

## Informe de las prácticas de experimentación y aplicación de los aprendizajes

### 1. Datos Informativos:

Facultad:	CIENCIAS ADMINISTRATIVAS GESTIÓN EMPRESARIAL E INFORMÁTICA
Carrera:	Software
Asignatura:	Estructura de datos
Ciclo:	3ro "A"
Docente:	Ing. Danilo Barreno
Título de la práctica:	Aplicación práctica de Pilas o Colas
No. de práctica:	1
Escenario o ambiente de aprendizaje de la practica	Laboratorio
No. de horas:	24
Fecha:	Marzo 2025
Estudiantes:	Mikaela Alejandra Paredes Villacis Punina Ruiz Angeles Xiomara Calderón Cueva Ariel Alejandro
Calificación	

## 2. Introducción:

Las estructuras de datos son vitales para la programación, permiten organizar y gestionar los datos con algoritmos probados y eficientes, entre estas estructuras tenemos las Pilas (LIFO) y las Colas (FIFO) que pueden ser implementadas con arreglos (poco práctico) o con listas, permitiendo realizar diferentes operaciones como push, pop. Estas estructuras son muy utilizadas en la actualidad en redes, base de datos, sistemas operativos entre otros. **(Gonzales, 2024)**

## 3. Objetivo de la práctica:

Simular el funcionamiento de algoritmos que usen pilas y/o colas.

## 4. Descripción del desarrollo de la práctica:

Se realizó una simulación del historial de atención de clientes en un banco, utilizando una cola de prioridad. Se implementaron las operaciones de agregar y eliminar clientes de la cola, así como el registro del tiempo empleado en cada atención.

- **Atención de clientes:** Se agregaron clientes a la cola de prioridad y se registró el tiempo de atención. Navegación hacia el siguiente cliente: Se eliminaron clientes de la cola de prioridad y se actualizó el tiempo de atención.
- **Cierre del día:** Se vació la cola de prioridad y se calculó el tiempo total de atención.

## Implementación en Java

El sistema se compone de las siguientes clases principales:

- o **Banco:** Gestiona la cola de prioridad, permitiendo agregar clientes, atenderlos y calcular el tiempo total de atención al final del día.
- o **Cliente:** Representa a los clientes con atributos como nombre, prioridad y tiempo de atención.
- o **ColaPrioridad:** Implementa una cola de prioridad utilizando una lista enlazada, donde los clientes se insertan de acuerdo con su prioridad (1 = alta, 2 = media, 3 = baja).
- o **ListaEnlazada:** Estructura de datos utilizada por la cola de prioridad para gestionar los clientes de manera eficiente.
- o **UIMenu:** Proporciona una interfaz interactiva para que el usuario seleccione opciones a través de un menú.
- o **Main:** Punto de entrada del programa que inicia el menú y permite la interacción con el usuario.

**Clase: Cola con Prioridad**

```
class ColaPrioridad<T> {  
    private ListaEnlazada<T> lista;  
  
    public ColaPrioridad() {  
        this.lista = new ListaEnlazada<>();  
    }  
  
    // Agregar un elemento con prioridad  
    public void agregar(T elemento, int prioridad) {  
        lista.agregarOrdenado(elemento, prioridad);  
    }  
  
    // Eliminar y devolver el elemento de mayor prioridad  
    public T eliminar() {  
        return lista.eliminarPrimero();  
    }  
  
    // Verificar si la cola está vacía  
    public boolean estaVacia() {  
        return lista.estaVacia();  
    }  
  
    // Imprimir elementos en la cola:  
    public void imprimir(){  
        lista.imprimirElementos();  
    }  
}
```

