Universidad Tecmilenio

ESTRUCTURAS DE DATOS

Alumnos:

Edwin Manuel Aguilar Márquez

Jesús Alberto Sandoval Serrano

Citlaly Guadalupe Zeferino Sierra

Avance Proyecto Final

1. Introducción.

En este proyecto se utilizará lo aprendido en la materia de Estructuras de datos para la creación de una aplicación sobre una Taquería, en la cual se usaran las Colas para registrar el número de clientes y así atenderlos de una manera más ordenada y rápida, las Pilas se utilizaran para registrar el número de platos sucios, lavarlos y quitar el plato limpio de la pila de los sucios, las Colas de Prioridad para utilizar eficientemente los ingredientes para que no se caduquen, Las listas enlazadas para poder mostrar el menú a los clientes, esto incluye una breve descripción de cada platillo. Elegimos la opción de la Taquería ya que implementa los temas revisados en clase. La cola de clientes se usa porque los clientes deben de ser atendidos en base al orden en el que llegaron (FIFO), Pilas para los platos se usa porque los plantos sucios que entran son los primero en que se lavan (LIFO)

2. Diseño del programa:

a. **Estructura del programa:** divide tu programa en tres partes principales, cada una correspondiente a una estructura de datos (pila, cola, lista).

b. **Pilas:** utiliza pilas para manejar tareas urgentes que deben realizarse lo antes posible. implementa operaciones básicas como *push*, *pop* y *peek*.

c. **Colas:** emplea colas para administrar tareas en una secuencia FIFO (primero en entrar, primero en salir), ideal para tareas programadas regularmente. implementa operaciones básicas como **enqueue**, **dequeue** y **front**.

d. **listas:** usa listas para tareas que pueden requerir acceso aleatorio, como la consulta de tareas pendientes por departamento. implementa operaciones básicas como **insert**, **delete** y **find**.

3. Funcionalidad adicional:

a. Implementa una interfaz de usuario sencilla en la consola para interactuar con el sistema.

b. Permite al usuario agregar, ver y eliminar tareas en cada estructura de datos.

c. Incluye una opción para ver todas las tareas pendientes, ordenadas por urgencia y departamento.

El objetivo de la actividad es simular cómo se administran los clientes,platos, ingredientes y el menú aplicando los conceptos de colas, pilas, listas y colas de prioridad.

En la primera parte se encuentra la clase Node, que nos sirve como base para todas las estructuras de datos. Este Nodo puede aguardar el dato de un puntero al siguiente Nodo y en las colas prioritarias tiene un campo de prioridad.

```java
// Clase Nodo - Estructura básica para todas nuestras estructuras de datos
class Node {
    Object dato;           // Almacena el dato (puede ser cualquier tipo de objeto)
    Node siguiente;        // Puntero al siguiente nodo en la estructura
    int prioridad;         // Campo especial para cola de prioridad

    // Constructor básico para nodos normales
    public Node(Object dato) {
        this.dato = dato;
        this.siguiente = null;
        this.prioridad = 0;
    }

    // Constructor para nodos con prioridad (ingredientes)
    public Node(Object dato, int prioridad) {
        this.dato = dato;
        this.siguiente = null;
        this.prioridad = prioridad;
    }
}
```

```java
// Clase Lista Enlazada - Base para todas nuestras estructura
class LinkedList {
    private Node cabeza;   // Primer nodo de la lista
    private int tamaño;    // Contador de elementos

    public LinkedList() {
        cabeza = null;
        tamaño = 0;
    }
```

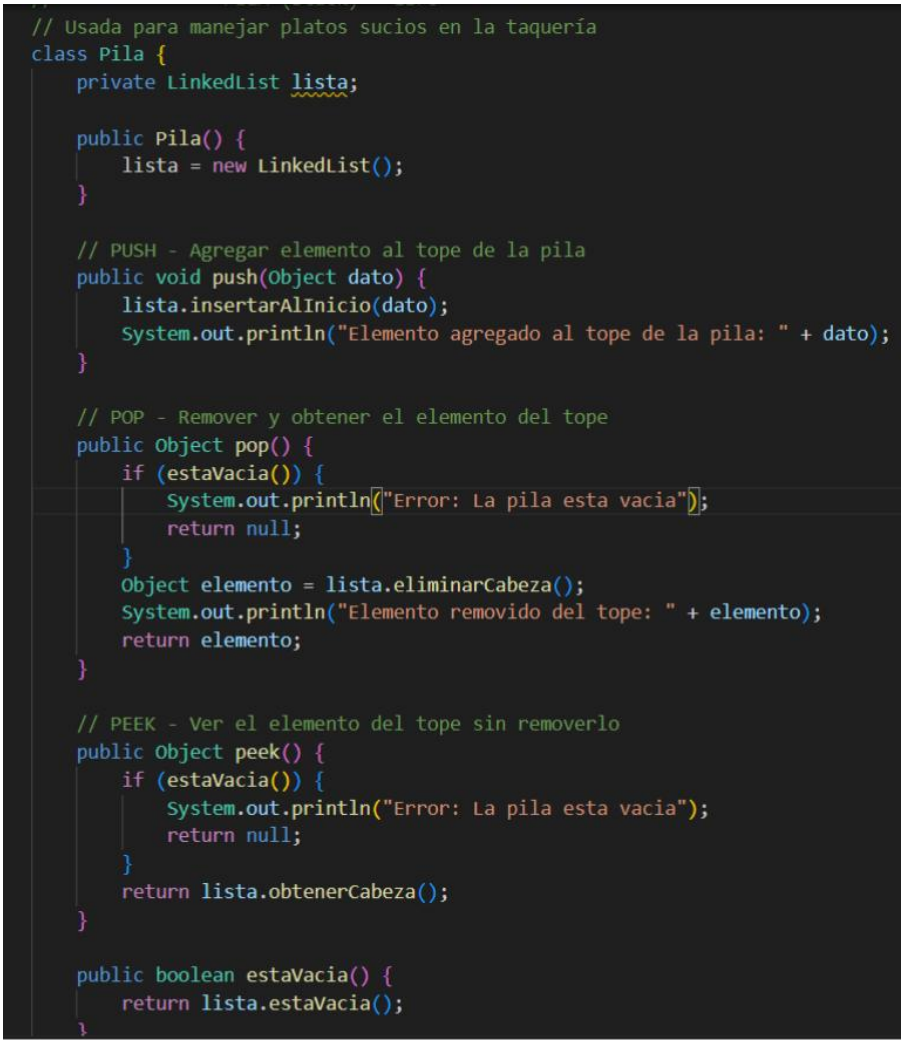
```java
// Insertar al inicio - Útil para pilas
public void insertarAlInicio(Object dato) {
    Node nuevo = new Node(dato);
    nuevo.siguiente = cabeza;
    cabeza = nuevo;
    tamaño++;
}

// Insertar al final - Útil para colas
public void insertarAlFinal(Object dato) {
    Node nuevo = new Node(dato);
    if (cabeza == null) {
        cabeza = nuevo;
    } else {
        Node actual = cabeza;
        while (actual.siguiente != null) {
            actual = actual.siguiente;
        }
        actual.siguiente = nuevo;
    }
    tamaño++;
}
```

