

## Objetivos

### Unidad 4: Algoritmos de Ordenamiento y Búsqueda

Al finalizar esta unidad, el estudiante estará en capacidad de:

- OE4.1 Implementar algoritmos clásicos de ordenamiento de datos en estructuras de datos lineales y aplicarlos en la solución de un problema.
- OE4.2 Implementar algoritmos clásicos de búsqueda de información en estructuras de datos lineales y aplicarlos en la solución de un problema.
- OE4.3 Reconocer la diferencia entre orden natural y orden parcial de los objetos por medio de la descripción de utilidades de las interfaces Comparable y Comparator.
- OE4.4 Calcular el tiempo de ejecución de un algoritmo por medio de las operaciones de tiempo del sistema
- OE4.5 Implementar métodos que permitan generar muestras con datos aleatorios.

## Preparación

- Lea cuidadosamente el enunciado, la documentación suministrada y cada uno de los puntos que debe desarrollar antes de empezar su desarrollo. Pregunte a su profesor cualquier duda respecto al enunciado o a los requerimientos funcionales que debe desarrollar.
- Lea cuidadosamente la rúbrica del laboratorio de la unidad 4 (ver rúbrica).
- El trabajo debe ser realizado individualmente.
- El trabajo será entregado en la fecha y hora establecida en Moodle.

## Enunciado

### PANTALLA DE AEROPUERTO



TIME	AIRLINE	FLIGHT	TO	TERM.	CHK-IN	GATE	EXPE.	REMARKS
14:45	Avianca	AV 080	CARACAS	1		7		GO TO GATE
15:25	Avianca	AV 284	NEW YORK	1		5		GO TO GATE
15:50	Avianca	MM 110	CURACAO	1		4		GO TO GATE
15:50	Avianca	MM 113	GUAYAQUIL	1		7		GO TO GATE
16:10	TACA	LR 690	SAN JOSE C.R.	1	76-80			
16:16	Avianca	CM 302	PANAMA	1	57-60			
16:50	Avianca	AV 067	QUITO	1		1		
17:10	TACA	TA 133	LIMA	1	71-75			
17:58	IB	6740	MADRID	1	23-30		18:30	DELAYED
18:00	Avianca	P5 622	PANAMA	1	53-56			

CERRADO: LETICIA

Imagen 1. Pantalla de un terminal aéreo

Con la aparición de las aerolíneas de bajo costo son cada vez más las personas que viajan en avión en el país. Durante 2018 viajaron 37 millones de personas en avión en Colombia, un incremento del 6,1 respecto de 2017<sup>1</sup>. Como parte de toda la operación logística para que tantas personas puedan movilizarse a través de las terminales aéreas (aeropuertos), es fundamental la información que en ellos se da de los vuelos a través de todas las pantallas dispuestas para tal fin. En la imagen 1 se muestra la pantalla de un terminal aéreo.

Usted ha sido contratado para simular una pantalla similar que permita mostrar la información de los vuelos pero con hora no militar, pues las personas se han quejado que no entienden muy bien el formato de hora militar. Por tanto usted debe mostrar la hora en formato AM/PM.

Para la simulación su programa debe generar aleatoriamente un listado de vuelos en diferentes fechas, horarios, diferentes aerolíneas, diferentes números de vuelo (éste debe ser único), ciudades destino y puertos de embarque. El orden por defecto en que los vuelos son mostrados es la hora de salida de menor a mayor.

<sup>1</sup> <https://www.elcolombiano.com/colombia/cuantas-personas-viajan-en-avion-en-colombia-DH10062818>

Una mejora que introducirá su desarrollo de la nueva pantalla de información de vuelos, es que será interactiva. Hasta ahora, las pantallas solo despliegan información pero no permiten que el usuario interactúe con ella, busque su vuelo o reordene los vuelos de acuerdo a diferentes criterios. Su programa permitirá esta interacción y será posible porque ahora las pantallas estarán físicamente al alcance de los usuarios.

Ya que es una simulación, el programa debe permitir cada vez que el usuario lo desee, generar aleatoriamente una nueva lista de vuelos. Debe tener la posibilidad de decidir cuántos vuelos se van a generar, aunque en pantalla el número será limitado, por lo que la vista podrá ser paginada, es decir, el programa despliega los primeros  $n$  vuelos ( $n$  es la cantidad máxima de vuelos a desplegar en pantalla, pueden ser 20, por ejemplo), y tendrá la posibilidad a través de algún control (botones por ejemplo) de mostrar los siguientes  $n$  vuelos y así hasta llegar a los últimos. También debería tener la posibilidad de mostrar los  $n$  vuelos anteriores. Es decir, la navegación entre páginas de  $n$  vuelos podrá ser hacia adelante y hacia atrás.