## Clase: Lista Enlazada

```
class ListaEnlazada<T> {
    private class Nodo {
        T elemento;
        int prioridad;
        Nodo siguiente;

        public Nodo(T elemento, int prioridad) {
            this.elemento = elemento;
            this.prioridad = prioridad;
            this.siguiente = null;
        }
    }

    private Nodo cabeza;

    public ListaEnlazada() {
        this.cabeza = null;
    }

    // Agregar elemento en orden de prioridad
    public void agregarOrdenado(T elemento, int prioridad) {
        Nodo nuevo = new Nodo(elemento, prioridad);

        if (cabeza == null || cabeza.prioridad > prioridad) {
            nuevo.siguiente = cabeza;
            cabeza = nuevo;
        } else {
            Nodo actual = cabeza;
            while (actual.siguiente != null && actual.siguiente.prioridad <= prioridad) {
                actual = actual.siguiente;
            }
            nuevo.siguiente = actual.siguiente;
            actual.siguiente = nuevo;
        }
    }

    // Eliminar y devolver el primer elemento (mayor prioridad)
    public T eliminarPrimero() {
        if (cabeza == null) {
            return null;
        }
        T elemento = cabeza.elemento;
        cabeza = cabeza.siguiente;
        return elemento;
    }

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

    // Imprimir elementos:
    public void imprimirElementos() {
        if (cabeza == null) {
            System.out.println("[!] La lista esta vacia");
            return;
        }
        Nodo actual = cabeza;
        while (actual != null) {
            System.out.println(actual.elemento);
            actual = actual.siguiente;
        }
    }
}
```

## Clase: Banco

```
public class Banco {
    private ColaPrioridad<Cliente> cola;
    private int tiempoTotalAtencion = 0;

    public Banco() {
        this.cola = new ColaPrioridad<>();
    }

    // Agregar cliente a la cola
    public void agregarCliente(String nombre, int prioridad, int tiempoAtencion) {
        Cliente nuevoCliente = new Cliente(nombre, prioridad, tiempoAtencion);
        cola.agregar(nuevoCliente, prioridad);
        System.out.println(":: Cliente agregado: " + nuevoCliente);
    }

    // Atender al siguiente cliente
    public void atenderCliente() {
        Cliente atendido = cola.eliminar();
        if (atendido != null) {
            tiempoTotalAtencion += atendido.getTiempoAtencion();
            System.out.println(":: Atendiendo a: " + atendido);
        } else {
            System.out.println("[!] No hay clientes en espera.");
        }
    }

    // Cerrar el día
    public void cerrarDia() {
        System.out.println("[Cierre del día]: Atendiendo a todos los clientes restantes...");
        while (!cola.estaVacia()) {
            atenderCliente();
        }
        System.out.println(":: Tiempo total de atención: " + tiempoTotalAtencion + " min");
    }

    // Imprimir cola de Clientes:
    public void imprimirColaClientes() {
        cola.imprimir();
    }
}
```

## Clase: Cliente

```
public class Cliente {  
    private String nombre;  
    private int prioridad;  
    private int tiempoAtencion;  
  
    public Cliente(String nombre, int prioridad, int tiempoAtencion) {  
        this.nombre = nombre;  
        this.prioridad = prioridad;  
        this.tiempoAtencion = tiempoAtencion;  
    }  
  
    public int getTiempoAtencion() {  
        return tiempoAtencion;  
    }  
  
    @Override  
    public String toString() {  
        return ":: Cliente: " + nombre + " | Prioridad: " + prioridad + " | Tiempo Atención: " + tiempoAtencion + "  
min";  
    }  
}
```

## 5. Metodología:

La metodología para el desarrollo del Proyecto de Java para la gestión de turnos de un banco con cola de prioridad se basará en un enfoque ágil utilizando Scrum, el desarrollo se realizará con reuniones diarias para monitorear el progreso, se implementarán pruebas unitarias, de integración y de usuario para asegurar la calidad del sistema, posteriormente, se procederá al despliegue del sistema en producción, junto con la capacitación del personal del banco.

Finalmente, se establecerán mecanismos para la recopilación de retroalimentación y se realizarán revisiones periódicas para evaluar el rendimiento del sistema, asegurando así su mejora continua y adaptación a las necesidades de los usuarios (Cansino, 2025).

