

**Universidad Tecnológica Nacional  
Facultad Regional Avellaneda**



Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos

**Materia: Laboratorio de Programación II**

Apellido:		Fecha:	26/07/2018
Nombre:		Docente <sup>(2)</sup> :	F. Dávila / H. Dillon
División:		Nota <sup>(2)</sup> :	
Legajo:		Firma <sup>(2)</sup> :	
Instancia <sup>(1)</sup> :	<input type="checkbox"/> PP <input type="checkbox"/> RPP <input type="checkbox"/> SP <input type="checkbox"/> RSP <input type="checkbox"/> FIN		


(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.**
- **Cada tema vale 1 (un) punto (Herencia, Generics, Test Unitarios, etc.). La correcta documentación también será evaluada.**
- **Se deberán tener al menos el 60% de los temas de cada parcial bien para lograr la aprobación.**
- **Errores de conceptos de POO anulan el punto.**
- Colocar sus datos personales en el nombre del proyecto principal, colocando: Apellido.Nombre.AñoCursada. Ej: Pérez.Juan.2018. No se corregirán proyectos que no sea identificable su autor.
- **Salvo que se indique lo contrario, 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.

Al finalizar, colocar la carpeta de la Solución completa en un archivo ZIP que deberá tener como nombre

Apellido.Nombre.AñoCursada.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**. Luego retirarse del aula y aguardar por la corrección.

*TIEMPO MÁXIMO PARA RESOLVER EL EXAMEN 100 MINUTOS.*

1. General:
  - a. Utilizar la teoría de encapsulamiento en todas las clases.
2. Clase Asiento
  - a. Redefinir ToString para que retorne la información del Asiento de forma ordenada. Utilizar string.Format o StringBuilder.
  - b. La clase debe ser abstracta.
  - c. Crear un método abstracto llamado ProbarAsiento que retorne un bool.
3. Clase Sofa
  - a. Agregar un enumerado con los colores: Natural, Blanco, Negro, Rojo. Utilizar este enumerado en el atributo color de Sofa.
  - b. Generar una relación de herencia entre Asiento y Sofa.
  - c. Modificar los atributos y constructores según crea necesario.

- d. El método ProbarAsiento hará un Sleep de 5 segundos y retornará true o false de forma aleatoria (Random).
4. Formulario:
- a. Los datos que falten (por ejemplo el color del sofá) se deberán agregar de forma aleatoria con el uso del Random.

---

### *Primer Parcial*

---

5. Formulario:
- a. Agregar un atributo del tipo Muebleria e instanciarlo en el constructor.
  - b. En el evento click de btnAgregar: Agregar el elemento a la mueblería.
  - c. En el evento FormClosing del Formulario: Preguntar al usuario si está seguro que desea salir.
6. Clase Banqueta:
- a. Agregar un enumerado con los tipos de banqueta: Caño, Madera, Hierro. Utilizar este enumerado en el atributo tipo de Banqueta.
  - b. Generar una relación de herencia entre Asiento y Banqueta.
  - c. Modificar los atributos y constructores según crea necesario.
  - d. El método ProbarAsiento hará un Sleep de 3 segundos y retornará true.
  - e. Primer parcial: sobrecargar el == para que dos banquetas sean iguales sólo si tienen el mismo tipo.
7. Clase Muebleria:
- a. Agrega un diccionario del tipo Asiento y cantidad <Asiento, int>
  - b. Sobrecargar el operador + para agregar asientos al diccionario de la Muebleria
    - i. Si el asiento es nuevo, agregarlo y colocar la cantidad en 1.
    - ii. Si el asiento ya existe, aumentar la cantidad en 1.
  - c. Crear un constructor por defecto que instancie el diccionario.
  - d. Crear un indexador que retorne un elemento del diccionario. Si el índice indicado no existe, retornar null.
  - e. Crear un método que reciba un tipo de banqueta y si existe en el diccionario retorne un string informando el tipo y su cantidad.

---

### *Segundo Parcial y Final*

---

8. Formulario:
- a. Agregar un atributo del tipo Lista de Asiento e instanciarlo en el constructor.
  - b. Controlar excepciones en archivos.
  - c. Para el manejo de archivos agregar una interfaz genérica con los métodos V Guardar(string path, T elemento) y T Leer(string path)
  - d. Generar dos clases: ArchivoTexto y ArchivoXML que implementen dicha interfaz. Completar los métodos según corresponda.
  - e. Con el botón btnProbar correr la prueba de todos los asientos:
    - i. Dentro de la clase del formulario, agregar el método BancoDePrueba, el cual será lanzado en un Thread y deberá recorrer todos los asientos y probarlos. Si el programa se cierra antes de que finalicen las pruebas, el Thread deberá ser terminado de forma correcta para que el proceso se cierre por completo.
    - ii. Modificar los métodos ProbarAsiento de Asiento y Sofa para que retornen void.
    - iii. Agregar en Asiento el método InformarFinDePrueba que reciba por parámetro un bool. Este método invocará al evento que se nombra a continuación.
    - iv. Crear un evento FinPruebaCalidad en Asiento para que informe mediante un texto que asiento es y si la prueba pasó o no. Mostrar el resultado por pantalla. Hacer los pasos necesarios para que esto funcione correctamente.
    - v. En resumen:
      - 1. BancoDePrueba llamará al método ProbarAsiento de cada elemento en la lista.

2. ProbarAsiento hará la demora y llamará a InformarFinDePrueba de Asiento.
  3. InformarFinDePrueba invocará al evento FinPruebaCalidad.
  4. El manejador del evento mostrará un MessageBox por pantalla informando el resultado de la prueba.
- f. En el evento click de btnAgregar:
- i. Al presionar el botón agregar se deberá guardar la información a un archivo, anexando el nuevo Asiento al final.
  - ii. Agregar el elemento a la lista.
  - iii. Luego, leer el archivo y mostrarlo en el RichTextBox.
- g. En el evento Load del Formulario se deberá leer el archivo y mostrarlo en el RichTextBox.
- h. Antes de cerrar, en el evento FormClosing, serializar la lista en XML. Hacer las modificaciones necesarias para guardar todos los datos.
9. Test Unitario:
- a. A través de las clases y métodos creados anteriormente, serializar un objeto y luego leerlo. Comprobar que todos sus datos sean iguales.