Implementamos la clase LinkedList, está nos sirve como la parte central que manejan operaciones básicas como insertar el inicio o insertar al final, puede eliminar Nodos, consultar elementos, buscar por posición, mostrar la lista y verificar si está vacía.

```java
// Eliminar el primer elemento
public Object eliminarCabeza() {
    if (cabeza == null) return null;
    Object dato = cabeza.dato;
    cabeza = cabeza.siguiente;
    tamaño--;
    return dato;
}

// Eliminar el último elemento
public Object eliminarUltimo() {
    if (cabeza == null) return null;
    if (cabeza.siguiente == null) {
        Object dato = cabeza.dato;
        cabeza = null;
        tamaño--;
        return dato;
    }
    Node actual = cabeza;
    while (actual.siguiente.siguiente != null) {
        actual = actual.siguiente;
    }
    Object dato = actual.siguiente.dato;
    actual.siguiente = null;
    tamaño--;
    return dato;
}
```

```java
// Verificar si la lista está vacía
public boolean estaVacia() {
    return cabeza == null;
}

// Mostrar todos los elementos
public void mostrar() {
    Node actual = cabeza;
    int contador = 1;
    while (actual != null) {
        System.out.println(contador + ". " + actual.dato);
        actual = actual.siguiente;
        contador++;
    }
    if (contador == 1) System.out.println("Lista vacía");
}
```

Las clases pilas usan listas enlazadas para el comportamiento LIFO, el último elemento en entrar es el primero en salir, este se usa para los platos sucios que se van apilando y debe lavarse en orden inverso al que llegan.



```java
// Usada para manejar platos sucios en la taquería
class Pila {
    private LinkedList lista;

    public Pila() {
        lista = new LinkedList();
    }

    // PUSH - Agregar elemento al tope de la pila
    public void push(Object dato) {
        lista.insertarAlInicio(dato);
        System.out.println("Elemento agregado al tope de la pila: " + dato);
    }

    // POP - Remover y obtener el elemento del tope
    public Object pop() {
        if (estaVacia()) {
            System.out.println("Error: La pila esta vacia");
            return null;
        }
        Object elemento = lista.eliminarCabeza();
        System.out.println("Elemento removido del tope: " + elemento);
        return elemento;
    }

    // PEEK - Ver el elemento del tope sin removerlo
    public Object peek() {
        if (estaVacia()) {
            System.out.println("Error: La pila esta vacia");
            return null;
        }
        return lista.obtenerCabeza();
    }

    public boolean estaVacia() {
        return lista.estaVacia();
    }
```

La clase de colas nos apoya en las listas enlazadas, pero los implementa un esquema FIFO, En donde los primeros clientes en llegar son los primeros en atendidos