## 6. Conclusiones:

- La implementación de la simulación del historial de atención de clientes utilizando una cola de prioridad permitió poner en práctica los conceptos teóricos de estructuras de datos como las colas, demostrando su eficiencia para gestionar la atención de clientes según su nivel de prioridad.
- El proyecto permitió desarrollar habilidades en el diseño e implementación de soluciones informáticas basadas en estructuras de datos, lo cual es una competencia clave para abordar problemas del mundo real.

## 7. Recomendaciones:

- Considerar la implementación de funcionalidades adicionales, como la posibilidad de asignar prioridades dinámicas a los clientes en función de criterios específicos del banco, para mejorar la eficiencia del sistema.
- Realizar pruebas exhaustivas con diferentes escenarios de carga y flujo de clientes, a fin de evaluar la escalabilidad y robustez del sistema, y garantizar su adecuado funcionamiento en situaciones reales.

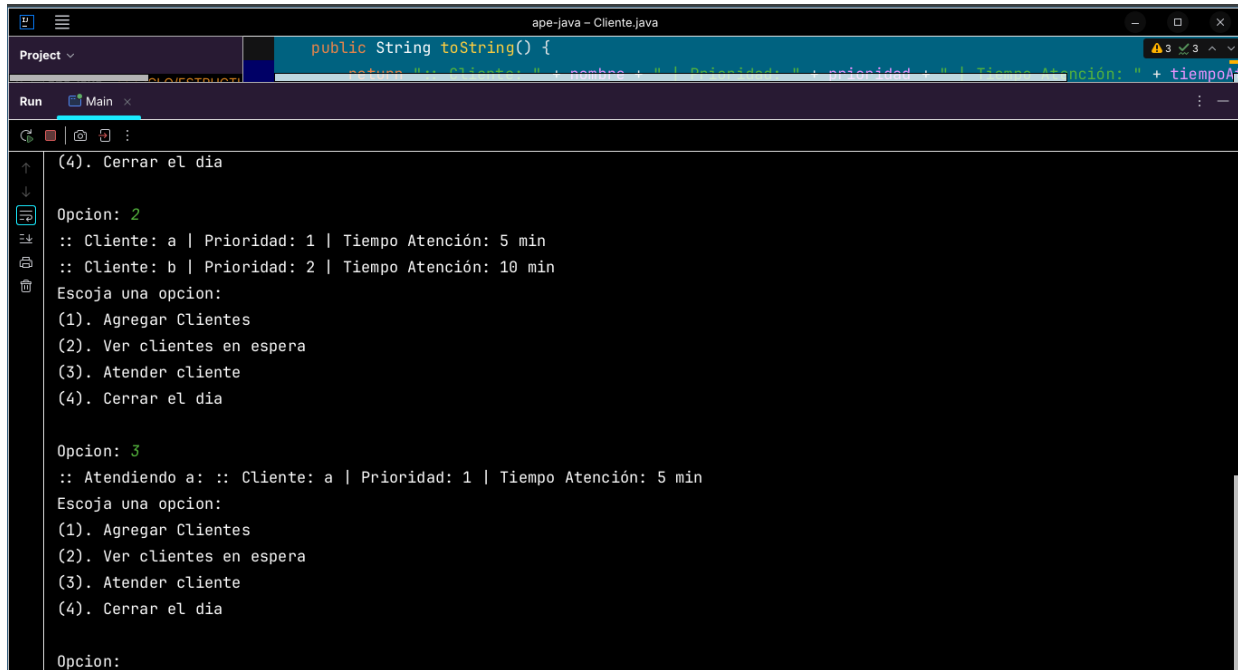
## 8. Bibliografía:

- Cansino, C. (13 de Febrero de 2025). Obtenido de ¿Cómo funcionan y para qué sirven las colas en programación?: <https://www.cesarcancino.com/>
- Gonzales, T. (05 de Junio de 2024). Obtenido de Uso de Colas y Pilas en programacion: <https://academiasanroque.com/uso-de-colas-y-pilas-en-programacion/>



