

**Universidad Tecnológica Nacional  
Facultad Regional Avellaneda**



Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos

**Materia: Laboratorio de Programación II**

Apellido:		Fecha:	26/06/2018
Nombre:		Docente <sup>(2)</sup> :	F. Dávila
División:	2ºD	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: 1º Parcial (PP), Recuperatorio 1º Parcial (RPP), 2º Parcial (SP), Recuperatorio 2º 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.
- **De explicitarse nada, 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. Modificar el nombre de la carpeta y la solución con el siguiente formato: [APELLIDO].[NOMBRE]
2. Las excepciones propias irán en un nuevo proyecto llamado Excepciones.
3. Dentro del proyecto Entidades crear la interfaz genérica IEntradaSalida con los métodos:
  - 3.1. Leer() : Z
  - 3.2. Guardar() : Z
4. Agregar un método de extensión FormatoTabla para la clase String que se le diga cuantos espacios ocupa un string y lo retorne con este formato, a fin de alinear los resultados al mostrar la salida por pantalla. Recordar que para esto se deberá utilizar string.Format y el formato deseado es {0, X} siendo X la cantidad de caracteres que utilizará esa columna.
5. Crear la clase GrupoDAO, la cual proveerá el acceso a los datos de la base para la clase Grupo. La clase contará con el método:
  - 5.1. Grupo ObtieneGrupo(Grupo grupo)
6. Clase Grupo:
  - 6.1. Implementar la interfaz genérica IEntradaSalida para el tipo Grupo.
    - 6.1.1.El método Leer:
      - 6.1.1.1. Tomará de la base de datos todos los equipos del Grupo que esta instancia represente.
      - 6.1.1.2. Cargará la lista de equipos.
    - 6.1.2.El método Guardar lanzará la excepción NotImplementedException con el mensaje "El Grupo no podrá ser serializado".

- 6.1.3. Validar en el **operator** + que si el grupo ya cuenta con el máximo de equipos (atributo maxCantidad), se lance la excepción propia **GrupoLlenoException** con el mensaje "El Grupo {0} ya cuenta con {1} equipos", siendo {0} la letra del grupo y {1} la cantidad máxima.
- 6.1.4. El método **MostrarTabla** deberá ordenar los datos antes de mostrarlos utilizando el método **Ordenar**.
- 6.1.5. Los métodos **Ordenar**, **Simular** y los constructores funcionan correctamente. Basándose en las necesidades antes mencionadas y las venideras, completar esta clase como corresponda.
7. Clase **Torneo**:
  - 7.1. La **constante** pública **MAX\_EQUIPOS\_GRUPO** tendrá valor 4.
  - 7.2. Implementar la interfaz genérica **IEntradaSalida** para el tipo **bool**:
    - 7.2.1. El método **Guardar** serializará como XML todos los grupos guardados en la lista, por separado, siendo el nombre de cada archivo "grupo-X.xml"; reemplazando X por la letra correspondiente al grupo (Desde A hasta D). Hacer las modificaciones necesarias para que guarde todos los datos.
    - 7.2.2. El método **Leer** buscará los archivos para los grupos aun no cargados en la lista y los cargará. O sea, si en la lista sólo tengo un objeto con el Grupo D, deberé buscar "grupo-A.xml", "grupo-B.xml" y "grupo-C.xml" y de existir dichos archivos cargarlos en la lista.
    - 7.2.3. Crear un método **SimularGrupos** que llame al método **Simular** de todos los objetos Grupo presentes en su lista.
8. Main:
  - 8.1. Crear un torneo llamado Rusia 2018
  - 8.2. Colocar como título de la consola "Copa Mundial Rusia 2018"
  - 8.3. Crear los objetos Grupo cuyo atributo Letras sea desde la A hasta la D.
  - 8.4. Leer los datos del grupo desde la base de datos.
  - 8.5. Dentro de la clase Program, crear un método **ImprimirResultados** que imprima los datos del Grupo que recibe como parámetro. Este reutilizará el método de la clase Grupo llamado **MostrarTabla**.
  - 8.6. **ImprimirResultados** será el manejador del evento **eventoResultados** de la clase Torneo.
  - 8.7. Lanzar en un hilo el método **SimularGrupos** de la clase Torneo.
    - 8.7.1. Dentro de **SimularGrupos** se llamará al método **Simular** de cada grupo en la lista y se lanzará el evento **eventoResultados**.
  - 8.8. Colocar un cartel que diga "Presione una tecla para continuar...", cuando el usuario lo haga limpiar la pantalla y continuar.
  - 8.9. Generar otro objeto Torneo, y en este utilizar el método **Leer**.
  - 8.10. Volver a simular los grupos mediante un Thread.
  - 8.11. Utilizar el método **Guardar** del objeto del tipo Torneo. Este se realizará como última acción antes de cerrar la aplicación.
9. Realizar los Test Unitarios necesarios para validar los siguientes casos:
  - 9.1. Que el método **Guardar** de la clase Grupo lance la excepción **NotImplementedException**.
  - 9.2. Que valide que los archivos de la clase Torneo se guarden y existan.
  - 9.3. Que el operador + de una instancia de la clase Grupo lance la excepción **GrupoLlenoException**.

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.