Proyecto Final (4 semanas)
Sistema de Gestión de Aerolínea por Consola en Java

Uso de Pilas, Colas, Listas y Recursividad Lineal

Asignatura: Estructura de datos
Docente: Diego Alejandro Franco
Fecha de inicio: 19.-11-2025
Fecha de entrega: 10-12-2025

1. Descripción General

En este proyecto desarrollarás una aplicación de consola en Java que simula la gestión básica de una aerolínea. El objetivo es integrar los conceptos de pilas, colas, listas y recursividad lineal trabajando sobre un caso práctico. No se utilizará interfaz gráfica; toda la interacción será mediante menús de texto en consola.

La aplicación debe permitir, como mínimo:

- Registrar y consultar vuelos.
- Registrar pasajeros y asociarlos a vuelos.
- Manejar listas de espera de pasajeros usando colas.
- Llevar un historial de operaciones y permitir deshacer la última acción usando pilas.
- Realizar búsquedas y conteos usando métodos recursivos lineales.

2. Objetivos de Aprendizaje

Al finalizar el proyecto, el estudiante será capaz de:

- Modelar un problema real (aerolínea) usando estructuras de datos: listas, pilas y colas.
- Implementar métodos recursivos lineales simples para búsqueda, conteo y recorrido de estructuras.
- Diseñar un programa modular con clases y métodos bien definidos.
- Trabajar con menús en consola, validación básica de datos y documentación del código.
- Comprender el uso de la pila de llamadas en la recursividad y el comportamiento de pilas y colas en memoria.

3. Alcance Funcional Mínimo

3.1. Clases Mínimas Requeridas

El sistema debe manejar, como mínimo, las siguientes clases en Java:

- Clase Pasajero: atributos sugeridos: id, nombre, documento, nacionalidad.
- Clase Vuelo: atributos sugeridos: codigoVuelo, origen, destino, capacidadMaxima, listaPasajeros, listaEspera.
- Clase opcional Reserva: puede representar la relación entre un Pasajero y un Vuelo, si el estudiante lo considera útil.

Nota: Se pueden agregar más atributos (como fecha, hora, precio, etc.), pero no es obligatorio. Lo importante es que las estructuras de datos y las operaciones pedidas estén claramente implementadas.

3.2. Menú Principal en Consola

La aplicación debe ofrecer un menú principal en consola que permita acceder a las diferentes funcionalidades. El siguiente es un ejemplo de estructura de menú (puede ser ajustado, pero las funcionalidades deben estar presentes):

1. Gestionar Vuelos

   1.1 Registrar nuevo vuelo

   1.2 Listar todos los vuelos

   1.3 Buscar vuelo por código (usar recursividad sobre la lista de vuelos)

2. Gestionar Pasajeros y Reservas

   2.1 Registrar pasajero en un vuelo (crear reserva)

   2.2 Cancelar reserva de pasajero en un vuelo

   2.3 Mostrar lista de pasajeros de un vuelo

   2.4 Mostrar lista de espera de un vuelo

3. Listas de Espera (Colas)

   3.1 Agregar pasajero a lista de espera si el vuelo está lleno

   3.2 Procesar lista de espera cuando se libera un cupo

4. Historial y Deshacer (Pilas)

   4.1 Registrar operaciones en una pila (reservas, cancelaciones)

   4.2 Deshacer la última operación (undo)

5. Reportes (usando Recursividad)

   5.1 Contar recursivamente el total de pasajeros de un vuelo

   5.2 Contar recursivamente cuántos pasajeros van a un destino dado

   5.3 Buscar recursivamente un pasajero por documento en todos los vuelos

0. Salir

Importante: El menú puede tener un diseño distinto, pero todas las funcionalidades descritas deben estar implementadas de alguna forma clara.

4. Requisitos de Uso de Estructuras de Datos

4.1. Listas (ArrayList o LinkedList)

Se deben usar listas para:

- Mantener la lista global de vuelos (por ejemplo: List<Vuelo>).
- Mantener la lista de pasajeros que tienen reserva confirmada en cada vuelo.

Al menos dos operaciones sobre estas listas deben implementarse mediante recursividad lineal, por ejemplo:

- Búsqueda recursiva de un vuelo por código en la lista de vuelos.
- Conteo recursivo de pasajeros en la lista de un vuelo.
- Conteo recursivo de pasajeros que viajan a un cierto destino entre todos los vuelos.

4.2. Colas (Queue)

Se debe utilizar una cola (por ejemplo, Queue<Pasajero> implementada con LinkedList) para gestionar la lista de espera de cada vuelo.

- Si un vuelo está lleno y se intenta agregar un nuevo pasajero, este debe ir a la lista de espera (cola).
- Cuando se cancele una reserva y se libere un cupo, se debe ofrecer ese cupo al primer pasajero en la cola de espera (dequeue).

