararBarcos: dos barcos serán iguales si comparten el mismo nombre.
d. El método ToString, construirá un StringBuilder con los datos del barco.

Clase Pirata:

a. Sobrescribir la propiedad Tripulación. Si el barco no tiene tripulación, generar un número random entre 10 y 30.
b. Sobrescribir el método CalcularCosto. Generar un número random entre 2000 y 12000 y asignar al atributo.
c. El método ToString, construirá un StringBuilder con los datos del barco pirata.

Clase Marina:

a. Sobrescribir la propiedad Tripulación. Si el barco no tiene tripulación, generar un número random entre 30 y 60.
b. Sobrescribir el método CalcularCosto. Generar un número random entre 5000 y 25000 y asignar al atributo.
c. El método ToString, construirá un StringBuilder con los datos del barco de la Marina.

Clase Taller:

a. El constructor será el encargado de instanciar la lista.
b. El metodo EncontrarBarco retornará true si el barco se encuentra en el taller, false en caso contrario.
c. El metodo IngresarBarco validará que el barco no se encuentre en la lista y lo agregará.
d. El método Reparar:
    i. Validará que el objeto que recibe como parámetro sea de tipo taller.
    ii. Recorrerá la lista de barcos. Si el barco no fue reparado (tener en cuenta la propiedad EstadoReparado):
        1. Calcular el costo de la reparación.
        2. Guardar la reparación en la BD.
    iii. Cambiar el estado del barco a Reparado.

El método retorna true en caso de éxito, false en caso de error en alguno de los pasos

Interfaz: generar una Interfaz IArchivos.

Clase XmlManager:
Implementar IArchivos. Esta clase será la responsable de serializar y deserializar una lista de Barcos en XML.

Clase AccesoDatos:

a. Se utilizará para guardar en la BD por cada reparación: El mensaje: $"Se reparó el {nombre} a un costo de {costo} berries". Y el nombre del alumno.
b. La estructura de la tabla en la base de datos será la siguiente:

id INT PRIMARY KEY autoincremental,
mensaje VARCHAR (255) NOT NULL,

alumno VARCHAR (255) NOT NULL,

Formularios:
Cuentan con los comentarios necesarios sobre los métodos donde deberán realizar determinadas acciones. Estos comentarios inician de la siguiente forma **//TODO**.

i. **FrmPrincipal**:
   1. El evento LOAD del formulario, cargará en la lista de Barcos, todos los barcos serializados en XML si es que existe el archivo.
   2. El evento CLICK del botón Cargar Barco agregará el barco al taller si DialogResult retorna OK y el barco no está repetido. Informar de lo acontecido.
   3. El evento CLICK del botón Guardar, guardará el taller en un archivo XML en el mismo directorio que el proyecto.
   4. Al intentar cerrar el formulario preguntar si se desea salir.

ii. **FrmBarco:**
   1. El vento CLICK del botón Cargar instanciará el barco y retornará OK.

iii. **FrmReparacion:**
   1. Su constructor recibe un Taller como parámetro.
   2. El evento LOAD deberá mostrar en el listbox todos los barcos del taller

Segunda parte:

Grabar un video (máximo 15 minutos) **mostrando y explicando** el código del parcial. Podrán seguir el siguiente guión a efectos de organizar la defensa y el tiempo presupuestado del video:

a. Mostrar el programa en tiempo de ejecución (**3 min**): ejecutar cada una de las opciones del programa. La demostración deberá evidenciar la lectura y escritura en archivos y cada una de las opciones del programa.
b. Mostrar y explicar el desarrollo de los componentes (clases e interfaces) del proyecto (12 min): mostrar y explicar los métodos solicitados. En la explicación deberán evidenciar el conocimiento que tienen sobre los temas solicitados. Es decir, no queremos que lean código, sino que expliquen qué hicieron, cómo lo hicieron y cómo se relaciona la pieza de código con los temas vistos.

Una vez terminado el video, subirlo a drive o a youtube y agregar el enlace al readme del repo. Recuerden darnos permiso para verlo. De lo contrario el examen será desaprobado.

No se corregirán exámenes que no cuenten con el video defensa. Es decir, si entregan el código, pero no el video están desaprobados.

Entrega: subir al espacio del parcial en moodle un archivo comprimido con toda la solución. Agregar en un comentario el repositorio, esto a efectos que la entrega quede registrada en el campus. Por otro lado completar el formulario de asistencia marcando la finalización del parcial.

CUALQUIER INTENTO DE COPIA ANULARÁ EL PARCIAL DE TODOS LOS ESTUDIANTES INVOLUCRADOS EN EL PLAGIO