



UNIVERSIDAD AUTÓNOMA DE ZACATECAS

Practica: Colas

Estudiante:

José Francisco Hurtado Muro

Profesor:

Dr. Aldonso Becerra Sánchez

April 13, 2025

Tabla de Contenidos

1	Introducción	2
2	Actividades:	3
2.1	Actividad inicial:	3
2.2	Actividad 1:	3
2.3	Actividad 2:	3
2.3.1	TDA - ColaPrioridad	5
2.4	actividad 3:	7
2.4.1	Control de pistas de Aeropuerto	7
3	Código Agregado - UML	8
4	Pre-evaluación del Alumno	9
5	Conclusión	9
6	Referencias:	9

1 Introducción

"En muchos ambientes es requerido realizar procesos donde se requiere ir almacenando elementos e irlos agregando a una cola de espera, la cual será procesada conforme llegaron."

2 Actividades:

Seleccionando dos escenarios de la práctica 1, genere los diagramas de flujo de datos de ellos, considerando mínimo los niveles 0 y 1 (del nivel 2 en adelante son a consideración de cada escenario, siendo optativo en muchos casos).

2.1 Actividad inicial:

Generar el reporte en formato IDC

2.2 Actividad 1:

Primero genere la Introducción

2.3 Actividad 2:

Defina un programa propietario que implemente la funcionalidad de una cola de prioridad (debe analizarse si se hará herencia, por ejemplo, o no) usando la elección de su preferencia de acuerdo a las formas de implementar vistas en clase (o alguna variante de ahí).

Las colas de prioridad modifican el orden en el que los procesos son atendidos, esto debido a esquemas de orden definido por importancia, relevancia o necesidad. Por ejemplo:

Colas de prioridad					
datos		D	H	T	J
	0	1	2	3	4
datos		D	H	T	J
prioridad		9	10	7	12
	0	1	2	3	4

Figure 1: ejemplo 1

En el esquema de arriba el renglón de prioridad indica (en este caso ASC) que los elementos que tienen mayor prioridad son atendidos primero, independientemente de si llegaron antes o después de otros. En este caso el primero en ser atendido es J, con prioridad 12, después H con prioridad 10, y así sucesivamente. La forma de implementar es libre al programador.

Posibles formas de como implementarlas

* Arreglos paralelos

datos		D	G	H	J
prioridad		9	7	10	12
	0	1	2	3	4

* Tener una cola para cada tipo de prioridad

prioridad	datos
1	D F G
2	X H
6	
9	S Ñ P

* Con objetos complejos (contenido, prioridad)

(S, 8)	(Ñ, 7)	(D, 10)		
0	1	2		

Figure 2: ejemplo 2

2.3.1 TDA - ColaPrioridad

Teniendo en cuenta las características de este tipo de cola, se dejó que los métodos básicos de una cola actuaran normal como lo es: **poner()**, **lleno()** y **vacía()**, estos métodos funcionan exactamente igual a lo que ya hemos visto, en el método **poner()** no importa como los vaya agregando, pero lo que se hizo es ponerlos de manera consecutiva, o sea que cada elemento que se vaya agregando se continuación del otro, pero para este punto aun no importa la prioridad de cada elemento, eso es hasta el método **quitar()**, ya que en este lo que tiene que hacer es quitar el elemento con mas alta prioridad, respetando el orden si es que hay mas de uno que tenga la misma prioridad, como podemos ver en la figura 3.

