Universidad Tecnológica Nacional Facultad Regional Avellaneda											
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos											
Materia: Laboratorio de Programación II											
Apellido:				Fecha:	05/08/2021						
Nombre:				Docente(2):							
División:	2			Nota(2):							
Legajo:				Firma(2):							
Instancia(1):	**PP**		**RPP**		**SP**		**RSP**	X	**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.

1 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 sé corregirán proyectos que no sea identificable su autor.
- **TODAS** las clases deberán ir en una Biblioteca de Clases llamada Entidades, a no ser que se indique explícitamente otra cosa.
- 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 como se indicó anteriormente en este documento.
2) No agregar ninguna clase que no sea explícitamente requerida.

Esquema de Clases – Proyecto Entidades

3) Clase **Restaurant**
   a. Es la clase encargada de simular el despacho de los pedidos.
   b. Para llevar a cabo esta tarea, se utilizará el método Cocinar, dicho método debe:
      I. Validar que el parámetro Object sea de tipo Comida.
      II. Castear el parámetro Object a Comida.
      III. Dormir el hilo durante 2 segundos.
      IV. Validar que el evento PedidoEntregado tenga suscriptores.
      V. Invocar dicho evento, sumarle al atributo caja del Restaurant el precio de la comida (validar antes que el precio sea mayor a cero).
   c. El constructor de Restaurant debe establecer el atributo caja en cero.
   d. El DelegadoMozo será compatible con métodos que no retornen ni reciban parámetros.

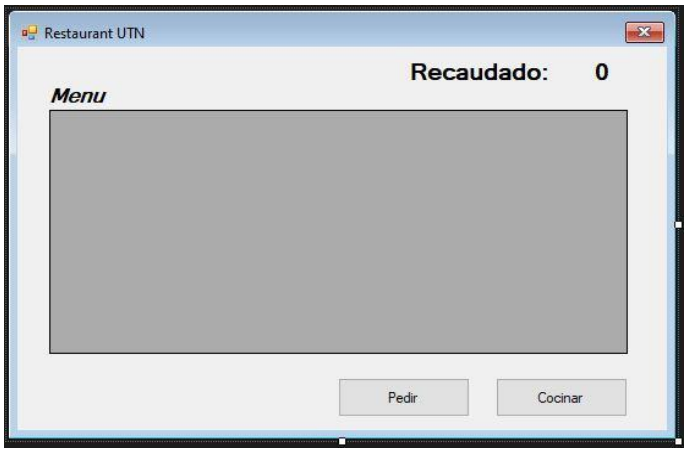
4) Clase **Comida**
   a. La clase comida tendrá todos sus atributos privados, propiedades get/set para cada uno de ellos y dos constructores (uno sin parámetros y otro que deberá establecer todos los atributos).

5) Clase **ComidaDAO**
   a. Esta clase genérica se encargará de traer desde la base de datos todas las comidas, para poder generar el menú de nuestro Restaurant (método traerDatosBD, el mismo retorna una lista).
   b. Dicha clase también implementa la interfaz genérica IComprobantes, ésta provee las firmas de dos métodos:
      i. GenerarTicket: Deberá generar archivos de texto, dichos archivos se deben guardar en la ruta recibida por parámetro, a dicha ruta se le debe concatenar el nombre de la comida al final para poder generar varios tickets distintos.
      ii. Escribirá en cada archivo una línea con el formato:
      `$"Detalle: {datos.Nombre}\nTotal{datos.Precio}"`
      iii. SerializarXML: Deberá serializar y guardar el parámetro T recibido, utilizar a la hora de guardar el archivo la ruta indicada.

6) Clase **ArchivoException**
   a. Esta excepción deberá lanzarse en caso de error a la hora escribir archivos (ya sea serialización xml o texto), El constructor de esta excepción deberá poder recibir en su constructor cualquier excepción que se produzca con anterioridad, y debe mostrar el mensaje de esta misma, conjunto a información propia, al usuario.

7) Clase **SinPedidosException**
   a. Esta excepción deberá lanzarse en caso de que la lista de pedidos del FrmRestaurant esté vacía al momento de ser utilizada, (Punto 12.a).

Esquema de Formularios – Proyecto View



La vista del formulario debe ser la presentada en este archivo, y su lógica debe ser la siguiente:

8) Lógica del constructor:
   a. El constructor del FrmRestaurant ya viene con código escrito, además de esto, debe ser capaz de instanciar todos los atributos excepto el de tipo Thread.
   b. Instanciar el atributo restaurant, para esto es necesario obtener la lista de comidas provenientes de la base de datos.
   c. El formulario cuenta con un atributo de tipo BindingSource, el mismo debe ser instanciado.
   d. Instanciar la lista de nombre “pedidos”, esta lista es auxiliar y sirve para acumular los pedidos provenientes del usuario.

9) Lógica del manejador del evento Load:
   a. Asignar a la propiedad DataSource del atributo bindingSource la lista de nombre “menu”, proveniente del atributo restaurant.
   b. Asignar a la propiedad DataSource del atributo DgMenu (de tipo DataGridView), el bindingSource.
   c. Suscribir al evento PedidoEntregado el método descrito en el punto 10.

10) Al evento PedidoEntregado del Restaurant se le debe suscribir un método propio, dicho método será creado dentro de la lógica del FrmRestaurant y debe ser capaz de actualizar el Label de nombre lblRecaudado (utilizar el atributo caja del restaurant para este fin) y de escribir por pantalla “Pedido entregado”..

11) El método manejador del botón Pedir será quien obtenga la Comida seleccionada del DataGridView (el código necesario para esto ya está escrito), dicho objeto se debe agregar a la lista “pedidos”. Hecho esto, se debe obtener la ruta al escritorio para usarla con los métodos GenerarTicket y SerializarXml del atributo comidaDao del FrmRestaurant.

12) El método manejador del botón Cocinar, deberá instanciar hiloCocinero y darle inicio (validando antes que el hilo no esté ejecutándose), pasarle por parámetro a este hilo un método que:
   a. Valide que la lista de pedidos tenga al menos 1 elemento, caso contrario lanzar excepcion SinPedidosException e imprimir por pantalla el mensaje de error.
   b. Recorrer la lista de pedidos, pasarle al método Cocinar del Restaurant cada uno de sus elementos.
   c. Al terminar de recorrer la lista, vaciarla. De esta manera la lista estará preparada para la siguiente toma de pedidos.

13) Test Unitarios
   a. Realizar un test unitario que compruebe la correcta obtención de los datos provenientes de la base de datos.
   b. Realizar un test unitario que compruebe que SerializarXML o GenerarTicket crea archivos.

Al finalizar, colocar la carpeta Solución completa en un archivo ZIP que deberá tener como nombre Apellido.Nombre.division.zip y compartir este por Slack sólo con el docente titular de la cursada.