Por defecto, el ordenamiento de los vuelos es por fecha y hora, pero la pantalla ofrecerá al usuario la posibilidad de ordenar los vuelos por cualquiera de los otros criterios. También ofrecerá la posibilidad de buscar un vuelo por cualquiera de los criterios, es decir, se pueden buscar vuelos por fecha, por hora, por número de vuelo, por ciudad, etc. Si hay más de un vuelo que concuerde con el criterio buscado, entonces se mostrará el primer vuelo encontrado.

Cada vez que se hace una búsqueda o un ordenamiento, el programa debe mostrar el tiempo que se tardó en hacer dicha operación.

En su programa usted deberá:

1. Implementar y utilizar los tres métodos de ordenamiento clásicos: burbuja, selección e inserción.
2. Implementar y utilizar las dos estrategias de búsqueda clásicas: secuencial y binaria.
3. Utilizar la interface Comparable.
4. Utilizar la interface Comparator.
5. Utilizar el método de ordenamiento de la clase Arrays utilizando:
  - a. Comparable.
  - b. Comparator.

Ya que en el programa hay que hacer diferentes ordenamientos y búsquedas, para cumplir con los requisitos anteriores, usted utilizará un algoritmo de ordenamiento en un caso, otro algoritmo en otro caso, el método de ordenamiento de Arrays con Comparable en otro caso y el ordenamiento de Arrays con Comparator en otro caso. Lo mismo para las búsquedas.

#### **Entregables.**

1. Requerimientos Funcionales.
2. Diagrama de clases de modelo y control de la interfaz (no generado automáticamente)
3. Implementación completa de todos los requerimientos en Java.
4. Tabla de trazabilidad de requerimientos vs métodos (tabla con una columna de los requerimientos, tal que, por cada requerimiento se indica en la columna siguiente todos los métodos que contribuyen a resolverlo).<sup>3</sup>

**Fecha de Entrega:** Martes 9 de Abril de 2019 a las 11:59 AM a través de Moodle. El laboratorio debe trabajarse y entregarse individualmente.

César Leonardo Canales Rivera

A00345026

## Requerimientos funcionales

<b>Nombre</b>	<b>R. # 1. Mostrar la información de los vuelos.</b>
<b>Resumen</b>	El programa muestra la información de n vuelos en pantalla y si son más de n vuelos, el programa da la posibilidad de paginar o de tener un número de páginas, cada una con n vuelos, y de moverse hacia adelante y hacia atrás de las páginas mediante los botones de la interfaz.
<b>Entradas</b>	
Ninguna.	
<b>Resultados</b>	
Información de los vuelos en pantalla.	

<b>Nombre</b>	<b>R. # 2. Generar aleatoriamente una lista de vuelos.</b>
<b>Resumen</b>	El usuario es capaz de decidir cuántos vuelos se van a generar y una vez lo haga, se generará un listado de vuelos en diferentes fechas, horarios, diferentes aerolíneas, diferentes números de vuelo (éste es único), ciudades destino y puertas de embarque.
<b>Entradas</b>	
Número de vuelos a generar.	
<b>Resultados</b>	
Nuevo listado de vuelos en pantalla.	

<b>Nombre</b>	<b>R. # 3. Ordenar los vuelos por distintos criterios.</b>
<b>Resumen</b>	El programa por defecto organiza los vuelos por fecha y hora pero la pantalla ofrecerá al usuario la posibilidad de ordenar los vuelos por cualquiera de los otros criterios(aerolínea, número de vuelo, ciudades destino y puertas de embarque).
<b>Entradas</b>	
Criterio de ordenamiento.	
<b>Resultados</b>	
Nuevo orden de los vuelos.	

<b>Nombre</b>	<b>R. # 4. Buscar los vuelos por distintos criterios.</b>
<b>Resumen</b>	El programa ofrecerá al usuario la posibilidad de buscar un vuelo por cualquiera de los siguientes criterios: Fecha, hora, aerolínea, número de vuelo, ciudades destino y

	puertas de embarque. Si hay más de un vuelo que concuerde con el criterio buscado, entonces se mostrará el primer vuelo encontrado.
<b>Entradas</b>	
Criterio de búsqueda.	
<b>Resultados</b>	
Vuelo encontrado.	