Las colas de prioridad se generan directamente con los Nodos, este nos ordena los productos por su nivel de prioridad, y se utiliza para para manejar los ingredientes en función de su fecha de caducidad.

```java
// Usada para manejar la fila de clientes en la taquería
class Cola {
    private LinkedList lista;

    public Cola() {
        lista = new LinkedList();
    }

    // ENQUEUE - Agregar elemento al final de la cola
    public void enqueue(Object dato) {
        lista.insertarAlFinal(dato);
        System.out.println("Cliente agregado a la cola: " + dato);
    }

    // DEQUEUE - Remover y obtener el primer elemento de la cola
    public Object dequeue() {
        if (estaVacia()) {
            System.out.println("Error: No hay clientes en la cola");
            return null;
        }
        Object elemento = lista.eliminarCabeza();
        System.out.println("Cliente atendido: " + elemento);
        return elemento;
    }

    // PEEK/FRONT - Ver el primer elemento sin removerlo
    public Object front() {
        if (estaVacia()) {
            System.out.println("Error: No hay clientes en la cola");
            return null;
        }
        return lista.obtenerCabeza();
    }

    public boolean estaVacia() {
```

```java
public ColaPrioridad() {
    cabeza = null;
    tamaño = 0;
}

// ENQUEUE con prioridad - Insertar elemento según su prioridad
public void enqueue(Object dato, int prioridad) {
    Node nuevo = new Node(dato, prioridad);

    // Si la cola está vacía o el nuevo elemento tiene mayor prioridad que el primero
    if (cabeza == null || prioridad < cabeza.prioridad) {
        nuevo.siguiente = cabeza;
        cabeza = nuevo;
    } else {
        // Buscar la posición correcta para insertar según prioridad
        Node actual = cabeza;
        while (actual.siguiente != null && actual.siguiente.prioridad <= prioridad) {
            actual = actual.siguiente;
        }
        nuevo.siguiente = actual.siguiente;
        actual.siguiente = nuevo;
    }
    tamaño++;
    System.out.println("Ingrediente agregado con prioridad " + prioridad + ": " + dato);
}

// DEQUEUE - Remover elemento de mayor prioridad
public Object dequeue() {
    if (estaVacia()) {
        System.out.println("Error: No hay ingredientes en la cola de prioridad");
        return null;
    }
    Object elemento = cabeza.dato;
    int prioridad = cabeza.prioridad;
    cabeza = cabeza.siguiente;
```

En la clase Elemento Menú en el código representa cada platillo de nuestra taqueria, contiene el nombre, la descripción y el precio.

```java
class ElementoMenu {
    private String nombre;
    private String descripcion;
    private double precio;

    public ElementoMenu(String nombre, String descripcion, double precio) {
        this.nombre = nombre;
        this.descripcion = descripcion;
        this.precio = precio;
    }

    public String getNombre() { return nombre; }
    public String getDescripcion() { return descripcion; }
    public double getPrecio() { return precio; }

    @Override
    public String toString() {
        return nombre + " - $" + precio;
    }

    public String detalles() {
        return "=== " + nombre + " ===" +
               "\nDescripción: " + descripcion +
               "\nPrecio: $" + precio;
    }
}
```

En la clase de lista menú contiene una lista enlazada, inicia con tacos, quesadillas y bebidas, y permite mostrar la lista o acceder a los detalles de cada elemento según la posición seleccionada. También los permite mostrar el menú, eliminar un elemento del menú agregar al menú y buscar en el menú.

```java
class ListaMenu {
    private LinkedList lista;

    public ListaMenu() {
        lista = new LinkedList();
        inicializarMenu();
    }

    // Inicializar menú con elementos predeterminados
    private void inicializarMenu() {
        agregar(new ElementoMenu("Taco de Carnitas",
                "Delicioso taco de carnitas con cebolla, cilantro y salsa verde", 25.0));
        agregar(new ElementoMenu("Taco de Pollo",
                "Taco de pollo a la plancha con pico de gallo y salsa roja", 20.0));
        agregar(new ElementoMenu("Taco de Pastor",
                "Taco al pastor con piña, cebolla, cilantro y salsa", 22.0));
        agregar(new ElementoMenu("Quesadilla",
                "Quesadilla de queso Oaxaca con opción de agregar carne", 30.0));
        agregar(new ElementoMenu("Agua de Horchata",
                "Refrescante agua de horchata casera", 15.0));
    }

    // INSERT - Agregar elemento al menú
    public void agregar(ElementoMenu elemento) {
        lista.insertarAlFinal(elemento);
    }

    // FIND - Buscar elemento por número de posición (MÉTODO CORREGIDO)
    public ElementoMenu buscar(int posicion) {
        if (posicion < 1 || posicion > lista.getTamaño()) {
            return null;
        }

        Object elemento = lista.obtenerPorPosicion(posicion);
```

La clase principal Taquería, Aquí se integran las estructuras: una cola para gestionar clientes en orden de llegada, una pila para administrar los platos sucios que deben lavarse, una cola de prioridad para el control de ingredientes y una lista enlazada que almacena el menú de la taquería.

```java
// ============= CLASE PRINCIPAL DEL SISTEMA =============
public class Taqueria {
    private static Scanner scanner = new Scanner(System.in);

    // Instancias de nuestras estructuras de datos
    private Cola colaClientes;           // FIFO para atender clientes en orden
    private Pila pilaPlatos;             // LIFO para manejar platos sucios
    private ColaPrioridad colaIngredientes; // Para ingredientes por fecha de caducidad
    private ListaMenu menu;              // Lista enlazada para acceso aleatorio al menú

    public Taqueria() {
        colaClientes = new Cola();
        pilaPlatos = new Pila();
        colaIngredientes = new ColaPrioridad();
        menu = new ListaMenu();

        System.out.println("Bienvenido al Sistema de Gestion de la Taqueria!");
    }
```

