



# Tecnológico Nacional de México Campus Colima

## Problema de programación, Restaurant.

Luis Javier Robles Topete

Ingeniería en Sistemas Computacionales

Programación Orientada a Objetos

Proyecto Final



### TopFood

Este programa de escritorio, desarrollado en Java y operado desde la línea de comandos (CLI), permite a una cafetería gestionar de forma eficiente sus mesas, pedidos y productos, todo almacenado en archivos locales.



### Funcionalidades



#### Gestión de Meseros

- Registro y eliminación de meseros.
- Inicio de sesión mediante **código de mesero** y contraseña.



#### Gestión de Mesas

- Cada mesa tiene:
  - Un número distintivo.
  - Un mesero asignado.
  - Estado de cuenta (solicitada o no).
  - Total del consumo.
  - Registro de alimentos ordenados.
- Funciones:
  - Tomar pedidos.
  - Unir cuentas entre mesas.
  - Imprimir cuenta.
  - Eliminar cuenta para reutilizar la mesa.



#### Registro de Alimentos

- Cada alimento incluye:
  - Nombre.
  - Precio base.
  - Comentario opcional (ej. sin azúcar, poco picante).
  - Indicador de disponibilidad.



#### Café

- Parámetros personalizados:
  - Con o sin cafeína.
  - Caliente o con hielo.
  - Tipo de leche (normal o almendra).
  - Tamaño: chico, mediano (+10%), grande (+20%).
- Si se elige leche de almendra, se suma un costo adicional.

## 1. Problema.

Una cafetería necesita un programa que agilice la gestión del servicio en sus mesas. El sistema debe permitir que el mesero, quien contará con un nombre, un código identificador y una contraseña para iniciar sesión, pueda tomar pedidos, unir cuentas, imprimir la cuenta de una mesa y eliminarla del registro para reutilizarla después. Cada mesa debe tener un mesero asignado, tener un número distintivo, registrar si ha solicitado la cuenta, calcular el total de consumo y llevar el control de los alimentos ordenados. Los alimentos deben registrarse con nombre, precio, un comentario opcional para especificaciones del cliente y una indicación de si están disponibles o no para su preparación. En el caso del café, se debe especificar si contiene cafeína, si se sirve caliente o con hielo, el tipo de leche y el tamaño (chico, mediano o grande). El precio del café puede variar: si se elige leche de almendras, se incrementa un costo adicional, y si el tamaño es mediano o grande, el precio base aumenta en un 10 % o 20 % respectivamente. Para los snacks o postres, debe indicarse si se trata de una porción individual, una porción completa o un paquete, y el precio también puede depender del tamaño. Los paquetes o versiones completas (como un pastel entero o una caja de galletas) están compuestos por un número definido de porciones individuales, y su precio equivale a la suma de las porciones individuales menos el costo de dos de ellas; por ejemplo, si un paquete contiene 10 galletas, se cobra el precio equivalente a 8 galletas. Además, el sistema debe permitir llevar un control tanto de los alimentos como de los meseros, permitiendo agregar nuevos elementos al menú, registrar o eliminar meseros, y dar de alta o baja productos según su disponibilidad.

## 2. Escenario de ejemplo.

### 1. Gestión del Menú:

- El administrador registra los alimentos disponibles en el sistema. Cada alimento incluye su nombre, precio y un indicador de disponibilidad.
- Por ejemplo, se ingresan:
  - Nombre: "Pastel", "Zanahoria", Precio: \$55.00, Disponible: Sí.
  - Nombre: "Muffin ", "Chocolate", Precio: \$40.00, Disponible: Sí.
  - Nombre: "Galletas", "Avena", Precio: \$30.00, Disponible: No.
- El barista o encargado de bebidas registra las opciones disponibles con información detallada como si contienen cafeína, el tipo de leche y el tamaño.
- Ejemplos:
  - Nombre: "Café Americano", Precio: \$38.00, Disponible: Sí.
  - Nombre: "Latte Frío", Precio: \$52.00, Disponible: Sí.

### 2. Atención al Cliente:

- Llega un grupo de 3 personas y es asignado a la Mesa 2. El mesero Luis (código: 21) los atiende.
- El mesero inicia sesión en el sistema con su código y contraseña, consulta la disponibilidad de alimentos y les proporciona el menú en carta.

