**Universidad Autónoma de Zacatecas**

Unidad Académica de Ingeniería Eléctrica

Programa Académico de Ingeniería de Software.

**Nombre de la Práctica** Listas ligadas.

**Numero de Práctica** 18

**Nombre de la carrera** Ingeniería de Software

**Nombre de la materia** Lab. Estructuras de Datos

**Nombre del alumno** Jesús Manuel Juárez Pasillas

**Nombre del docente** Aldonso Becerra Sánchez

**Fecha:** 04/10/2021

**Práctica 18:** Listas ligadas.

**Introducción:**

Las listas ligadas nos permiten tener un gran numero indefinido de elementos dentro de la lista, por lo que resultan ser muy eficaces a la hora de utilizarlas para almacenar una gran cantidad de elementos y los cuales no tienen un numero definido como para utilizar arreglos. Con estas listas es fácil utilizarlas para dar soluciones a problemas más reales.

**Desarrollo:**

En esta practica se nos pide dar solución a unos cuantos puntos de un tema en particular el cual es de un restaurante, las comidas que se tienen, sus ingredientes y los cocineros que las hacen. Teniendo esto como problemática principal se requiere que se obtengan los siguientes datos de relevancia.

1. Imprimir los nombres de las comidas (menú de platillos al cliente).
2. Imprimir todas las comidas y todos sus ingredientes.
3. Dado el nombre de una comida, imprima los ingredientes.
4. Dado el nombre de una comida agregue ingredientes.
5. Dado el nombre de alguna comida, elimine ingredientes.
6. Dé de alta una comida con todos sus ingredientes.
7. Dé de baja una comida con todos sus ingredientes.
8. Buscar las comidas que tengan un ingrediente X.
9. Que busque todos los platillos del chef Y.
10. Que muestre los platillos que necesitan estufa, horno, etc. Esto se define en el procedimiento de cocinado.
11. Que muestre los platillos que requieren más de 100 gramos para su preparación.
12. Que muestre los platillos que requieren ingredientes líquidos.
13. Que muestre los platillos que requieren cuchillo para su preparación.
14. Que muestre los platillos que no requieren electricidad para su preparación.
15. Que muestre el menú de platillos vegetariano.
16. Que elimine de manera global todos los platillos carnívoros.

A estas clases se les creo su respectiva interface, excepto por la clase Cocinero y por los enumerados TipoComida y TipoIngrediente.

Para dar solución a esto, se comenzó por hacer las clases y agregar los atributos que se obtuvieron de la conclusión del problema. Estas clases son:

* **Restaurante:** Esta clase será la clase principal, la cual manejara todas las comidas, es por ello que se le agrego un atributo de tipo ListaEncadenada llamado “comidas”, esta lista almacenara Objetos de tipo Comida.
  + ListaEncadenada comidas.
* **Comida:** Esta clase alamacenara los atributos de una comida, esto incluye el nombre, preparación, cocinero y una lista de ingredientes.
  + String nombre.
  + String preparación.
  + Cocinero cocinero.
  + ListaEncadenada ingredientes.
  + TipoComida tipo.
* **Cocinero:** Esta clase contendrá el nombre de un cocinero.
  + String nombre.
* **Ingrediente:** Esta clase contendrá los atributos de un ingrediente, el nombre, descripción, cantidad y el tipo. El tipo se definira por el enumerado TipoIngrediente el cual contiene las opciones de SOLIDO y LIQUIDO.
  + String nombre.
  + String descripción.
  + int cantidad.
  + TipoIngrediente tipo.

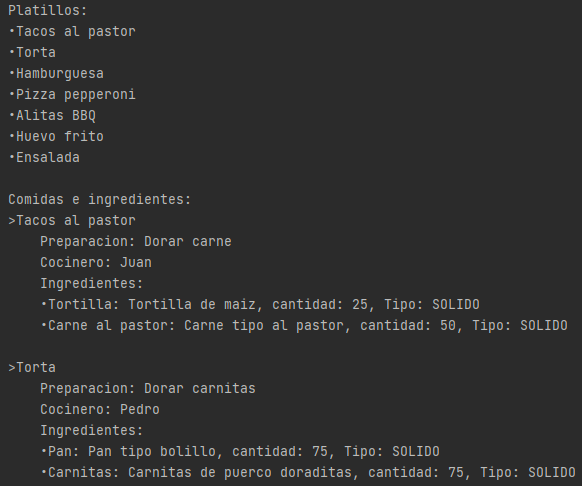
A estas clases se les agregaron los siguientes métodos:

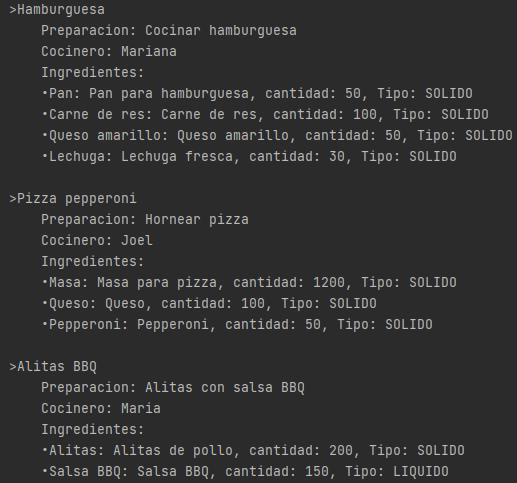
* **Restaurante:** Esta clase es la que más métodos tendrá ya que es la que manejara todos los métodos para acceder a los datos que se quieren obtener.
  + Método que agrega comidas por defecto (Método utilizado solo para agregar elementos a la lista de comidas).  
    private void iniciar();
  + Agrega una comida a la lista.  
    public void agregarComida();
  + Crea un nuevo cocinero y lo regresa.  
    private Cocinero nuevoCocinero();
  + Imprime los nombres de las comidas (menú de platillos para el cliente).   
    public void menuPlatillos();
  + Imprime todas las comidas junto con sus ingredientes.  
    public void comidasIngredientes();
  + Busca la comida con el nombre indicado e imprime sus ingredientes.  
    public boolean ingredientesComida(String nombre);
  + Dado el nombre de una comida, ingresar nuevos ingredientes a esa comida.  
    public void agregarIngredientes(String nombre);
  + Busca la comida a la cual se le quiere eliminar ingredientes.  
    public void eliminarIngredientes(String nombre);
  + Da la opción de eliminar alguna comida.  
    public void eliminarComida();
  + Recorre la lista de comidas e invoca el método de comidasConIngredientes dentro de la clase Comida.  
    public void comidasConIngrediente(String nombre);
  + Imprime las comidas que sean del chef escogido.  
    public void comidasDelChef(String nombre);
  + Muestra los platillos que requieren más de "cantidad" de gramos.  
    public void platillosConMas(int cantidad);
  + Evalúa que platillos requieren ingredientes de algún tipo.  
    public void comidaConIngTipo(TipoIngrediente tipo);
  + Imprime los platillos de un tipo dado.  
    public void platillosTipo(TipoComida tipo);
  + Elimina los platillos que son de un tipo.  
    public void eliminarPlatillosTipo(TipoComida tipo);
* **Comida:** En esta clase estarán algunos métodos mas centrados en atributos de esta clase.
  + Agrega ingredientes a la lista de ingredientes de la comida.  
    public void agregarIngredientes();
  + Regresa todos los atributos de la comida en una sola cadena de texto.  
    public String comida();
  + Elimina ingredientes seleccionados por el usuario.  
    public void eliminarIngredientes();
  + Imprime la comida que tenga el ingrediente pasado como parámetro.  
    public void comidaConIngrediente(String nombre);
* **Ingrediente:** A esta clase solo se le agrego el método el cual regrese una cadena de texto con sus tributos concatenados y también el método toString para comparaciones.
  + Concatena todos los atributos del ingrediente.  
    public String ingrediente();
  + public String toString();
* **Cocinero:** En esta clase solo se le agrego el metodo toString.
  + public String toString();