Se construyen los métodos para colas, pilas, colas de prioridad y listas

```java
// ============ MÉTODOS PARA COLA DE CLIENTES ============

private void agregarCliente() {
    System.out.print("Ingrese el nombre del cliente: ");
    String nombreCliente = scanner.nextLine();
    colaClientes.enqueue("Cliente: " + nombreCliente);
    System.out.println("Cliente agregado a la fila de espera.");
}

private void atenderCliente() {
    if (colaClientes.estaVacia()) {
        System.out.println("No hay clientes esperando.");
        return;
    }

    Object cliente = colaClientes.front();
    System.out.println("Atendiendo a: " + cliente);
    System.out.print("Presione Enter para completar el servicio...");
    scanner.nextLine();
    colaClientes.dequeue();
}

private void verProximoCliente() {
    Object siguiente = colaClientes.front();
    if (siguiente != null) {
        System.out.println("Próximo cliente a atender: " + siguiente);
    }
}

private void mostrarColaClientes() {
    colaClientes.mostrar();
}
```

```java
// ============ MÉTODOS PARA PILA DE PLATOS ============

private void apilarPlatoSucio() {
    System.out.print("Ingrese el número del plato sucio: ");
    String numeroPlato = scanner.nextLine();
    pilaPlatos.push("Plato #" + numeroPlato);
    System.out.println("Plato apilado en la zona de lavado.");
}

private void lavarPlato() {
    if (pilaPlatos.estaVacia()) {
        System.out.println("No hay platos para lavar.");
        return;
    }

    Object plato = pilaPlatos.peek();
    System.out.println("Lavando " + plato + "...");
    System.out.print("Presione Enter para terminar de lavar...
    scanner.nextLine();
    pilaPlatos.pop();
    System.out.println("Plato limpio y listo para usar!");
}

private void verTopePlatos() {
    Object tope = pilaPlatos.peek();
    if (tope != null) {
        System.out.println("Próximo plato a lavar: " + tope);
    }
}

private void mostrarPilaPlatos() {
    pilaPlatos.mostrar();
}
```

```java
// ============ MÉTODOS PARA COLA DE PRIORIDAD (INGREDIENTES) ========

private void agregarIngrediente() {
    System.out.print("Ingrese el nombre del ingrediente: ");
    String ingrediente = scanner.nextLine();

    System.out.println("Seleccione el nivel de prioridad:");
    System.out.println("1. Alta prioridad (1-7 días para caducar)");
    System.out.println("2. Prioridad media (8-30 días para caducar)");
    System.out.println("3. Prioridad baja (1-12 meses para caducar)");
    System.out.print("Opción: ");

    int prioridad = scanner.nextInt();
    scanner.nextLine(); // Limpiar buffer

    if (prioridad >= 1 && prioridad <= 3) {
        colaIngredientes.enqueue(ingrediente, prioridad);
    } else {
        System.out.println("Error: Prioridad debe ser 1, 2 o 3");
    }
}

private void procesarIngrediente() {
    if (colaIngredientes.estaVacia()) {
        System.out.println("No hay ingredientes para procesar.");
        return;
    }

    Object ingrediente = colaIngredientes.peek();
    System.out.println("Procesando ingrediente: " + ingrediente);
    System.out.print("Presione Enter para confirmar uso...");
    scanner.nextLine();
    colaIngredientes.dequeue();
    System.out.println("Ingrediente utilizado correctamente.");
}
```

```java
// ============ MÉTODOS PARA LISTA DEL MENÚ ============

private void mostrarMenu() {
    menu.mostrarMenu();
}

private void verDetallesMenu() {
    System.out.println("=== MENÚ DETALLADO ===");
    menu.mostrarMenu();
    System.out.print("Seleccione un número para ver detalles: ");

    try {
        int opcion = scanner.nextInt();
        scanner.nextLine(); // Limpiar buffer
        menu.mostrarDetalles(opcion);
    } catch (Exception e) {
        System.out.println("Error: Ingrese un numero valido");
        scanner.nextLine();
    }
}

private void mostrarEstadoGeneral() {
    System.out.println("\n==========================================
    System.out.println("          ESTADO GENERAL DE LA TAQUERIA");
    System.out.println("==========================================

    System.out.println("CLIENTES:");
    if (colaClientes.estaVacia()) {
        System.out.println("   No hay clientes esperando");
    } else {
        Object siguiente = colaClientes.front();
        System.out.println("   Proximo a atender: " + siguiente);
    }

    System.out.println("\nPLATOS:");
    if (pilaPlatos.estaVacia()) {
        System.out.println("   No hay platos sucios");
    } else {
        Object tope = pilaPlatos.peek();
        System.out.println("   Proximo a lavar: " + tope);
    }

    System.out.println("\nINGREDIENTES:");
    if (colaIngredientes.estaVacia()) {
        System.out.println("   No hay ingredientes registrados");
    } else {
        Object prioritario = colaIngredientes.peek();
        System.out.println("   Proximo a usar: " + prioritario);
    }

    System.out.println("==========================================");
}

public void mostrarMenuPrincipal() {
    int opcion;

    do {
        System.out.println("\n==========================================");
        System.out.println("      SISTEMA DE GESTION DE TAQUERIA");
        System.out.println("==========================================");
        System.out.println("GESTION DE CLIENTES (Cola - FIFO):");
        System.out.println("  1. Agregar cliente a la fila");
        System.out.println("  2. Atender proximo cliente");
        System.out.println("  3. Ver proximo cliente");
        System.out.println("  4. Mostrar cola completa de clientes");

        System.out.println("\nGESTION DE PLATOS (Pila - LIFO):");
        System.out.println("  5. Apilar plato sucio");
        System.out.println("  6. Lavar plato (del tope)");
        System.out.println("  7. Ver proximo plato a lavar");
```

```
    System.out.println("  8. Mostrar pila completa de platos");

    System.out.println("\nGESTION DE INGREDIENTES (Cola de Prioridad):");
    System.out.println("  9. Agregar ingrediente");
    System.out.println("  10. Usar ingrediente prioritario");
    System.out.println("  11. Ver proximo ingrediente a usar");
    System.out.println("  12. Mostrar cola completa de ingredientes");

    System.out.println("\nMENU DE LA TAQUERIA (Lista Enlazada):");
    System.out.println("  13. Mostrar menu");
    System.out.println("  14. Ver detalles de platillo");

    System.out.println("\nREPORTES:");
    System.out.println("  15. Estado general del sistema");

    System.out.println("\nSALIDA:");
    System.out.println("  16. Salir del sistema");

    System.out.println("==========================================");
    System.out.print("Seleccione una opcion: ");
```