<b>Nombre</b>	<b>R. # 5. Mostrar el tiempo que tardó un ordenamiento o una búsqueda.</b>
<b>Resumen</b>	El programa es capaz de que cada vez que se haga una operación búsqueda o de ordenamiento, mostrar el tiempo que tardó en completar dicha operación.
<b>Entradas</b>	
Operación.	
<b>Resultados</b>	
Tiempo que duró en pantalla.	

## Requerimientos no funcionales

<b>Nombre</b>	<b>RNF. 1. Visualizar el programa a través de una interfaz gráfica hecha con JavaFX.</b>
<b>Resumen</b>	El usuario podrá visualizar e interactuar con el programa a través de una interfaz gráfica construida en JavaFX que mostrará todas las opciones que el usuario debe llenar y además desplegará el cuadrado mágico.
<b>Entradas</b>	
<Ninguna>	
<b>Resultados</b>	
Interfaz gráfica producida con JavaFX.	

## Diagrama de clases

### Trazabilidad del Análisis al Diseño.


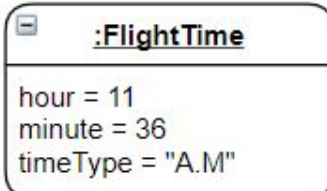


Requerimiento Funcional	Método	Clase
R1. Mostrar la información de los vuelos.	toString() toString() toString() initialize()	Flight FlightDate FlightTime AirlineController

	generateListView() fillList()	AirlineController AirlineController
R2. Generar aleatoriamente una lista de vuelos.	generateFlights() getRandomFlight() generateListView() fillList()	Airport Airport AirlineController AirlineController
R3. Ordenar los vuelos por distintos criterios.	bubbleSort() selectionSort() insertionSort() sortAirline() sortDate() sortDestination() sortFlightNumber() sortGate() sortTime() compareTo() compareTo() compareTo()	Airport Airport Airport AirlineController AirlineController AirlineController AirlineController AirlineController AirlineController Flight FlightDate FlightTime
R4. Buscar los vuelos por distintos criterios.	searchMenu() search() linearSearchDate() linearSearchTime() binarySearchAirline() binarySearchFlightNumber() binarySearchDestination() binarySearchBoardingGate()	AirlineController Airport Flight Flight Flight Flight Flight Flight
R5. Mostrar el tiempo que tardó un ordenamiento o una búsqueda.	searchMenu() sortAirline() sortDate() sortDestination() sortFlightNumber() sortGate() sortTime()	AirlineController AirlineController AirlineController AirlineController AirlineController AirlineController AirlineController

#### Configuración de los Escenarios

Nombre	Clase	Escenario
setupScenario1	FlightDateTest	vacío



setupScenary2	FlightDateTest	
setupScenary1	FlightTimeTest	vacío
setupScenary2	FlightTimeTest	
setupScenary1	FlightTest	vacío
setupScenary2	FlightTest	
setupScenary1	AirportTest	vacío
setupScenary2	AirportTest	

#### Diseño de Casos de Prueba

**Objetivo de la Prueba:** Verificar la correcta creación de una fecha de vuelo.

Clase	Método	Escenario	Valores de Entrada	Resultado
FlightDate	FlightDate	setupScenary1	day= 2 month= 11 year = 2018	true. Se ha creado una nueva fecha de vuelo exitosamente. Los atributos de esta están asignados de manera

				correcta.
--	--	--	--	-----------

**Objetivo de la Prueba:** Verificar la correcta comparación de una fecha de vuelo.

Clase	Método	Escenario	Valores de Entrada	Resultado
FlightDate	compareTo	setupScenario2	day= 17 month= 8 year = 2019	true. El método retorna el valor esperado al ser comparado con la fecha de vuelo del escenario 2.
FlightDate	equals	setupScenario2	day= 17 month= 8 year = 2019	true. El método retorna el valor esperado al ser comparado con la fecha de vuelo del escenario 2.

**Objetivo de la Prueba:** Verificar la correcta creación de una hora de vuelo.

Clase	Método	Escenario	Valores de Entrada	Resultado
FlightTime	FlightTime	setupScenario1	hour = 2 minute = 11 timeType = "A.M"	true. Se ha creado una nueva hora de vuelo exitosamente. Los atributos de esta están asignados de manera correcta.

**Objetivo de la Prueba:** Verificar la correcta comparación de una fecha de vuelo.

Clase	Método	Escenario	Valores de Entrada	Resultado
FlightTime	compareTo	setupScenario2	hour = 11 minute = 45 timeType = "A.M"	true. El método retorna el valor esperado al ser comparado con la hora de vuelo del escenario 2.
FlightTime	equals	setupScenario2	hour = 11 minute = 45 timeType = "A.M"	true. El método retorna el valor esperado al ser comparado con la hora de vuelo del escenario 2.

**Objetivo de la Prueba:** Verificar la correcta creación de un vuelo.

Clase	Método	Escenario	Valores de Entrada	Resultado
Flight	Flight	setupScenario1	airline = "Avianca" flightNumber = "AV2301"; destination = "Beijing"; boardingGate = 12; flightDate = 18/4/2018 flightTime = 11:23 A.M	true. Se ha creado un nuevo vuelo exitosamente. Los atributos de este están asignados de manera correcta.



**Objetivo de la Prueba:** Verificar la correcta comparación de un vuelo.

Clase	Método	Escenario	Valores de Entrada	Resultado
Flight	compareTo	setupScenario2	Ninguno.	true. El método retorna el valor esperado al ser comparado con el vuelo del escenario 2.

**Objetivo de la Prueba:** Verificar la correcta búsqueda de un vuelo.

Clase	Método	Escenario	Valores de Entrada	Resultado
Flight	linearSearchDate	setupScenario2	Ninguno.	true. El valor buscado es encontrado y se retorna su posición. Al comparar el valor en la posición con el buscado estos son iguales.
Flight	linearSearchTime	setupScenario2	Ninguno.	true. El valor buscado es encontrado y se retorna su posición. Al comparar el valor en la posición con el buscado estos son iguales.
Flight	binarySearchAirline	setupScenario2	Ninguno.	true. El valor buscado es encontrado y se retorna su posición. Al comparar el valor en la posición con el buscado estos son iguales. La lista se ordena antes de hacer la búsqueda.
Flight	binarySearchFlightNumber	setupScenario2	Ninguno.	true. El valor buscado es encontrado y se retorna su posición. Al comparar el valor en la posición con el buscado estos son iguales. La lista se ordena antes de hacer la búsqueda.
Flight	binarySearchDestination	setupScenario2	Ninguno.	true. El valor buscado es encontrado y se retorna su posición. Al comparar el valor en la posición con el buscado estos son iguales. La lista se ordena antes de hacer la búsqueda.
Flight	binarySearchBoardingGate	setupScenario2	Ninguno.	true. El valor buscado es encontrado y se retorna su posición. Al comparar el valor en la posición con el buscado estos son iguales. La lista se ordena antes de hacer la búsqueda.

**Objetivo de la Prueba:** Verificar la correcta creación de un aeropuerto.

Clase	Método	Escenario	Valores de Entrada	Resultado
-------	--------	-----------	--------------------	-----------



Airline	Airline	setupScenario1	ninguno.	true. Se ha creado un nuevo aeropuerto exitosamente. La lista de vuelos ha sido inicializada de forma correcta.
---------	---------	----------------	----------	--

**Objetivo de la Prueba:** Verificar la correcta generación de vuelos aleatorios.

Clase	Método	Escenario	Valores de Entrada	Resultado
Airline	generateFlights	setupScenario2	n = 10	true. Los vuelos fueron generados y el tamaño de la lista de vuelos es ahora 10.

**Objetivo de la Prueba:** Verificar el correcto ordenamiento de la lista de vuelos, dependiendo del criterio especificado.

Clase	Método	Escenario	Valores de Entrada	Resultado
Airline	bubbleSort	setupScenario2	n = 10	true. Los vuelos de la lista fueron ordenados de forma ascendente por el algoritmo de burbuja.
Airline	selectionSort	setupScenario2	n = 10	true. Los vuelos de la lista fueron ordenados de forma ascendente por el algoritmo de selección.
Airline	insertionSort	setupScenario2	n = 10	true. Los vuelos de la lista fueron ordenados de forma ascendente por el algoritmo de inserción.