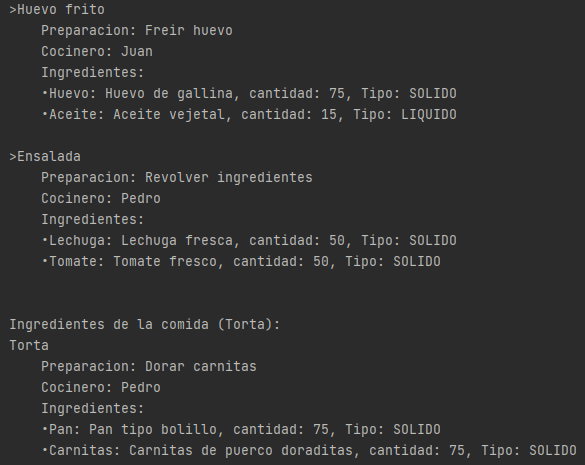
**Nota:** Toda la documentación del proyecto esta agregada en la carpeta “doc” dentro de la carpeta del proyecto (“edylab\_2021\_18/doc”).

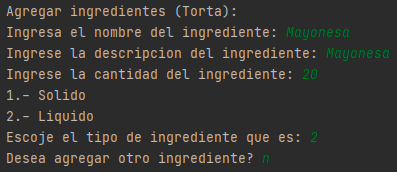
**Capturas del programa funcionando:**

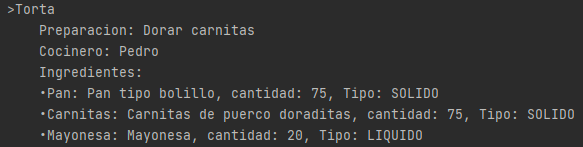
**PruebaRestaurante:** Para una mejor manipulacion del programa se agrego un metodo el cual agrega comida junto con sus ingredientes al iniciar el programa, por lo que ya se tienen datos dentro del programa para ser manipulados.

