

# Proyecto Práctico: Sistema de Gestión de Pedidos en una Pizzería

## Objetivo del Proyecto

Desarrollar un sistema en Java para gestionar los pedidos de una pizzería, donde se aplicarán tres métodos de ordenamiento (inserción, Shellsort y Quicksort) para organizar y gestionar la lista de pedidos según diferentes criterios: tiempo de preparación, precio total, y nombre del cliente.

## Contexto del Caso Real

En una pizzería, la gestión eficiente de los pedidos es crucial para asegurar que los clientes reciban sus órdenes en tiempo y forma. Este sistema permitirá al encargado ordenar los pedidos según diferentes necesidades operativas, como priorizar pedidos por tiempo de preparación (para agilizar el trabajo en la cocina) o por precio total (para facilitar la contabilidad).

## Ejemplo de la estructura del Proyecto

```
/gestion-pizzeria
|
├── /src
|   ├── Main.java           # Punto de entrada del sistema
|   ├── Pedido.java         # Clase que representa un pedido
|   ├── Pizzeria.java       # Clase que gestiona los pedidos de la pizzería
|   ├── Ordenador.java      # Clase que implementa los algoritmos de ordenamiento
|   └── TiempoOrdenamiento.java # Clase para medir el tiempo de ejecución de los
algoritmos
|
└── README.md               # Documentación del proyecto
```

## Funcionalidades Principales

- Gestión de Pedidos:**
  - Agregar y Eliminar Pedidos:** Permitir al encargado agregar nuevos pedidos o eliminar pedidos completados del sistema.
  - Actualizar Información del Pedido:** Modificar detalles como el tiempo estimado de preparación o el precio total.
- Ordenamiento de Pedidos:**
  - Por Tiempo de Preparación:** Usar el algoritmo de inserción para ordenar los pedidos según el tiempo estimado de preparación.
  - Por Precio Total:** Implementar Shellsort para ordenar los pedidos según el precio total.
  - Por Nombre del Cliente:** Usar Quicksort para ordenar los pedidos alfabéticamente por el nombre del cliente.
- Captura de Tiempos de Ejecución:**
  - Implementar funciones para medir y mostrar los tiempos de ejecución de cada algoritmo de ordenamiento con diferentes tamaños de listas de pedidos (por ejemplo, 100, 1000, y 10000 pedidos).

## Ejemplo de Código: Ordenar Pedidos por Precio

```

public class Ordenador {

    public void shellsort(List<Pedido> pedidos) {
        int n = pedidos.size();
        for (int gap = n / 2; gap > 0; gap /= 2) {
            for (int i = gap; i < n; i++) {
                Pedido temp = pedidos.get(i);
                int j;
                for (j = i; j >= gap && pedidos.get(j - gap).getPrecioTotal() >
temp.getPrecioTotal(); j -= gap) {
                    pedidos.set(j, pedidos.get(j - gap));
                }
                pedidos.set(j, temp);
            }
        }
    }
}

```

## Demo de la Funcionalidad

- **Escenario:** El encargado de la pizzería necesita priorizar los pedidos según el tiempo de preparación para asegurarse de que los pedidos más rápidos se completen primero.
- **Acción:** Selecciona la opción de ordenar por tiempo de preparación.
- **Resultado:** El sistema organiza la lista de pedidos utilizando el algoritmo de inserción, y muestra el pedido que debe prepararse primero.

## Requisitos Técnicos

- Implementación en Java.
- Aplicación de tres algoritmos de ordenamiento.