## 9. Anexos:

### Capturas del programa de terminal



```

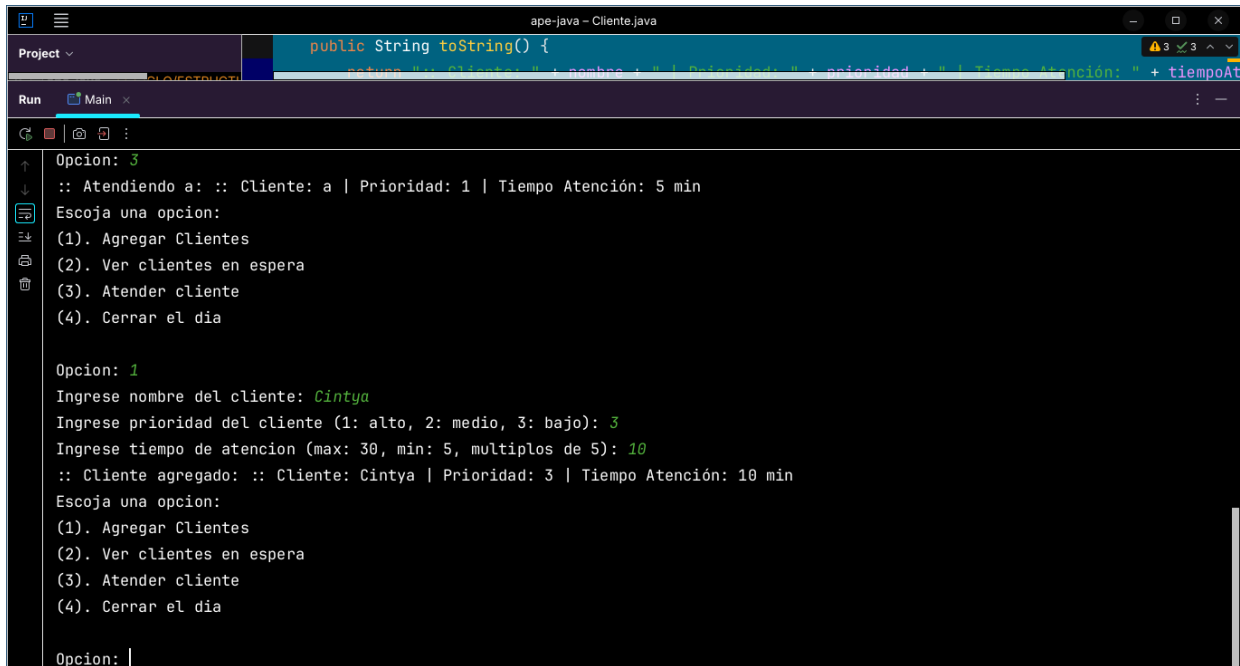
ape-java - Cliente.java
Project: ...
Run: Main
(4). Cerrar el dia

Opcion: 2
:: Cliente: a | Prioridad: 1 | Tiempo Atención: 5 min
:: Cliente: b | Prioridad: 2 | Tiempo Atención: 10 min
Escoja una opcion:
(1). Agregar Clientes
(2). Ver clientes en espera
(3). Atender cliente
(4). Cerrar el dia

Opcion: 3
:: Atendiendo a: :: Cliente: a | Prioridad: 1 | Tiempo Atención: 5 min
Escoja una opcion:
(1). Agregar Clientes
(2). Ver clientes en espera
(3). Atender cliente
(4). Cerrar el dia

Opcion:

```



```

ape-java - Cliente.java
Project: ...
Run: Main
Opcion: 3
:: Atendiendo a: :: Cliente: a | Prioridad: 1 | Tiempo Atención: 5 min
Escoja una opcion:
(1). Agregar Clientes
(2). Ver clientes en espera
(3). Atender cliente
(4). Cerrar el dia

Opcion: 1
Ingrese nombre del cliente: Cintya
Ingrese prioridad del cliente (1: alto, 2: medio, 3: bajo): 3
Ingrese tiempo de atencion (max: 30, min: 5, multiples de 5): 10
:: Cliente agregado: :: Cliente: Cintya | Prioridad: 3 | Tiempo Atención: 10 min
Escoja una opcion:
(1). Agregar Clientes
(2). Ver clientes en espera
(3). Atender cliente
(4). Cerrar el dia

Opcion: 1

```