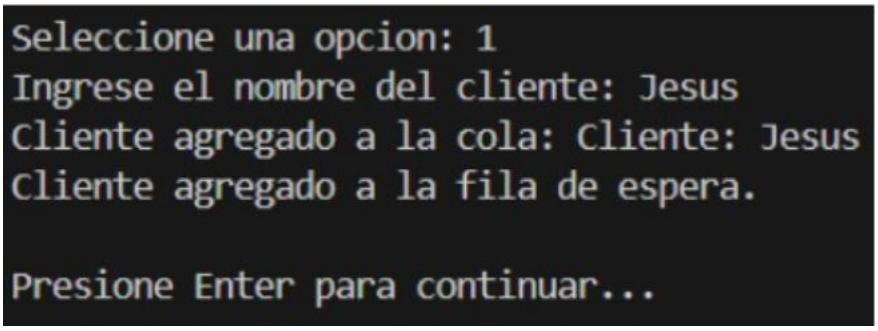
```
try {
    opcion = scanner.nextInt();
    scanner.nextLine();

    switch (opcion) {
        case 1: agregarCliente(); break;
        case 2: atenderCliente(); break;
        case 3: verProximoCliente(); break;
        case 4: mostrarColaClientes(); break;
        case 5: apilarPlatoSucio(); break;
        case 6: lavarPlato(); break;
        case 7: verTopePlatos(); break;
        case 8: mostrarPilaPlatos(); break;
        case 9: agregarIngrediente(); break;
        case 10: procesarIngrediente(); break;
        case 11: verProximoIngrediente(); break;
        case 12: mostrarColaIngredientes(); break;
        case 13: mostrarMenu(); break;
        case 14: verDetallesMenu(); break;
        case 15: mostrarEstadoGeneral(); break;
        case 16:
            System.out.println("Gracias por usar el Sistema de Gestion de la Taqueria!");
            break;
        default:
            System.out.println("Opcion invalida. Seleccione un numero del 1 al 16.");
            break;
    }

    if (opcion != 16) {
        System.out.print("\nPresione Enter para continuar...");
        scanner.nextLine();
    }

} catch (Exception e) {
    System.out.println("Error: Por favor ingrese un numero valido.");
```

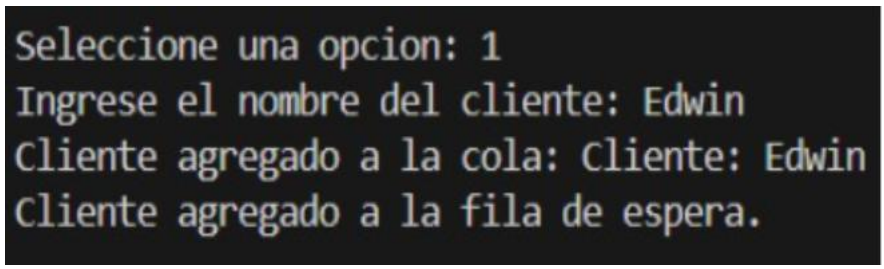
el método main inicia el sistema mostrando un mensaje de bienvenida, explica qué estructuras de datos están implementadas y abre el menú interactivo. A través de este menú el usuario puede navegar por las opciones y simular el funcionamiento de la taquería, hasta decidir salir del programa.

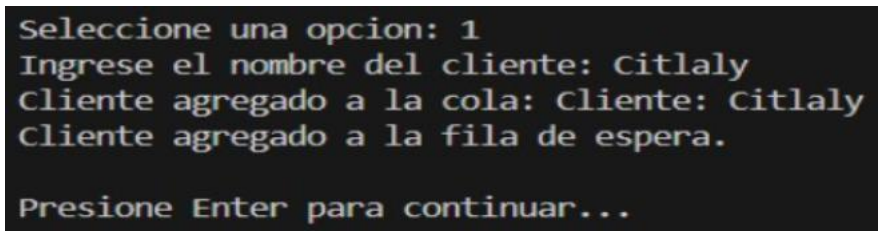
```
Run main | Debug main
public static void main(String[] args) {
    Taqueria taqueria = new Taqueria();

    System.out.println("\nESTRUCTURAS DE DATOS IMPLEMENTADAS:");
    System.out.println("Cola (FIFO): Para manejar clientes en orden de llegada");
    System.out.println("Pila (LIFO): Para manejar platos sucios");
    System.out.println("Cola de Prioridad: Para ingredientes segun fecha de caducidad");
    System.out.println("Lista Enlazada: Para acceso aleatorio al menu");

    System.out.print("\nPresione Enter para comenzar...");
    scanner.nextLine();

    taqueria.mostrarMenuPrincipal();
    scanner.close();
}
```

