

# Universidad Tecnológica Nacional Facultad Regional Avellaneda



Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos

**Materia: Laboratorio de Programación II**

Apellido:		Fecha:	21/11/2019
Nombre:		Docente <sup>(2)</sup> :	
División:	2°C	Nota <sup>(2)</sup> :	
Legajo:		Firma <sup>(2)</sup> :	
Instancia <sup>(1)</sup> :	<div style="display: flex; justify-content: space-around;"> <span>PP</span> <span>RPP</span> <span>SP</span> <span>X</span> <span>RSP</span> <span>FIN</span> </div>		

(1) Las instancias validas son: 1<sup>er</sup> Parcial (PP), Recuperatorio 1<sup>er</sup> Parcial (RPP), 2<sup>do</sup> Parcial (SP), Recuperatorio 2<sup>do</sup> Parcial (RSP), Final (FIN). Marque con una cruz.

(2) Campos a ser completados por el docente.

### IMPORTANTE:

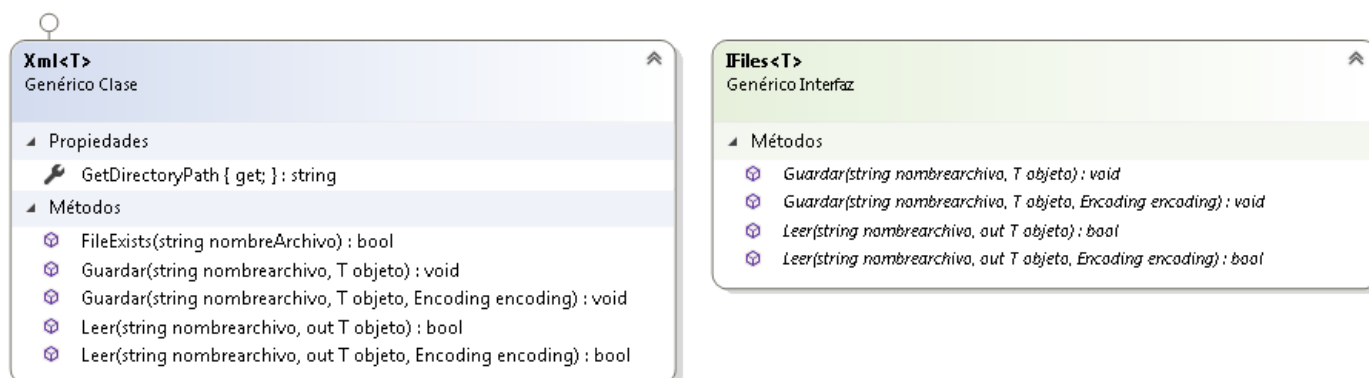
- **2 (dos) errores en el mismo tema anulan su puntaje.**
- **La correcta documentación y reglas de estilo de la cátedra serán evaluadas.**
- Colocar sus datos personales en el nombre del proyecto principal, colocando: Apellido.Nombre.Departamento. Ej: Pérez.Juan.2D. No se corregirán proyectos que no sea identificable su autor.
- **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.
- Colocar nombre de la clase (en estáticos), **this** o **base** en todos los casos que corresponda.

*TIEMPO MÁXIMO PARA RESOLVER EL EXAMEN 90 MINUTOS.*

1. Partir de la solución entregada. Modificar su nombre con el siguiente formato: [APELLIDO].[NOMBRE]

### Archivos

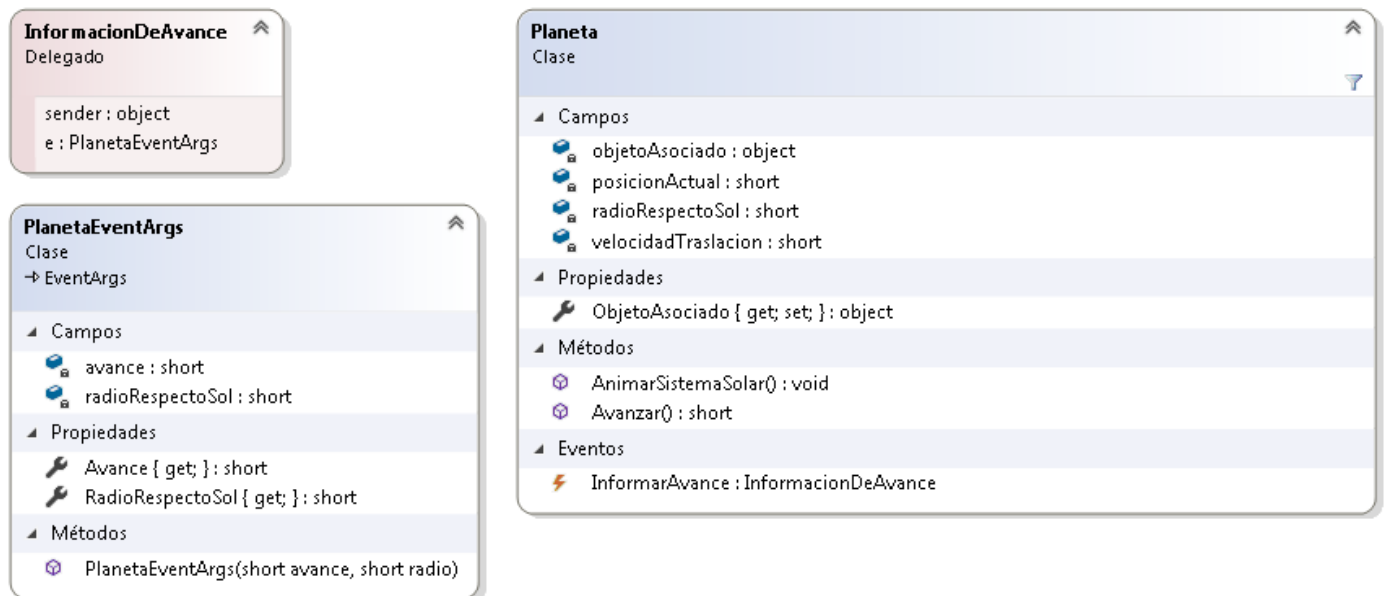
2. Dentro del proyecto **Archivos** (biblioteca de clases) se deberá respetar el siguiente esquema:



3. XML implementará la interfaz IFiles para escribir y consultar los datos serializados como XML.
4. Los atributos de los métodos serán:

- a. Nombrearchivo: sólo el nombre del archivo, sin path (al momento de hardcodear el nombre, será válido que el programador escriba "hola.xml" y no "C:\hola. xml").
  - b. Objeto a guardar/leer. Si falla la lectura, retornar el objeto por default.
  - c. Encoding del archivo. Por defecto UTF8.
    - i. ACLARACIÓN: Este se utilizará en el método Guardar para el constructor de XmlTextWriter y en el Leer no le daremos uso.
5. La propiedad GetDirectoryPath retornará la ruta al escritorio (debe ser independiente de la máquina en la que se abra el programa). Concatenar la barra final \ siendo la ruta retornada: C:\...\Desktop\.
- a. ACLARACIÓN: Al momento de guardar, el path completo será GetDirectoryPath + nombreArchivo.
6. FileExists comprobará si el archivo existe o no (sumará GetDirectoryPath al nombreArchivo recibido).
7. De producirse excepciones en los método Leer y Guardar deberá encapsularse esa excepción en una nueva excepción propia ErrorArchivosException y lanzarla. Crear esta excepción con al menos 2 constructores dentro del proyecto de archivos.
8. **Utilizar el finally para cerrar el archivo.** No utilizar using(...).

## Entidades



9. Crear la clase PlanetaEventArgs . Se cargará el atributo avance con el resultado del método Avance del Planeta y radioRespectoSol con el atributo del mismo nombre también del Planeta.
10. Crear el delegado dentro del archivo de la clase Planeta, como parte del namespace. Su retorno será void.
11. Agregar el evento InformarAvance en la clase Planeta.
12. Completar el método AnimarSistemaSolar con la invocación del evento.
13. Completar todo lo que haga falta para que se pueda serializar en XML guardando los datos velocidadTraslacion, posicionActual y radioRespectoSol presentes en la clase Planeta.

## Formulario


14. En el constructor deberán instanciarse los atributos planetas, animaciones y xml.
15. En el método Load del Form:
  - a. Asociar el manejador con el evento de cada planeta instanciado.
  - b. Preguntar si existe el archivo planetas (método FileExists de Xml), en cuyo caso se deberá leer el mismo y cargar la ubicación de los planetas.
16. En el evento click del botón Simular agregar un foreach, donde dentro se deberá:
  - a. Crear los hilos para el método AnimarSistemaSolar (un hilo por cada planeta).
  - b. Agregar dicho hilo a la lista de animaciones.
  - c. Iniciar cada hilo.
17. En el método DibujarAvancePlaneta lograr que se llame al hilo principal al mismo método, para que este pueda llamar a this.CalcularUbicacion.
18. En el método LimpiarAnimaciones abortar todos los hilos que aun estén vivos.

19. Crear en una clase aparte un método de extensión llamado EstadoSimulacion para la clase String que reciba un bool:
- Si el bool es true, retornará "Orbitando"
  - Si es false retornará "Detenido"
  - Lamarlo al iniciar o detener la simulación e imprimir el estado que corresponda en el Title del Formulario junto a su nombre y apellido.

### *Test Unitarios*

20. Agregar 2 test unitarios:
- Intentar guardar y leer un archivo correctamente, comprobando que los datos guardados sean iguales a los recuperados.
  - Intentar guardar un archivo en una ruta inválida (una ruta vacía o con caracteres inválidos servirá para dicho plan), comprobando que se lance la excepción `ErrorArchivosException`.

Al finalizar, colocar la carpeta de la Solución completa en un archivo ZIP que deberá tener como nombre Apellido.Nombre.division.zip y dejar este último en el Escritorio de la máquina.

Luego presionar el botón  de la barra superior, colocar un mensaje y apretar **Aceptar**. Finalmente retirarse del aula y aguardar por la corrección.