### **3. Realización del Pedido:**

- Los clientes deciden ordenar:
- 1 "Latte Frío", descafeinado, grande, con leche de almendra.
- 2 "Pastel de Zanahoria", una rebanada, sin decoracion,.
- El mesero Luis registra el pedido en el sistema, agregando observaciones específicas. El sistema verifica disponibilidad y concreta la orden.

### **4. Control de Mesas:**

- El sistema registra que la Mesa 2 está ocupada y es atendida por el mesero Luis, por lo que el mesero Raúl no puede registrar una mesa con ese numero ni por error.
- El mesero cree que pudo haber olvidado registrar un café, por lo que revisa la comanda completa de la mesa para verificar.

### **5. Gestión del Mesero:**

- Si es necesario, el administrador puede eliminar o registrar nuevos meseros en el sistema, asignarles un código único y establecer contraseñas de acceso.
- En caso de que por ejemplo, se terminara el pastel de tres leches, el encargado puede retirarlo de la oferta fácilmente.

### **6. Cierre de la Mesa:**

- Una vez que los clientes terminan de consumir, el mesero genera la cuenta desde el sistema y la marca como terminada.
- La cuenta para la Mesa 2 incluye:
- 1 "Latte Frío": \$52.00
- 2 "Pastel de Zanahoria": \$110.00
- Total: \$162.00
- Al momento de pagar, el sistema marca la Mesa 2 como disponible para futuros comensales y el registro del ticket queda almacenado para futuras consultas o reportes.

### 3. Encontrar los objetos principales.

Una cafetería necesita un **programa** que agilice la gestión del **servicio** en sus **mesas**. El sistema debe permitir que el **mesero**, quien contará con un **nombre**, un **código** identificador y una **contraseña** para iniciar sesión, pueda tomar **pedidos**, unir **cuentas**, imprimir la cuenta de una mesa y eliminarla del **registro** para reutilizarla después. Cada **mesa** debe tener un **mesero** asignado, tener un **número distintivo**, registrar si ha solicitado la cuenta, calcular el **total de consumo** y llevar el **control** de los alimentos ordenados. Los **alimentos** deben registrarse con **nombre**, **precio**, un **comentario** opcional para especificaciones del cliente y una indicación de si están **disponibles** o no para su preparación. En el caso del **café**, se debe especificar si contiene **cafeína**, si se sirve **caliente** o con **hielo**, el tipo de **leche** y el **tamaño** (**chico**, **mediano** o **grande**). El **precio** del café puede **variar**: si se elige **leche de almendras**, se incrementa un costo adicional, y si el tamaño es mediano o grande, el **precio base** aumenta en un **10 %** o **20 %** respectivamente. Para los **snacks o postres**, debe indicarse si se trata de una **porción individual**, una **porción completa** o un **paquete**, y el **precio** también puede depender del **tamaño**. Los **paquetes** o versiones completas (como un **pastel** entero o una caja de galletas) están compuestos por un **número definido de porciones individuales**, y su **precio** equivale a la **suma** de las porciones individuales menos el **costo** de **dos** de ellas; por ejemplo, si un paquete contiene 10 galletas, se cobra el precio equivalente a 8 galletas. Además, el **sistema** debe permitir llevar un **control** tanto de los **alimentos** como de los **meseros**, permitiendo **agregar nuevos elementos** al menú, **registrar** o **eliminar meseros**, y **dar de alta** o **baja productos** según su **disponibilidad**.

**Los objetos principales identificados en el problema son:**

- Comandera: El lugar donde se gestiona y se realizan los pedidos.
- Snacks: Acompañantes para la bebida.
- Cafés: Bebidas calientes y frías de la barra de cafetería.
- Mesa: Personas que realizan el pedido.
- Mesero: El empleado que toma los pedidos y los comunica a la cocina.

Mesa

Comandera

Mesero

Alimento

Cafe

Snack

## 4. Definir los datos miembro.

Los datos miembro (atributos) para cada objeto son:

- **Alimento:**
  - Nombre: El nombre del platillo (por ejemplo, "Pasta Alfredo").
  - Precio: El costo del platillo (por ejemplo, \$120.00).
  - Existencia: Si hay disponibilidad para preparar el pedido requerido o si no es posible.
- **Snack:**
  - Porción individual o grande: Saber si el producto se trata de una pieza o del paquete/pastel completo.
  - Cantidad en paquete: Saber cuantas piezas trae un paquete o cuantas rebanadas trae un pastel completo, esto servirá para moldear el precio.
- **Café:**
  - Cafeína: Importante para la preparación.
  - Caliente o fría: Algunas bebidas pueden servirse tanto calientes como frías.
  - Leche: El tipo de leche que quiere el cliente o ninguna (Puede aumentar el costo).
  - Tamaño: El cliente puede pedir sus bebidas de distintos tamaños (Aumenta el costo con el tamaño).
- **Mesa:**
  - Número: Identificador único de la mesa (por ejemplo, "Mesa 5").
  - Mesero: Nombre o código del mesero asignado a la mesa.
  - Comensales: Número de personas sentadas en la mesa.
  - Cuenta solicitada: Indicador de si ya se pidió la cuenta.
  - Total: Suma automática del costo de todos los productos ordenados.
  - Pedidos: Lista de alimentos (cafés o snacks) registrados por los clientes.
- **Mesero:**
  - Nombre: Identificación personal del mesero registrado en el sistema.
  - Código: Número único asignado al mesero para distinguirlo.
  - Contraseña: Clave requerida para iniciar sesión en el sistema.

Café
- cafeína : boolean - hielo : boolean - leche : String - size : String - milkList : String[] - sizeList : String[]

Comandera

Mesero
- nombre : String - código : int - password : int

Alimento
- nombre : String - costo : double - comentario : String - existencia : boolean

Snack
- porcionIndividual : boolean - porcionGrande : int

Mesa
- número : int - mesero : Mesero - comensales : int - cuentaSolicitada : boolean - total : double - pedido : Alimento[] - activo : boolean

## 5. Determinar el comportamiento deseado:

- **Mesa:**

- Mesa(): Constructor que inicializa una mesa con número, sin mesero asignado, inactiva y total en 0.
- Mesa(): Constructor que inicializa una mesa con número, mesero asignado, estado activo, pedido vacío y total en 0.
- getMesero(): Muestra el mesero asignado a la mesa.
- getNumero(): Muestra el número de la mesa.
- setNumero(): Asigna un nuevo número a la mesa.
- isActive(): Muestra si la mesa está activa.
- setActive(): Cambia el estado de actividad de la mesa.
- addPedido(): Agrega un alimento al pedido de la mesa.
- getPedido(): Muestra un alimento específico del pedido.
- printPedido(): Muestra todos los alimentos registrados en el pedido.
- addTotal(): Suma un monto al total acumulado de la mesa.
- getTotal(): Muestra el total acumulado del consumo en la mesa.

- **Mesero:**

- Mesero(): Constructor que crea un mesero con valores predeterminados.
- Mesero(): Constructor que crea un mesero con nombre, código y contraseña definidos.
- getNombre(): Muestra el nombre del mesero.
- setNombre(): Asigna el nombre del mesero.
- getCodigo(): Muestra el código del mesero.
- setCodigo(): Asigna el código del mesero.
- getPassword(): Muestra la contraseña del mesero.
- setPassword(): Asigna la contraseña del mesero.

- **Alimento:**

- Alimento(): Constructor que crea un alimento con nombre, costo, comentario y estado de existencia.
- getNombre(): Muestra el nombre del alimento.
- setNombre(): Asigna el nombre del alimento.
- getCosto(): Muestra el costo del alimento.
- setCosto(): Asigna el costo del alimento.
- getComentario(): Muestra el comentario del alimento (retorna una cadena vacía si es null).
- setComentario(): Asigna un comentario al alimento.
- isExistencia(): Muestra si el alimento está disponible.
- setExistencia(): Cambia el estado de disponibilidad del alimento.

