|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Universidad Tecnológica Nacional**  **Facultad Regional Avellaneda** | | | | | | | | | | | | |
| Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos | | | | | | | | | | | | |
| Materia: Laboratorio de Programación II | | | | | | | | | | | | |
| Apellido: |  | | | | Fecha: | | | | 05/12/2023 | | | |
| Nombre: |  | | | | Docente(2): | | | |  | | | |
| División: | 2ºC | | | | 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.

**IMPORTANTE:**

* **2 (dos) errores en el mismo tema anulan su puntaje.**
* La correcta documentación y reglas de estilo de la cátedra serán evaluadas.
* El proyecto debe ser creado en .Net 5.
* Colocar sus datos personales en el nombre de la carpeta principal y la solución: Apellido.Nombre.Div. Ej: Pérez.Juan.2D. No sé 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.
* Aplicar los principios de los 4 pilares de la POO.

*TIEMPO MÁXIMO PARA RESOLVER EL EXAMEN* ***120*** *MINUTOS.*

1. Partir de la solución entregada. Modificar su nombre con el siguiente formato: [APELLIDO].[NOMBRE].
2. Implementar la BD desde el backup enviado.

**Files**

1. Dentro del proyecto se deberá respetar el siguiente esquema:

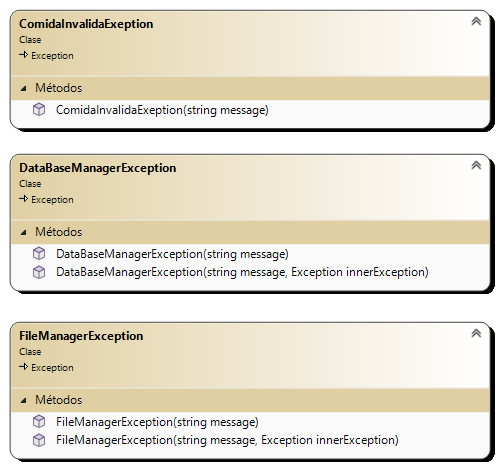
Texto

Descripción generada automáticamente con confianza media

1. FileManager será estatica.
   1. En el constructor:
      1. En el atributo path (privado) se almacenará la referencia al escritorio de la pc. Y se le concatenara un el nombre de la carpeta del parcial: ej {path escritorio}+\\20231207\_Alumn\\
      2. Llamar al método ValidaExistenciaDeDirectorio.
   2. ValidaExistenciaDeDirectorio:
      1. Sera privado.
      2. Si no existe el directorio almacenado en path, se creará.
      3. En caso de producirse una excepción al momento de la creación, esta deberá ser capturada y relanzada en una nueva excepción denominada FileManagerException, la cual contendrá el mensaje: “Error al crear el directorio”.
   3. Guardar:
      1. Sera el método para poder generar archivos de texto. El mismo se podrá usar para agregar información a un archivo ya existente o sobre escribirlo.
   4. Serializar:
      1. Sera genérico y solo aceptara tipos por referencia.
      2. Sera el método encargado de serializar en json.
      3. Retornara true al terminar la serialización;
   5. Reutilizar codigo donde crea necesario.

**Excepciones**

1. Dentro del proyecto respetar el siguiente esquema:



1. Controlar las posibles excepciones producidas y almacenarlas en un archivo txt denominado logs.txt

**Bases de datos**

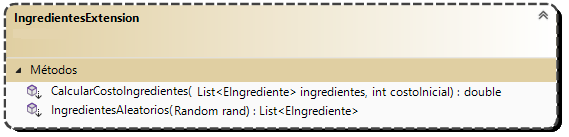
1. Dentro del proyecto se deberá respetar el siguiente esquema:
2. DataBaseManager será estática:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* 1. En el constructor de clase inicializar el string connection.
  2. ***GetImagenComida***, recibirá el tipo de comida, el cual usará para filtrar en la tabla **comidas**. Retornara la columna 2 (contando desde cero), la cual corresponder a la URL de la imagen almacenada en la BD. En caso de que no exista el tipo de comida se lanzara una excepción ***ComidaInvalidaException***.
  3. ***GuardarTicket***, será genérico, solo aceptará tipos que implementen la interfaz IComestible y tengan un constructor publico sin parámetros. Recibirá el nombre del cliente y la comida. En la tabla **tickets** almacenara el nombre del empleado y el ticket de la comida.
  4. Cualquier excepción que se pueda producir se deberá encapsular en una nueva denominada ***DataBaseMangerException***, haciendo referencia al error producido (“leer” o “escribir”)

