

**Universidad Tecnológica Nacional**  
**Facultad Regional Avellaneda**



|  |           |  |            |                          |            |  |            |   |            |  |
|--|-----------|--|------------|--------------------------|------------|--|------------|---|------------|--|
| Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos |           |  |            |                          |            |  |            |   |            |  |
| Materia: Laboratorio de Programación II                                      |           |  |            |                          |            |  |            |   |            |  |
| Apellido:  |           |  |            | Fecha:                   | 29-07-2021 |  |            |   |            |  |
| Nombre:  |           |  |            | Docente <sup>(2)</sup> : |            |  |            |   |            |  |
| División:  |           |  |            | Nota <sup>(2)</sup> :    |            |  |            |   |            |  |
| Legajo:  |           |  |            | Firma <sup>(2)</sup> :   |            |  |            |   |            |  |
| Instancia <sup>(1)</sup> :   | <b>PP</b> |  | <b>RPP</b> |                          | <b>SP</b>  |  | <b>RSP</b> | X | <b>FIN</b> |  |

(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.

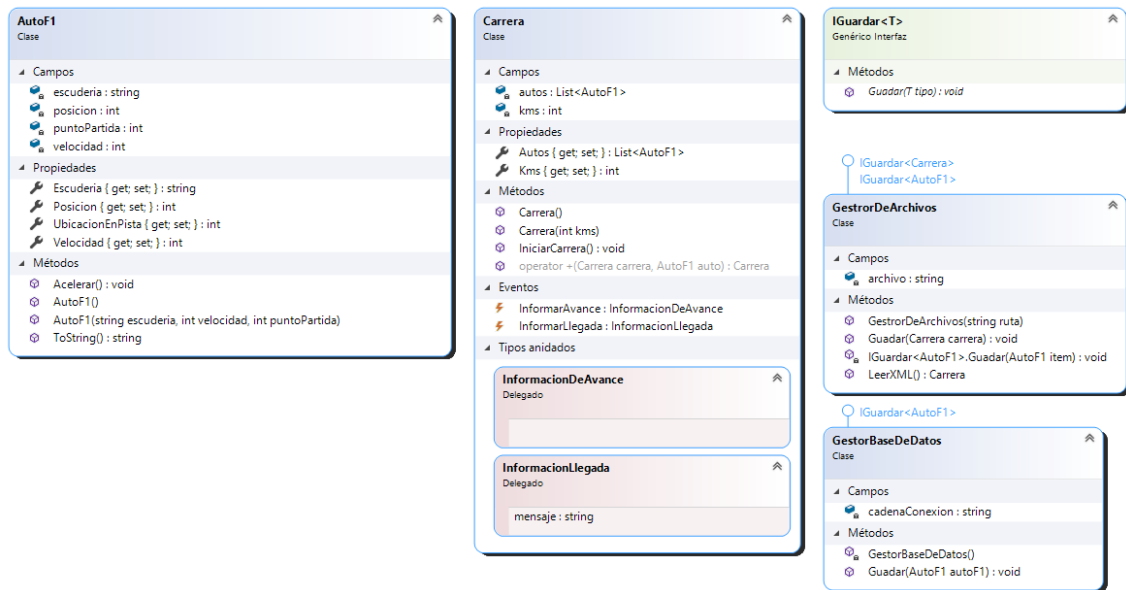
(2) Campos a ser completados por el docente.

**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 de la carpeta principal y la solución: Apellido.Nombre.Div. Ej: Pérez.Juan.2D. No se 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.

Se desea desarrollar una aplicación que simule una carrera entre 2 vehículos de F1.  
Para ello se debe:

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



## 2. Clase AutoF1:

- Sobre escribir el método ToString, el cual expondrá la escudería y la posición del vehículo:
  - `$"Escuderia: {this.Escuderia} - Posicion: {this.Posicion}"`
- La propiedad UbicacionEnPista retornara el punto de partida.
- Acelerar: sumara el atributo velocidad a puntoPartida.

## 3. Clase Carrera:

- Sera la clase encargada de simular la carrera.
- El método Iniciar carrera será ejecutado en un hilo secundario y deberá:
  - Se deberá iterar hasta que todos los autos se les haya asignado posición.
  - Recorrer la lista de vehículos de la carrera, acelerar cada vehículo.
  - Informar avance del vehículo.
  - Realizar un Sleep de 10 milisegundos.
  - Si la ubicación en pista del vehículo es mayor a Kms de carrera y la posición del Auto aun no fue asignada:
    - Se asignará la posición de llegada del vehículo, al ganador se le asignará 1 y al siguiente 2, etc.
    - Se informará la llegada del vehículo a la meta, reutilizar el ToString de AutoF1.
  - Aplicar el método del punto 5.b . Almacenar la llegada de los vehículos a la meta.
  - Aplicar el método del punto 6.a.i. En un archivo de texto almacenar la llegada de los vehículos a la meta.

## 4. Interfaz:

- Generar una Interfaz IGuardar, genérica que solo permita recibir tipos por referencia y que posean un constructor sin parámetros.

## 5. Base de Datos:

- a. El constructor será de clase y es donde se asignara el Connection String.
- b. Script para crear BD:

```
USE MASTER
GO
CREATE DATABASE [20210717-RSP]
GO
USE [20210717-RSP]
GO
CREATE TABLE [dbo].[resultados](
    [idposiciones] [int] IDENTITY(1,1) NOT NULL,
    [escuderia] [varchar](50) NOT NULL,
    [posicion] [int] NOT NULL,
    [horaLlegada] [varchar](50) NOT NULL,
    CONSTRAINT [PK_resultados] PRIMARY KEY CLUSTERED
(
    [idposiciones] ASC
))
```

- c. Generar una clase que implemente la Interfaz, la cual se utilizara para almacenar la llegada de los vehículos a la meta.
6. Archivos:
- a. Generar una clase que implemente la interfaz de forma implícita y explícita (ver diagrama), la cual se utilizara para:
    - i. Texto: Almacenar la llegada de los vehículos a la meta.
    - ii. Serializar en formato XML los datos de la Carrera al cerrar la aplicación.
  - b. De no poder leer el archivo, lanzar excepción propia `ArchivoException`.
7. Formularios:
- a. `FrmContador`: No hacer nada.
  - b. `FrmCarrera`: Cuanta con los comentarios necesarios sobre los métodos donde deberán realizar determinadas acciones. Estos comentarios inician de la siguiente forma `//Alumno.`
8. Test Unitarios:
- a. En al menos 1 test probar el punto 6.A.ii.
  - b. Probar el lanzamiento de `ArchivoException` en un Test Unitario al intentar leer un archivo que no es XML.