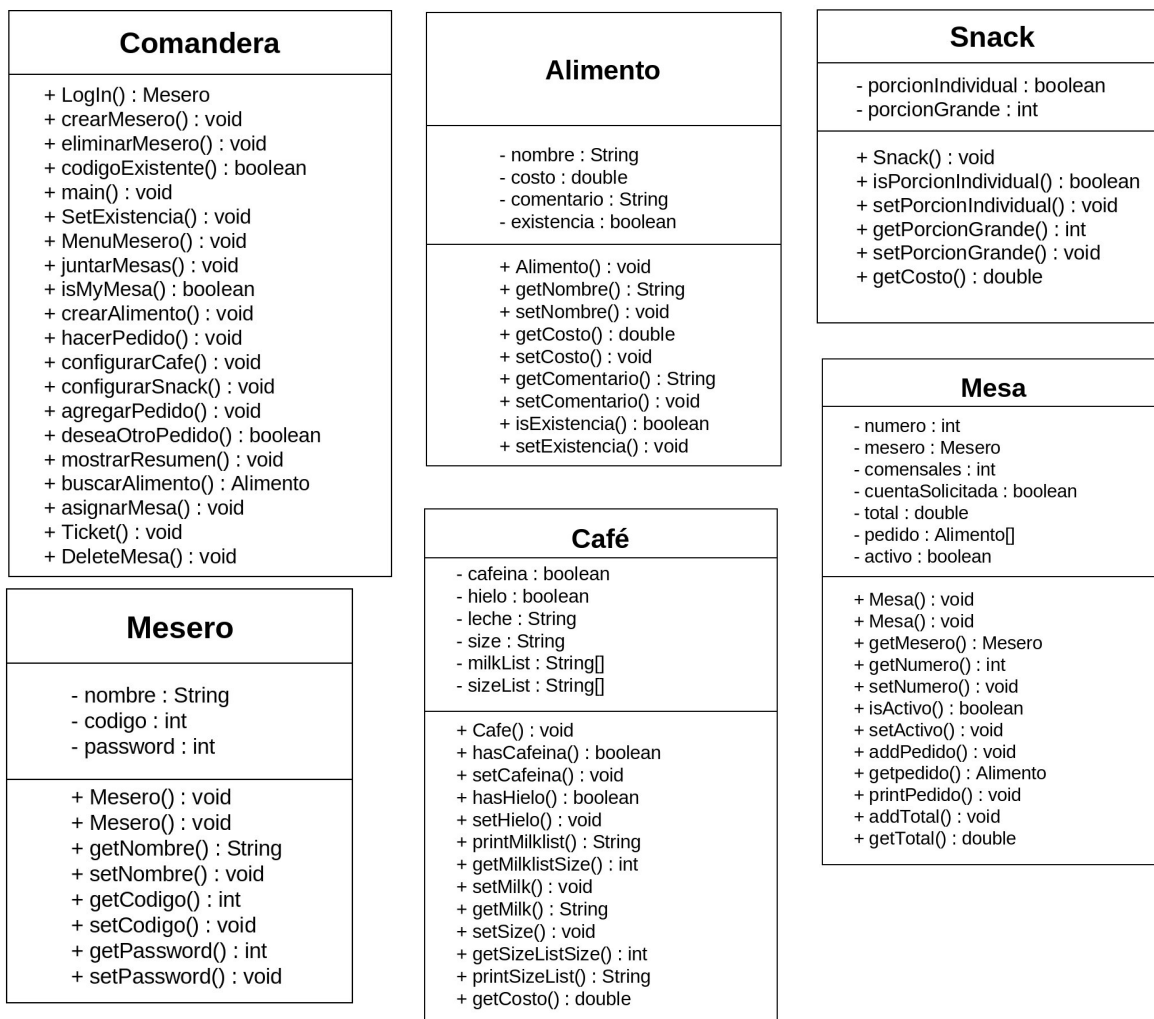
- o **Snack:**

- Snack(): Constructor que crea un snack con información base del alimento y cantidad de porciones grandes, marcando por defecto como porción individual.
- isPorcionIndividual(): Muestra si el snack es una porción individual.
- setPorcionIndividual(): Asigna si el snack es una porción individual.
- getPorcionGrande(): Muestra la cantidad de porciones que contiene una presentación grande o paquete.

- setPorcionGrande(): Asigna la cantidad de porciones que contiene una presentación grande o paquete.
- getCosto(): Muestra el costo del snack, aplicando descuento si no es porción individual (precio = costo × (porciones - 2)).

#### o Café:

- Cafe(): Constructor que crea un café con valores por defecto: contiene cafeína, sin hielo, leche normal y tamaño chico.
- hasCafeina(): Muestra si el café contiene cafeína.
- setCafeina(): Asigna si el café contiene cafeína.
- hasHielo(): Muestra si el café se sirve con hielo.
- setHielo(): Asigna si el café se sirve con hielo.
- printMilklist(): Devuelve un listado en texto de las opciones de leche disponibles.
- getMilklistSize(): Muestra cuántas opciones de leche existen.
- setMilk(): Asigna el tipo de leche según su índice en la lista.
- getMilk(): Muestra el tipo de leche asignado al café.
- setSize(): Asigna el tamaño del café según su índice en la lista.
- getSizeListSize(): Muestra cuántos tamaños disponibles existen.
- printSizeList(): Devuelve un listado en texto de los tamaños de café disponibles.
- getCosto(): Muestra el costo total del café, ajustado según el tipo de leche y tamaño seleccionado.



## 6. Asociación:

