Universidad Tecnológica Nacional
Facultad Regional Avellaneda

Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos

Materia: Laboratorio de Programación II

Apellido:		Fecha:	19-11-2020
Nombre:		Docente(2):	F. Dávila
División:	2°C	Nota(2):	
Legajo:		Firma(2):	

Instancia(1):	PP		RPP		SP	X	RSP		FIN	

(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:

- **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. Ej: Pérez.Juan. No sé corregirán proyectos que no sea identificable su autor.
- No se corregirán exámenes que no compilen.
- **Reutilizar** tanto código como crean necesario.
- Colocar nombre de la clase (en estáticos), **this** o **base** en todos los casos que corresponda.
- Aplicar los principios de los 4 pilares de la POO.
- La entrega será en un archivo comprimido, el cual debe contar con Apellido y Nombre, al igual que la solución. Se entregará al finalizar, mediante Slack, por MD al docente a cargo.

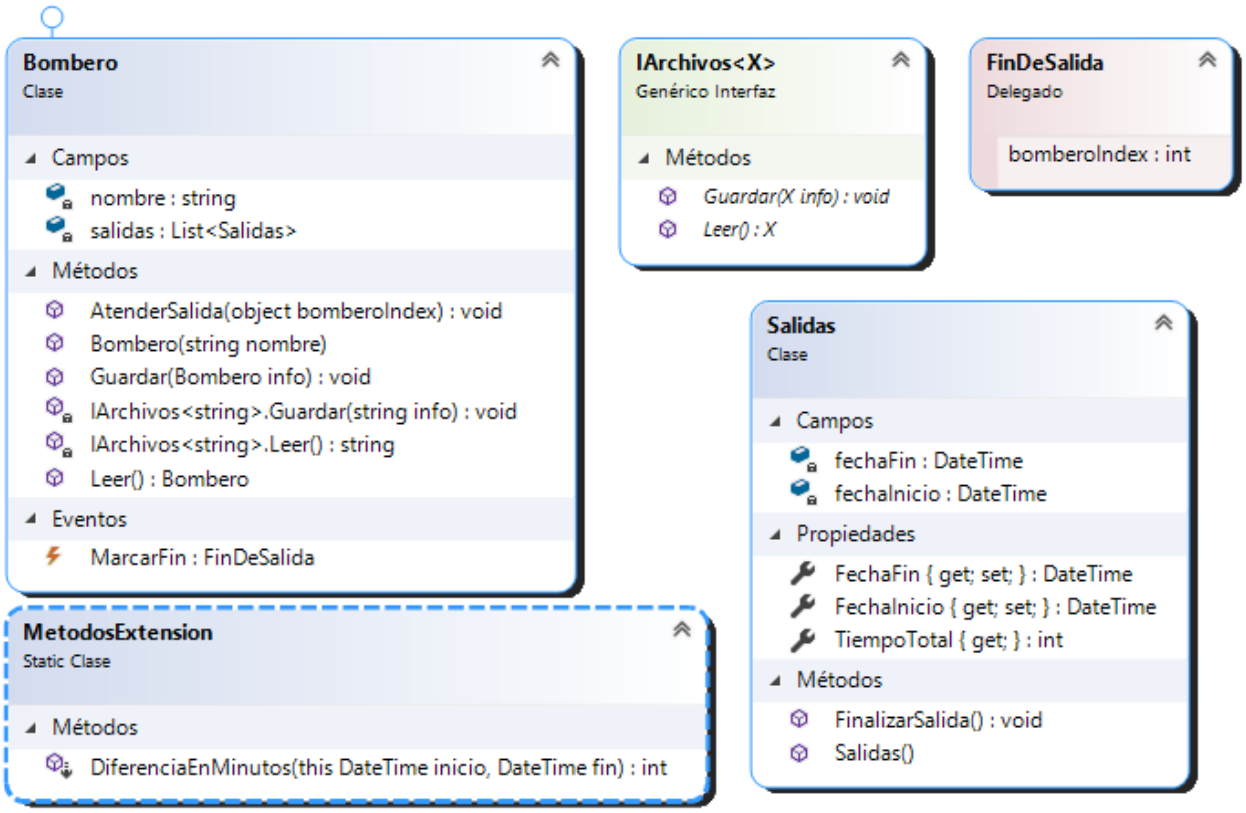
Se realizará una aplicación para despachar bomberos a distintos servicios.

- Partir del formulario dado.
- Crear la base de datos 20201119-sp y correr el siguiente script:

```
USE [20201119-sp]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[log](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[entrada] [varchar](100) NOT NULL,
	[alumno] [varchar](60) NOT NULL
) ON [PRIMARY]
GO
```

Todos los ítems que siguen serán evaluados, con puntaje, preste atención:

1. Cambiar el nombre de la carpeta de la Solución con sus datos personales: Apellido.Nombre.
2. Crear dentro un proyecto de Bibliotecas de Clase con el nombre Entidades.
3. *Diagrama de clases*:



4. Extender la clase DateTime para calcular la diferencia en minutos entre una fecha de inicio y una fecha de fin: (fin - inicio).Minutes;.
5. **Salidas**:
    a. El constructor dará inicio a la salida colocando DateTime.Now en el atributo correspondiente.
    b. FinalizarSalida dará fin a la salida colocando DateTime.Now en el atributo correspondiente.
    c. TiempoTotal retornará la diferencia en minutos entre la fecha de inicio y fin.
6. **Bombero**:
    a. Implementará la interfaz IArchivos tantas veces como muestra el diagrama. Esto requerirá implementar la interfaz de forma implícita y de forma explícita, como muestra el diagrama:
        i. Para Bombero: serializará y deserializará el objeto en binario.
        ii. Para string: guardará y leerá de la base de datos dada el mensaje recibido como parámetro.
    b. AtenderSalida:
        i. Agregará una nueva salida a la lista del Bombero.
        ii. Agregar un sleep de entre 2 y 4 segundos.
        iii. Finalizar la salida.
        iv. Guardar el log en base de datos.
        v. Avisar mediante el evento MarcarFin que se terminó la salida (utilizar bomberoIndex para informar al Form cuál Bombero fue).
7. **Cuartel**:
    a. FinalDeSalida: será el manejador del evento. Hacer todo lo posible para que la línea this.fuegos[bomberoIndex].Visible = false; funcione.
    b. El evento FormClosing deberá asegurarse que ningún hilo quede activo al salir.
    c. DespacharServicio deberá salir AtenderSalida en un nuevo Thread.
    d. Si se quiere despachar a un bombero que ya está atendiendo otra salida, se deberá lanzar la excepción BomberoOcupadoException y controlarla en el evento Click de cada botón, guardando en el log "Salida bombero X no concretada".
    e. Los botones Reporte serializarán el objeto Bombero correspondiente.
8. **Test Unitarios**:
    a. En al menos 1 tests, probar los puntos 6.a.i relacionado con serialización. Probar tanto Guardar como Leer.

b. Hacer otro test que pruebe el método de extensión.