4. **GESTIÓN DE CLIENTES (Cola – FIFO)**

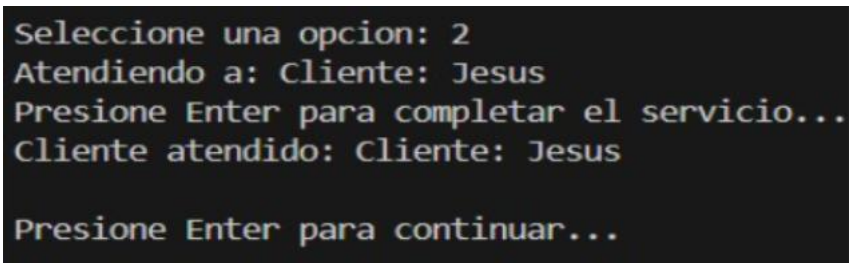
Agregar cliente a la fila



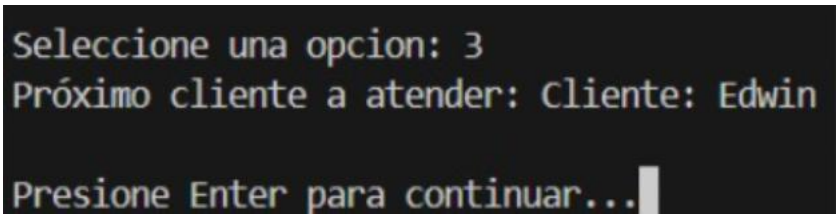




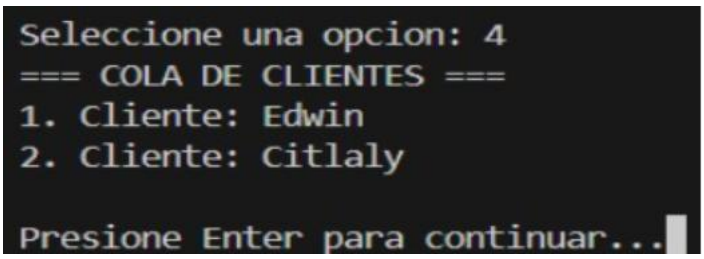
Atender próximo cliente



Ver próximo cliente

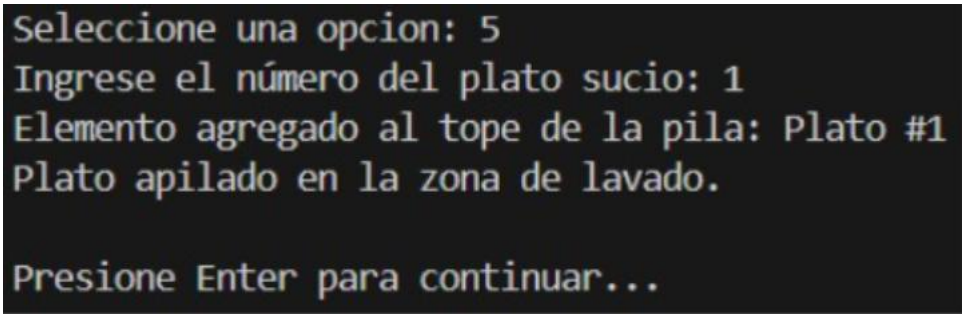


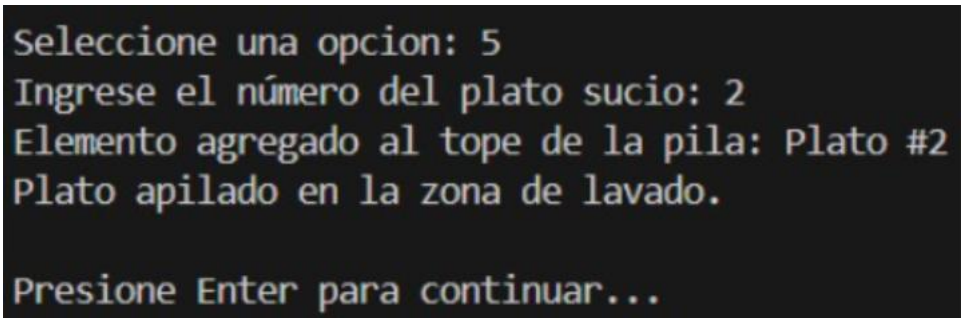
Mostrar cola completa de clientes

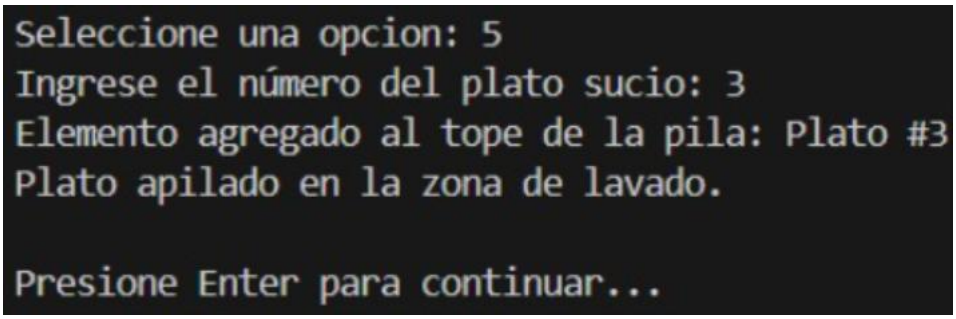


5. GESTIÓN DE PLATOS (Pila – LIFO)