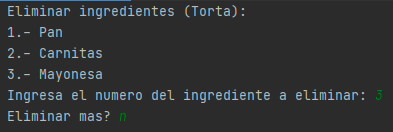


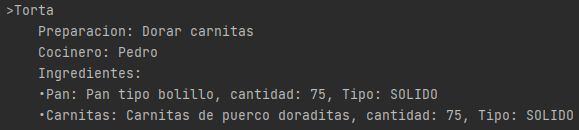


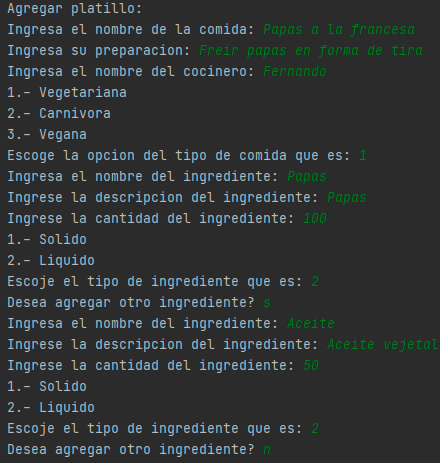


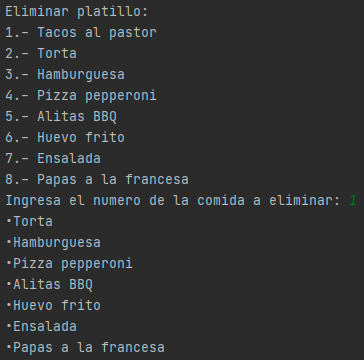


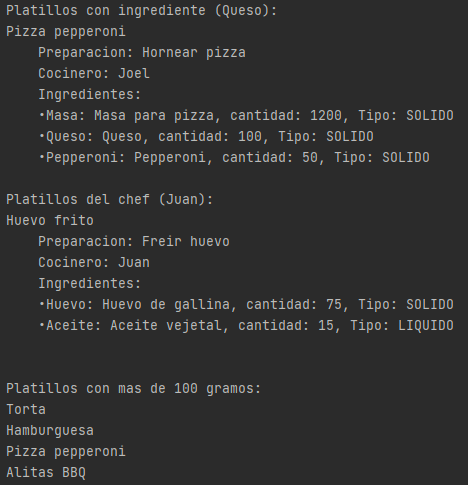


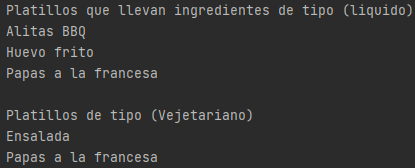








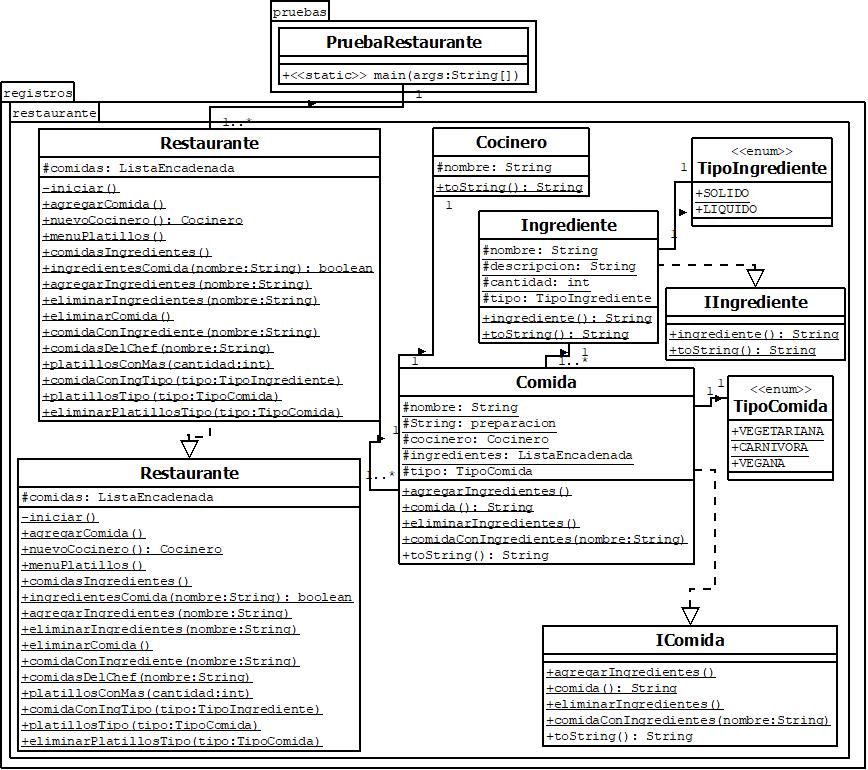






**Código agregado:**

Todo lo que se ve en el diagrama es nuevo, a excepción del paquete “pruebas” y el paquete “registros”.

****

**Pre-evaluación:**

|  |  |
| --- | --- |
| Pre-Evaluación para prácticas de Laboratorio de Estructuras de Datos | PRE-EVALUACIÓN DEL ALUMNO |
| CUMPLE CON LA FUNCIONALIDAD SOLICITADA. | No |
| DISPONE DE CÓDIGO AUTO-DOCUMENTADO. | Sí |
| DISPONE DE CÓDIGO DOCUMENTADO A NIVEL DE CLASE Y MÉTODO. | Sí |
| DISPONE DE INDENTACIÓN CORRECTA. | Sí |
| CUMPLE LA POO. | Sí |
| DISPONE DE UNA FORMA FÁCIL DE UTILIZAR EL PROGRAMA PARA EL USUARIO. | Sí |
| DISPONE DE UN REPORTE CON FORMATO IDC. | Sí |
| LA INFORMACIÓN DEL REPORTE ESTÁ LIBRE DE ERRORES DE ORTOGRAFÍA. | Sí |
| SE ENTREGÓ EN TIEMPO Y FORMA LA PRÁCTICA. | Sí |
| INCLUYE LA DOCUMENTACIÓN GENERADA CON JAVADOC. | Sí |
| INCLUYE EL CÓDIGO AGREGADO EN FORMATO UML. | Sí |
| INCLUYE LAS CAPTURAS DE PANTALLA DEL PROGRAMA FUNCIONANDO. | Sí |
| LA PRÁCTICA ESTÁ TOTALMENTE REALIZADA (ESPECIFIQUE EL PORCENTAJE COMPLETADO). | 90% |
| Observaciones: | |

**Conclusión:**

Todo problema donde se tenga que tener una cantidad indefinida de elementos, como en este caso las comidas de un restaurante, es muy conveniente utilizar listas encadenadas para almacenar estos elementos, ya que estas nos permiten tener una gran cantidad de elementos almacenados, y sin tener que definirle un tamaño predeterminado. Esto nos da solución al problema que se tenia con los arreglos que se tenia que utilizar siempre una cantidad definida de elementos.