4.3. Pilas (Stack o Deque)

Se debe utilizar una pila para llevar un historial de operaciones importantes (por ejemplo, altas y bajas de reservas), de forma que el sistema pueda deshacer la última operación realizada.

- Registrar en la pila operaciones como: "RESERVA: pasajero X en vuelo Y" o "CANCELACIÓN: pasajero Z en vuelo W".
- Implementar una opción de menú "Deshacer última operación" que lea el tope de la pila, interprete la operación y la revierta si es posible.
- Luego de deshacer, esa operación debe eliminarse de la pila.

5. Requisitos Mínimos de Recursividad

Se deben implementar al menos cuatro métodos recursivos lineales relacionados directamente con la lógica de la aerolínea. Ejemplos de métodos que pueden ser recursivos (lineales, una sola llamada recursiva por nivel):

- Búsqueda recursiva de un vuelo por código en la lista de vuelos.
- Conteo recursivo de pasajeros en un vuelo.
- Conteo recursivo de pasajeros que viajan a un determinado destino en todos los vuelos.

- Búsqueda recursiva de un pasajero por documento recorriendo todos los vuelos y sus listas de pasajeros.

Cada método recursivo debe tener:

- Un caso base bien definido (condición de parada).
- Un caso recursivo que reduzca el problema y avance hacia el caso base.
- Comentarios breves en el código explicando el caso base y el caso recursivo.

6. Plan de Trabajo Sugerido (4 Semanas)

Semana 1: Análisis y Diseño

Actividades sugeridas:

- Definir con el equipo qué funciones exactas tendrá el sistema (alcance).
- Identificar las clases principales: Pasajero, Vuelo, (opcional) Reserva.
- Diseñar un diagrama simple de clases (papel o herramienta digital).
- Definir qué estructuras de datos (lista, cola, pila) se usarán en cada parte.
- Definir qué operaciones se implementarán de forma recursiva.

Entregable sugerido de la semana 1: breve documento (1-2 páginas) con el diseño y decisiones.

Semana 2: Implementación Base (Listas y Menú)

Actividades sugeridas:

- Implementar las clases Pasajero y Vuelo con sus atributos y métodos básicos (constructores, getters, toString, etc.).
- Crear la estructura principal de datos: lista de vuelos.
- Implementar el menú principal en consola (aunque inicialmente algunas opciones estén vacías).
- Implementar las operaciones básicas: registrar vuelos, listar vuelos, registrar pasajeros en vuelos (si hay cupo), mostrar pasajeros de un vuelo.

Al final de la semana 2 el sistema debería permitir, al menos, crear vuelos y asignar pasajeros sin listas de espera ni undo.

Semana 3: Pilas, Colas y Recursividad

Actividades sugeridas:

- Agregar la cola de espera a cada vuelo (Queue<Pasajero>).
- Agregar la pila de operaciones para el historial (Stack<String> o Deque<String>).
- Implementar la lógica: si un vuelo está lleno, agregar el pasajero a la lista de espera.

- Implementar la opción de cancelar reserva y mover el primer pasajero de la lista de espera a la lista principal cuando haya cupo.
- Implementar al menos cuatro métodos recursivos lineales y conectarlos con opciones del menú (búsquedas, conteos, etc.).

Semana 4: Pruebas, Mejora y Documentación

Actividades sugeridas:

- Probar el sistema con varios vuelos y pasajeros (incluyendo casos en los que se llenen los vuelos).
- Probar la lista de espera y el deshacer operaciones con la pila.
- Mejorar mensajes por consola y validaciones básicas (por ejemplo, evitar índices fuera de rango).
- Documentar el código con comentarios claros.
- Preparar un documento final que explique el funcionamiento del sistema y cómo se usaron pilas, colas, listas y recursividad.

7. Criterios de Evaluación (Sugeridos)

La siguiente rúbrica es una guía para la evaluación del proyecto:

- Uso correcto de estructuras de datos (30%): listas para vuelos/pasajeros, colas para listas de espera, pilas para undo.
- Implementación de recursividad (25%): al menos cuatro métodos recursivos lineales bien justificados y comentados.
- Diseño y modularidad del código (20%): clases con responsabilidades claras, métodos bien definidos.
- Funcionalidad general (15%): el menú funciona, las operaciones principales están implementadas y son coherentes.
- Documentación y presentación (10%): comentarios en el código, documento explicativo, claridad en la entrega.

8. Recomendaciones para los Estudiantes

- Trabajen paso a paso: no intenten programar todo el sistema de una vez.
- Compilen y prueben frecuentemente cada nueva funcionalidad que agreguen.
- Agreguen mensajes por consola que ayuden a entender qué está pasando internamente.
- Comenten sus métodos recursivos explicando el caso base y el caso recursivo.
- Si algo deja de funcionar, revisen primero los últimos cambios realizados.
- Usen nombres de variables y métodos descriptivos (por ejemplo, registrarPasajeroEnVuelo en lugar de rpv).