

**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 Comparador.
- 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.

**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).

**1. Requerimientos funcionales.**

Nombre	R1: mostrar la información de los vuelos con hora no militar.
Resumen	se requiere que el programa muestre la hora con hora no militar para que sea más legible a las personas.
Entradas	hora.
Resultado	Se muestra la hora en formato no militar.

Nombre	R2 :generar aleatoriamente un listado de vuelos
Resumen	se genera aleatoriamente un listado de vuelos en diferentes fechas, horarios, diferentes aerolíneas, diferentes números de vuelo (éste debe ser único), ciudades destino y puertas de embarque. (El orden por defecto en que los vuelos son mostrados es la hora de salida de menor a mayor).
Entradas	Fecha. Hora. Aerolíneas. Número de vuelo. Ciudades destino. Puertas de embarque
Resultado	Se genera exitosamente un listado de vuelos con los atributos correspondientes.

Nombre	R3:permitir que el usuario interactúe con con la pantalla.
Resumen	Se debe permitir que el usuario interactúe con la pantalla ya sea a partir de menús o botones.

**Algoritmos de Ordenamiento y Búsqueda, en Listas Enlazadas**

Entradas	Ninguna
Resultado	Se crean formas de que el usuario interactúe con la pantalla

Nombre	R4: permitir cada vez que el usuario lo desee, generar aleatoriamente una nueva lista de vuelos
Resumen	Según sea la decisión del usuario se debe generar una lista de vuelos aleatoria.
Entradas	lista de vuelos.
Resultado	Se permite generar una lista de vuelos aleatoria según las condiciones dadas por el usuario.

Nombre	R5: mostrar los siguientes n vuelos y los n vuelos anteriores.
Resumen	Se debe tener la opción de mostrar en pantalla los vuelos siguientes y los vuelos anteriores.
Entradas	Ninguna.
Resultado	Se muestran los vuelos siguientes y los anteriores.

Nombre	R6: ordenar los vuelos es por fecha y hora.
Resumen	Inicialmente los vuelos deben estar ordenados según la fecha y la hora.
Entradas	Hora. Fecha.
Resultado	Se debe ordenar los vuelos según su hora y su fecha.

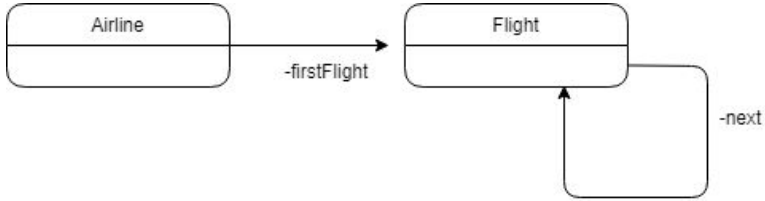
Nombre	R7: permitir al usuario buscar los vuelos por cualquier criterio.
Resumen	El usuario debe poder buscar los vuelos por cualquiera de los criterios.
Entradas	Lista de vuelos.
Resultado	Se le permite al usuario seleccionar una opción de búsqueda para los vuelos.

Nombre	R8: permitir al usuario ordenar los vuelos por cualquiera de los otros criterios.
Resumen	El usuario debe poder escoger por cuál criterio ordenar los vuelos.
Entradas	Lista de vuelos.
Resultado	Se le permite al usuario seleccionar una opción de ordenamiento para los vuelos.

**Algoritmos de Ordenamiento y Búsqueda, en Listas Enlazadas**

R#	Método	Clase
R1	generateHour(): String	Hour
R2	generateNewFlights(int amount) getFlights(): ObservableList<Flight> addFlight(): void	Airport AirlineController
R3	start(ActionEvent event): void Search(ActionEvent event): void	AirlineController
R4	Search(ActionEvent event) addFlight(): void	AirlineController Airport
R5		
R6	SortingDate()	Bubble
R7	searchDate(Flight first,String date): Flight searchHour(Flight first,String Hour): Flight searchAirline(Flight first,Airlines airline): Flight searchIdentifier(Flight first, Code id): Flight searchGate(Flight first, Doors sDoor): Flight searchDestination(Flight first, Citys dCity): Flight	lineal
R8		

**Configuración de los Escenarios**

Nombre	Clase	Escenario
SetupScenary1	Airport	 <pre> graph LR     Airline[Airline] -- "-firstFlight" --&gt; Flight[Flight]     Flight -- "-next" --&gt; Flight </pre>

**Diseño de Casos de Prueba:**

**Algoritmos de Ordenamiento y Búsqueda, en Listas Enlazadas**

Objetivo de la prueba: Crear una lista aleatoria de vuelos.

Clase	Método	Escenario	Valores de Entrada	Resultado
Airport	generateNewFlights(int amount)	setupScenario 1	ninguno	Se genera exitosamente una lista donde todos sus atributos son aleatorios

Objetivo de la prueba: Generar una lista de vuelos aleatorios del tamaño dado por el usuario

Clase	Método	Escenario	Valores de Entrada	Resultado
AirlineController	addFlight	setupScenario 1	3	Se agregan nuevos elementos a la lista.

**Objetivo de la Prueba:** Ordenar los listados de vuelos por hora y fecha.

Clase	Método	Escenario	Valores de Entrada	Resultado
Bubble	SortDate()	setupscenario 1	Ninguno	Se ordenan correctamente los valores de la lista

Searching

**Lineal**

+searchAirline(Flight first,Airlines airline): Flight

+searchDate(Flight first,String date): Flight

+searchHour(Flight first,String Hour): Flight

+searchIdentifier(Flight first, Code id): Flight

+searchIdentifier(Flight first, Code id): Flight

+searchDestination(Flight first, Citys dCity): Flight

Sorting

**Bubble**

-firstFlight: Flight

+SortingDate(): void

**Insertion**

+ field: type

+ method(type): type

**Selection**

+ field: type

+ method(type): type

userInterface

**Main**

.

+Main(String [] args): void

+start(Stage stage):void

FXML

**AirlineController**

-airport : Airport

-TABLE : TableView<Flight>

-borderpane: BorderPane

-flights: ObservableList<Flight>

-txtNVuelos:TextField

-start: Button

-Search: Button

-Sort: ComboBox<String>

-SearchF: ComboBox<String>

-random: Random

-amount: Integer

-lineal: Lineal