**Métodos de extensión**



1. ***CalcularCostoIngrediente*** extenderá la clase List<EIngrediente> la cual, adicionalmente recibirá un valor como parámetro, el cual se denomina costo inicial. Su tarea será, tomar el costo inicial e incrementar su valor porcentualmente en base a los valores de la lista de Eingredientes. Retornara este valor incrementado.
2. ***IngredientesAleatorios*** extenderá la clase Random. Su tarea será generar una lista de ingredientes ej:  
    List<EIngrediente> ingredientes = new List<EIngrediente>()

{

EIngrediente.QUESO,

EIngrediente.PANCETA,

EIngrediente.ADHERESO,

EIngrediente.HUEVO,

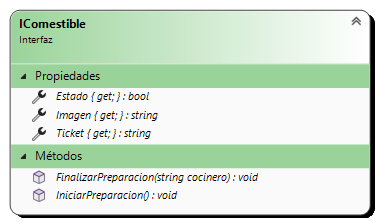
EIngrediente.JAMON,

};

Generará un numero random desde 1 hasta el largo de la lista + 1. Y retornara una lista de ingredientes en base a la cantidad obtenida de forma aleatoria. Usar el siguiente método: return ingredientes.Take(“acá va el numero aleatorio”).ToList();

**Interfaces**

1. Dentro del proyecto se deberá respetar el siguiente esquema:



**Entidades**

1. Dentro del proyecto se deberá respetar el siguiente esquema:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. **Hamburguesa**, deberá implementar IComestible.
   1. ***AgregarIngredientes***, agregara los ingredientes de forma aleatoria.
   2. ***IniciarPreparacion***, si el estado es **false,** generara un numero aleatorio de 1 hasta 9 y asignara la imagen de la hamburguesa, para ello se le deberá enviar al método que obtiene la imagen el siguiente string: $"Hamburguesa\_{“Acá va el numero aleatorio”}". Luego llamara a agregar ingredientes.
   3. ***FinalizarPreparacion***, asignará el costo a la hamburguesa, este será en relación al costo base y los ingredientes de la hamburguesa. Luego cambia el estado de la hamburguesa.
2. **Cocinero**, será genérica, solo podrá recibir tipos que implementen la interfaz **IComestible** y posean un constructor publico sin parámetros:
3. La propiedad HabilitarCocina:
   1. El GET retornara True, si la tares no es nula y estado de la tarea es Running o WaitingToRun o WaitingForActivation.
   2. En el SET, si el valor recibido es TRUE y la tarea es nula o su estado no es Running o no es WaitingToRun o no es WaitingForActivation, se instanciará un nuevo CancelationTokenSource y se llamará a InciarIngreso. De lo contrario se llamará al método Cancel de cancellation.
4. El método ***IniciarIngreso*** será privado y:
   1. Ejecutara en un hilo secundario la acción de que:
      1. Mientras no se requiera cancelación de la tarea de:
         1. Invocara al mensaje NotificarNuevoIngreso y
         2. EsperarProximoIngreso.
         3. Incrementar cantidad de pedidos finalizados en 1.
         4. Guardar ticket en la BD.
5. El método ***NotificarNuevoIngreso***, verificara si el evento OnIngreso posee suscriptores y en caso exitoso realizara:
   1. Instanciara un nuevo menú
   2. Iniciar la preparación del menú.
   3. Notificara el menú.
6. El método ***EsperarProximoIngreso*** si posee un suscriptor notificara los segundos transcurridos mientras que (Utilizar Thread.Sleep para dormir el hilo 1 segundos antes de ir decrementando):
   1. El hilo secundario no requiera cancelación.
   2. El estado del pedido **false**.
   3. Al finalizar incrementar el valor de demoraPreparacionTotal en base al tiempo transcurrido.
7. La propiedad ***TiempoMedioDePreparacion*** retornara el resultante de dividir la demora en preparación total sobre la cantidad de pedidos finalizados

**Formulario**

1. Desarrollar todo lo indicado con comentario //Alumno:

**Test Unitarios**

1. Darle un nombre claro al proyecto, sus clases y sus métodos
2. Agregar 2 test unitarios:
   1. Forzar, mediante el código la ejecución de FileManagerException, validar que suceda de forma correcta.
   2. Al instanciar un nuevo cocinero, la cantidad de pedidos finalizados debe ser igual a 0 (cero).

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