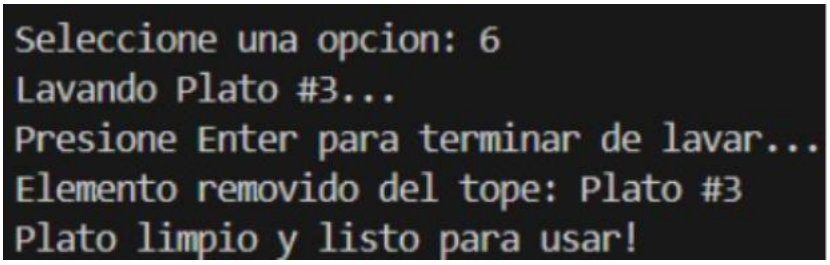
Apilar plato sucio



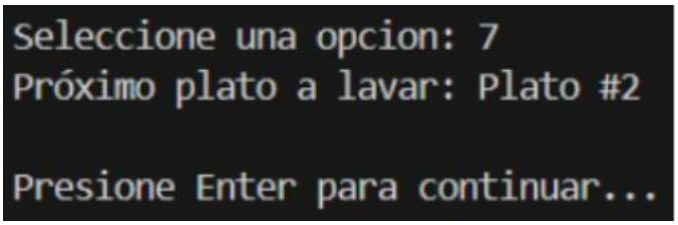




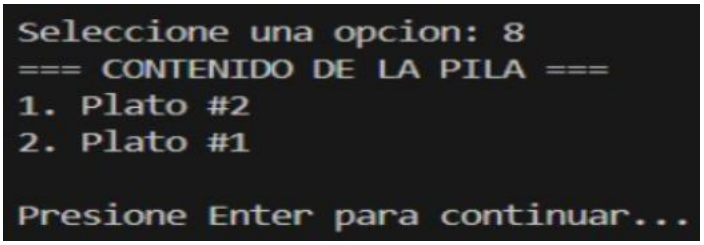
Lavar plato (del tope)



Ver próximo plato a lavar

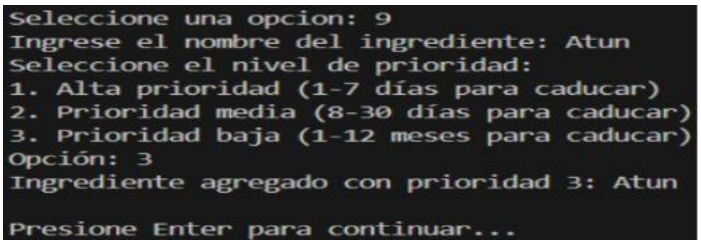


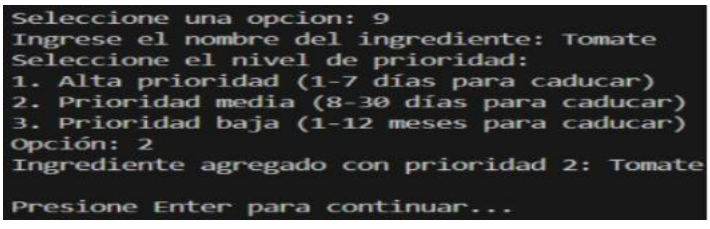
Mostrar pila completa de platos

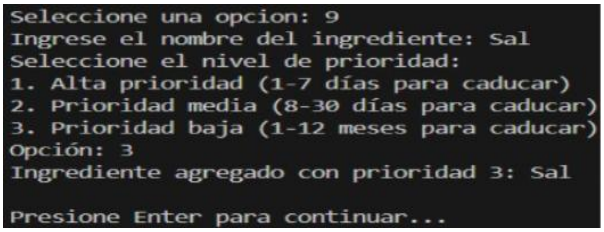


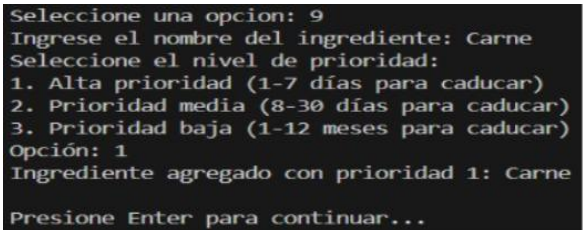
6. GESTIÓN DE INGREDIENTES (Cola de Prioridad)