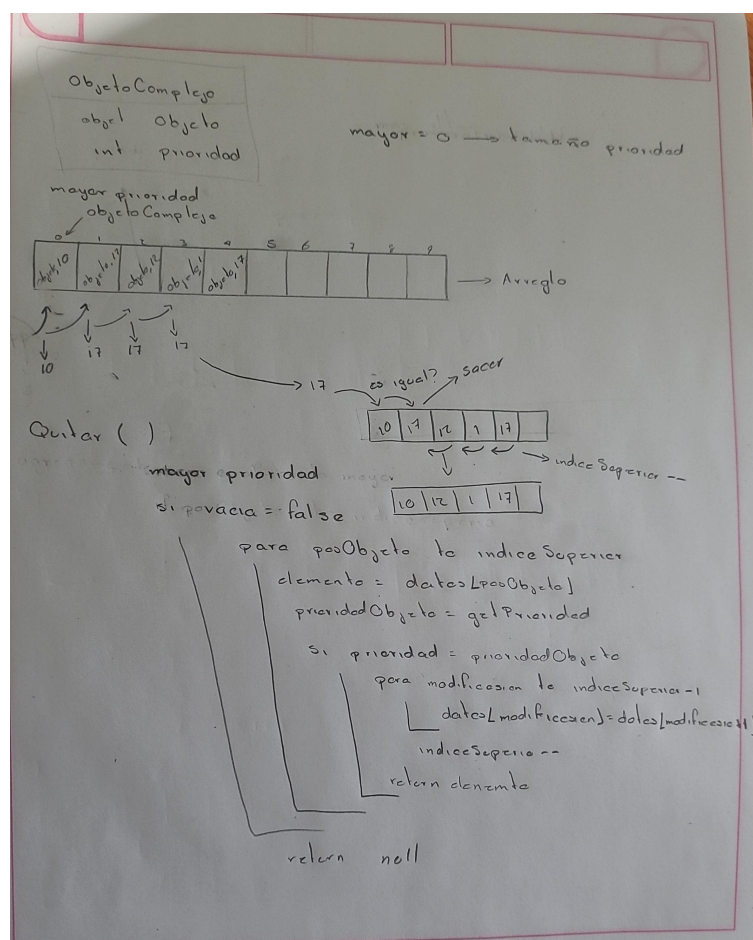


Figure 3: Analisis: quitar() en ColaPrioridad

Explicación

Para este caso de quitar se planteó que el arreglo de **datos[]** recibiera solo elementos de tipo **ObjetosComplejos**, que tendrían dos elementos, un objeto cualquiera y una prioridad en número entero, donde para poder quitar un elemento tenemos que buscar cual es la prioridad mas alta, una vez obtenida, volvemos a buscar el primer elemento que tenga esa prioridad y lo sacamos del arreglo, recorriendo una posición los elementos que se encontraban después de este elemento, y es así como regresamos el valor quitado como podemos ver en la figura 4.

```

¿La cola está vacía? true

Cola después de agregar elementos:
dato: Elemento1, prioridad: 2
dato: Elemento2, prioridad: 5
dato: Elemento3, prioridad: 5
dato: Elemento4, prioridad: 4

¿La cola está llena? false

Elemento quitado: EstructuraDatos.EDLineal.colaPrioridad.ObjetosComplejos@eed1f14

Cola después de quitar un elemento:
dato: Elemento1, prioridad: 2
dato: Elemento3, prioridad: 5
dato: Elemento4, prioridad: 4

Cola después de agregar otro elemento:
dato: Elemento1, prioridad: 2
dato: Elemento3, prioridad: 5
dato: Elemento4, prioridad: 4
dato: Elemento5, prioridad: 3

Elemento quitado: EstructuraDatos.EDLineal.colaPrioridad.ObjetosComplejos@4c873330
Cola actual:
dato: Elemento1, prioridad: 2
dato: Elemento4, prioridad: 4
dato: Elemento5, prioridad: 3

Elemento quitado: EstructuraDatos.EDLineal.colaPrioridad.ObjetosComplejos@119d7047
Cola actual:
dato: Elemento1, prioridad: 2
dato: Elemento5, prioridad: 3

Elemento quitado: EstructuraDatos.EDLineal.colaPrioridad.ObjetosComplejos@776ec8df
Cola actual:
dato: Elemento1, prioridad: 2

Elemento quitado: EstructuraDatos.EDLineal.colaPrioridad.ObjetosComplejos@4eec7777
Cola actual:

¿La cola está vacía? true
PS C:\Users\Josef\OneDrive\Documentos\Mis_docs\4toSemestre\Estructura-Datos>
  
```

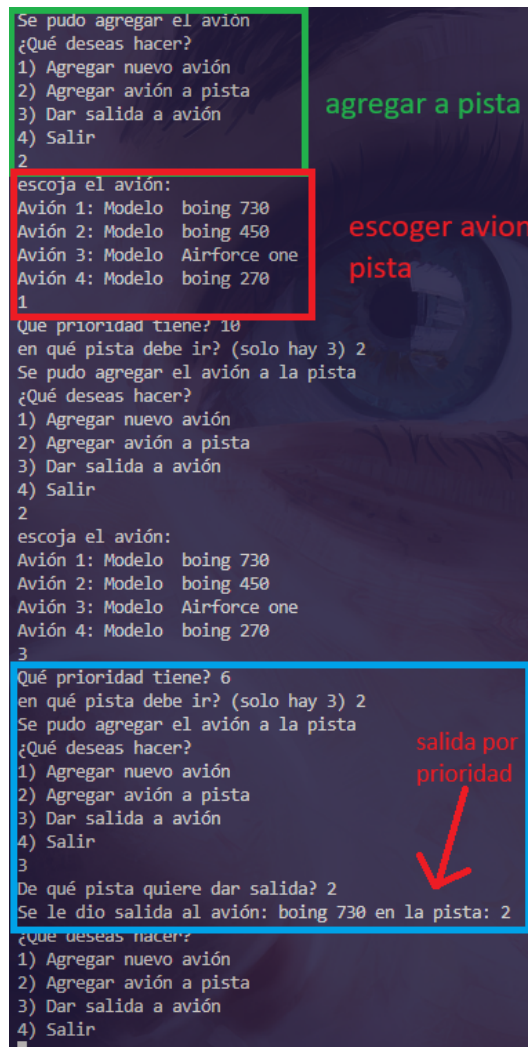
Figure 4: Funcionamiento: quitar() en ColaPrioridad

2.4 actividad 3:

Realice una simulación de una cola de: aviones que tienen que despegar de una misma pista compartida (asuma que hay varias pistas). La prioridad puede ser el tiempo que lleva esperando el avión desde que tenía planificada la salida. No tiene porqué comportarse como una cola tradicional ya que puede haber vuelos con mayor prioridad.

2.4.1 Control de pistas de Aeropuerto

Para lograr el propósito de este programa se involucro 3 clases importantes, **Avión**, que es la entidad que almacena los datos de un avión (**modelo, capacidad y aerolínea**), otra clase que se tomo mucho en cuenta una clase controladora **GestorAeropuerto**, esta clase se encarga de tener la lógica de como se va estar comportando el sistema según lo que el usuario le mande, en esta clase podemos encontrar la composición de elementos como métodos de **Entrada y Salida**, así como objetos que constituyen al funcionamiento de la lógica como lo es la **ColaPrioridad**, por ultimo tenemos la clase **Main**, que esta solo se encarga instanciar al gestor y darle paso libre a que la lógica del programa se ejecute como el usuario tenga planeado usarlo como podemos ver en la figura 5.



```

Se pudo agregar el avión
¿Qué deseas hacer?
1) Agregar nuevo avión
2) Agregar avión a pista
3) Dar salida a avión
4) Salir
2
escoja el avión:
Avión 1: Modelo boing 730
Avión 2: Modelo boing 450
Avión 3: Modelo Airforce one
Avión 4: Modelo boing 270
1
Que prioridad tiene? 10
en qué pista debe ir? (solo hay 3) 2
Se pudo agregar el avión a la pista
¿Qué deseas hacer?
1) Agregar nuevo avión
2) Agregar avión a pista
3) Dar salida a avión
4) Salir
2
escoja el avión:
Avión 1: Modelo boing 730
Avión 2: Modelo boing 450
Avión 3: Modelo Airforce one
Avión 4: Modelo boing 270
3
Que prioridad tiene? 6
en qué pista debe ir? (solo hay 3) 2
Se pudo agregar el avión a la pista
¿Qué deseas hacer?
1) Agregar nuevo avión
2) Agregar avión a pista
3) Dar salida a avión
4) Salir
3
De qué pista quiere dar salida? 2
Se le dio salida al avión: boing 730 en la pista: 2
¿Qué deseas hacer?
1) Agregar nuevo avión
2) Agregar avión a pista
3) Dar salida a avión
4) Salir