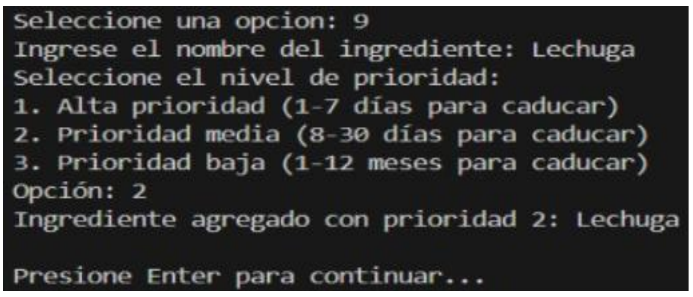
Agregar ingrediente

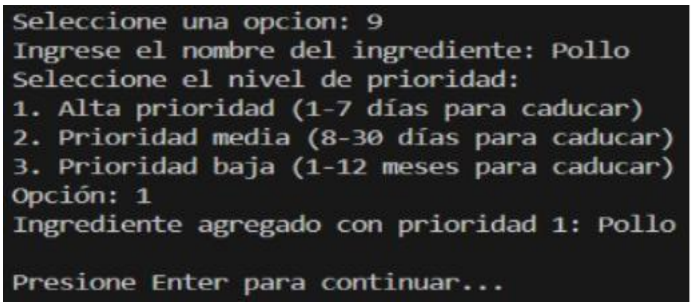




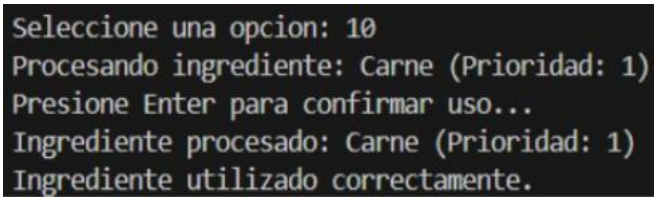




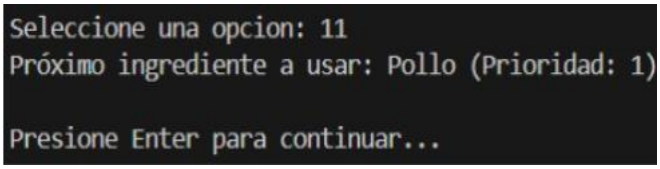




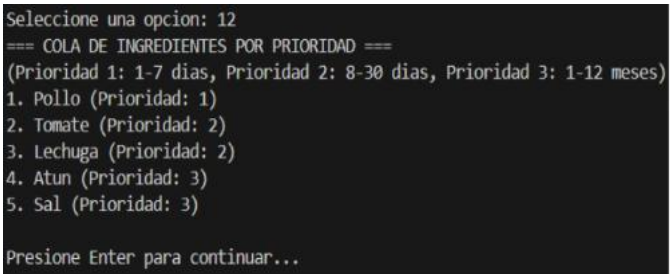
Usar ingrediente prioritario



Ver próximo ingrediente a usar

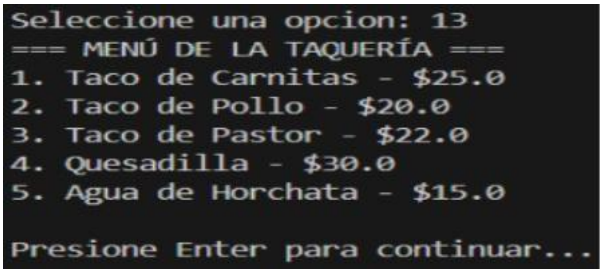


Mostrar cola completa de ingredientes

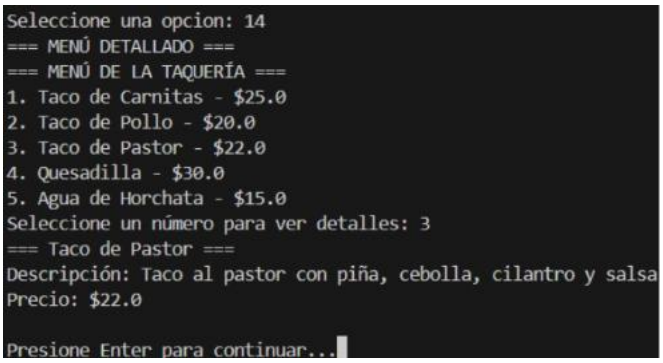


7. MENÚ DE LA TAQUERÍA (Lista Enlazada)

Mostrar menú

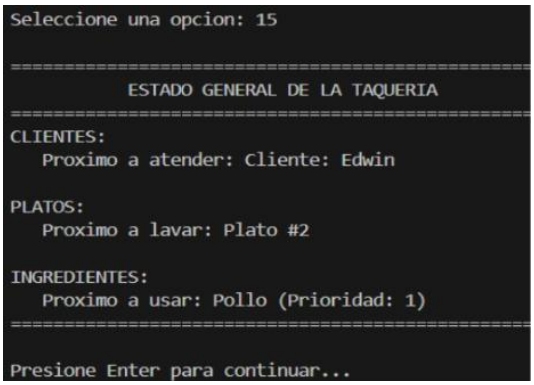


Ver detalles de platillo

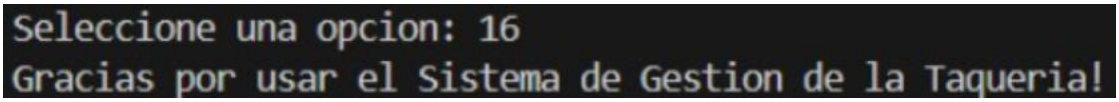


8. REPORTES Y SALIDA

Estado general del sistema



Salir del sistema



9. Conclusión

Este proyecto permitió aplicar los conocimientos de estructuras de datos en un contexto realista. Cada estructura fue utilizada de manera coherente con su propósito: la cola para clientes en orden de llegada (FIFO), la pila para platos sucios (LIFO), la cola de prioridad para ingredientes según caducidad y la lista enlazada para gestionar el menú. Esto demostró la importancia de elegir la estructura correcta para cada situación, lo cual facilita la resolución de problemas en sistemas reales.