```

Figure 5: Funcionamiento: programa Aeropuerto

3 Código Agregado - UML

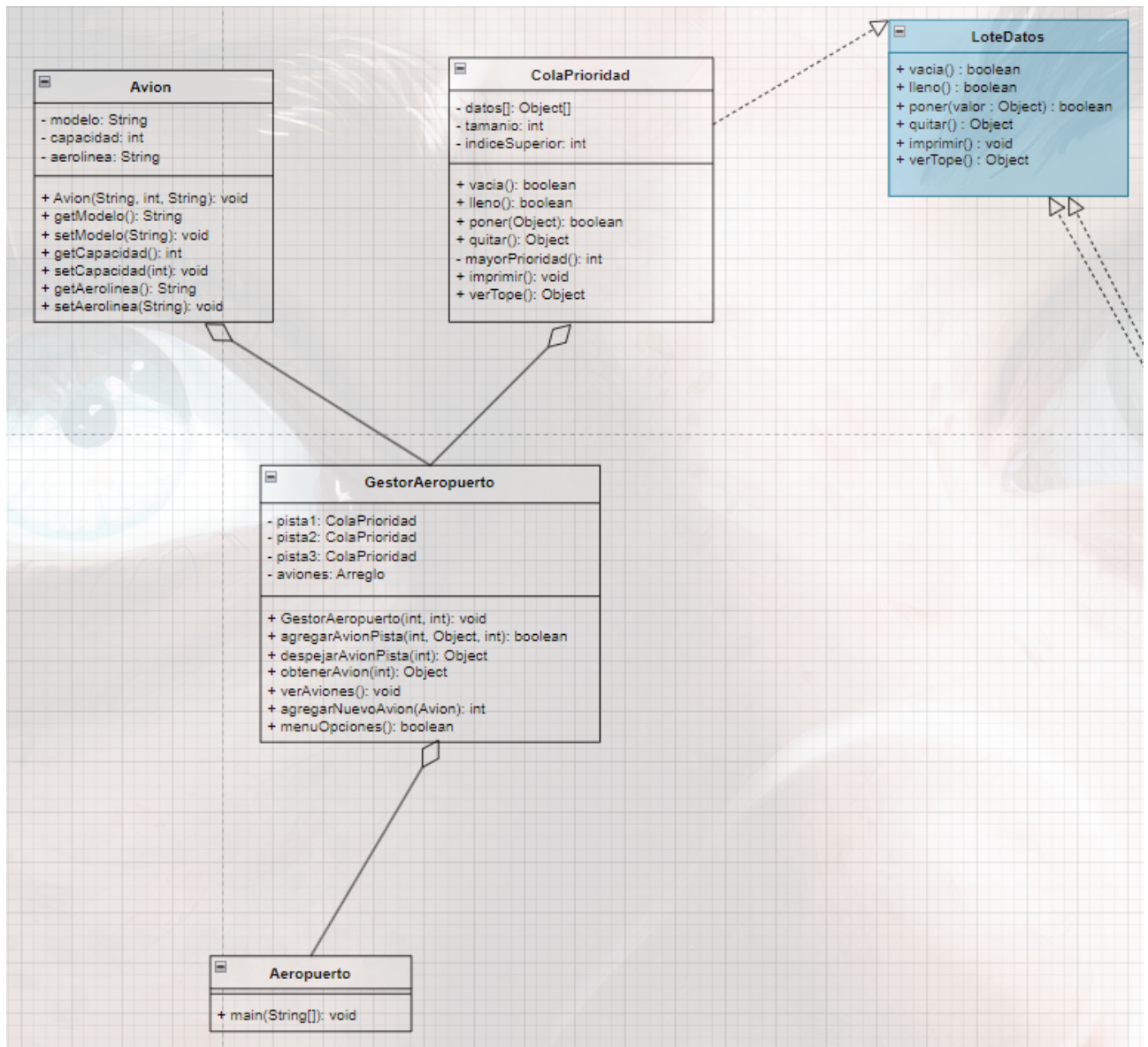


Figure 6: UML

4 Pre-evaluación del Alumno

Criterio	Evaluación
Cumple con la funcionalidad solicitada	Sí
Dispone de código auto-documentado	Sí
Dispone de código documentado a nivel de clase y método	Sí
Dispone de indentación correcta	Sí
Cumple la POO	Sí
Dispone de una forma fácil de utilizar el programa para el usuario	Sí
Dispone de un reporte con formato IDC	Sí
La información del reporte está libre de errores de ortografía	Sí
Se entregó en tiempo y forma la práctica	No
Incluye el código agregado en formato UML	Sí
Incluye las capturas de pantalla del programa funcionando	Sí
La práctica está totalmente realizada (especifique el porcentaje completado)	100%

Table 1: Evaluación de la práctica

5 Conclusión

Se identifico que el uso de colas de prioridad pueden ser utilizadas para poder priorizar procesos por encima de otros aun que otros hayan llegado antes que una de alta prioridad.

6 Referencias:

- Cairo, Osvaldo; Guardati, Silvia. *Estructura de Datos, Tercera Edición*. McGraw-Hill, México, Tercera Edición, 2006.
- Mark Allen Weiss. *Estructura de datos en Java*. Ed. Addison Wesley.
- Joyanes Aguilar, Luis. *Fundamentos de Programación. Algoritmos y Estructuras de Datos*. Tercera Edición, 2003. McGraw-Hill.