Tarea programada # 2

SISTEMA DE BÚSQUEDA DE VUELOS

En este laboratorio el o la estudiante utilizará conceptos de programación orientada a objetos para resolver un problema, utilizando para ello arreglos de objetos y buscando una solución parametrizada.

Descripción del problema

Encontrar y comprar los vuelos más baratos para ir de una ciudad a otra puede ser una tarea muy difícil. Debido a que hay muchísima información y muchísimas opciones relacionadas con diferentes aerolíneas, para obtener una buena opción hay que organizar la información de acuerdo con las prioridades de la persona. Además, el tiempo invertido en esta tarea de organización podría ser nula, pues los valores cambian continuamente, y el comportamiento es impredecible.

Múltiples sitios en Internet brindan la opción de optimizar estas búsquedas, integrando la información de diferentes aerolíneas y sitios de turismo para tratar de hacer sugerencias de acuerdo con los criterios acordados en un inicio.

En esta práctica, usted deberá diseñar un sistema que le permita a la persona investigar sobre los datos que se tienen de los vuelos disponibles, hacer búsquedas rápidas y sugerir las mejores opciones para ser compradas.

Cómo funciona el sistema

El sistema carga toda la información de los vuelos. Luego, los vuelos se agrupan en aerolíneas. Finalmente, el control general del sistema procesa la información de todas las aerolíneas según sea necesario.

Las aerolíneas almacenan los vuelos en una matriz de 3 dimensiones: para cada ciudad de despegue y su correspondiente destino (2 dimensiones), guarda todos los vuelos relacionados (3ra dimensión).

Respetando el paradigma de Programación Orientada a Objetos, se requiere diseñar un sistema que sea capaz de realizar las siguientes tareas:

A- Cargar de un archivo de texto todos los vuelos disponibles para una aerolínea.
B- Realizar una búsqueda de la **opción más barata** de compra.

Cuando realiza una búsqueda, debe tomar en consideración los siguientes aspectos como mínimo:

- Ciudad de partida
- Ciudad de destino
- Fecha de inicio del viaje
- Cuántas escalas permitiría realizar como máximo
- Si admite escalas, entonces si podrían utilizarse varias aerolíneas en su itinerario.

Aspectos estratégicos por tomar en consideración

A. Se puede asumir un tamaño máximo de vuelos que una Aerolínea hace de un destino a otro, en 150.

B. Se puede asumir un tamaño máximo de Aerolíneas que mantiene el sistema, puede fijarse en 50.

C. El sistema carga la información de archivos de texto. Cada archivo de texto representa a una aerolínea diferente. El formato y extensión que debe seguir estos archivos de texto puede asumirlo, pero debe explicarlo en el README, y también brindar algunos ejemplos cuando envíe sus archivos finales.

D. No es necesario implementar la opción de cambiar la información de los vuelos en ejecución. Se asume que la información que se carga de los archivos es actualizada y que si hubo cambios se harán sobre los archivos de texto y se volvería a correr el programa.

E. El manejo de las escalas máximas es muy importante. Una escala es una parada en el itinerario de viaje, pero que es diferente al destino final. Esto tiene varias implicaciones en los algoritmos de búsqueda sobre los vuelos. La persona tendrá la opción de solicitar 0 escalas, por lo que solo se hará la búsqueda con vuelos directos. No obstante, también tendrá la opción de hacer escalas, en cuyo caso se deben evaluar todas las opciones disponibles.

F. Si se especifica que se permiten escalas a la hora de hacer la búsqueda, se debe considerar que para ir de un destino de A hacia B, es posible que el vuelo directo de A hacia B sea más caro que si se hace una escala en C (es decir, que puede que sea más barato hacer el recorrido A-C-B). Inclusive, puede que sea más barato hacer 2 escalas (por ejemplo A-C-D-B). Por ello, se solicita a la persona que coloque un número máximo de escalas permitidas. En este ejemplo, si la persona solo permitió una escala como máximo, la opción A-C-D-B no debe ser la opción correcta.

G. Cuando se sugieren itinerarios con más de un vuelo, es importante que se tome en consideración que para tomar un vuelo, debe haber aterrizado el anterior. Por ejemplo, si el vuelo A-B arriba a las 7 pm, y luego un vuelo B-C sale a las 5 pm del mismo día, es imposible que se tomen esos vuelos para hacer la ruta A-B-C.

H. La lista de ciudades en las que los vuelos operan depende de las aerolíneas. Cada aerolínea mantendrá registrados los puntos en donde opera, para despegues y destinos. Esta información será relevante para interpretar la información que contiene guardado en su matriz de vuelos. Puede asumir que las aerolíneas tienen entre sí convenciones de nombres de 3 letras para referirse a un lugar. Por ejemplo, para la ciudad de Alajuela en Costa Rica siempre se usará el código SJO.

Cómo se debe ver

La interacción del programa debe ser clara y precisa. No debe permitir entradas inválidas. Al inicio puede solicitar nombres de archivos de los cuales debe cargar la información. Al cargar toda la información, puede pasar a la etapa de búsquedas.

- Si se hace búsqueda de vuelos directos, se debe mostrar la opción más barata de cada aerolínea.
- Si se hace búsqueda permitiendo escalas, pero de una misma aerolínea, se debe mostrar la opción más barata de cada aerolínea.
- Si se hace búsqueda permitiendo escalas, de varias aerolíneas, se debe mostrar una opción: la más barata.

Opcional

OP1. Implemente que las búsquedas de varias escalas y varias aerolíneas no retornen únicamente un resultado, sino el top 10 de opciones de itinerario según precio.

OP2. Siempre que tenga más de un resultado, muéstrelos ordenados del más barato al más caro.

Requisitos mínimos de implementación

Usted puede realizar su diseño e incorporar cuantas clases considere necesarias, pero debe implementar la solución usando **como mínimo** los siguientes componentes:

- Diseñe una clase **Vuelo**. Los objetos de tipo Vuelo cuentan con las siguientes características como mínimo:
    - Representa un vuelo directo, por lo que tienen todas las características necesarias. Como mínimo: precio, lugar fecha y hora de despegue, lugar fecha y hora de destino, cantidad de asientos disponibles, código único que lo identifica.
    - Permite ver la información de forma ordenada e intuitiva.
    - Podría implementar opciones para definir compatibilidad con otros vuelos en un itinerario.
    - Podría implementar operaciones de uso interno de la clase.

- Diseñe una clase **Aerolínea.** Los objetos de tipo Aerolínea representan toda la información relacionada a los vuelos y destinos con que cuenta una aerolínea. Cuentan con las siguientes características como mínimo:
    - Mantiene la información básica de la aerolínea, como el nombre.
    - Mantiene un arreglo unidimensional con las ciudades donde opera.
    - Mantiene una matriz multidimensional de vuelos.
    - Permite obtener el vuelo óptimo directo de un destino a otro.
    - Permite obtener el itinerario óptimo de un destino a otro, si se consideran escalas.
    - Podría implementar operaciones de uso interno de la clase.

- Diseñe una clase **Búsqueda**, que funciona como clase controladora, pues manipula y hace llamados a las demás instancias que almacenan información. Los objetos de tipo Búsqueda cuentan con las siguientes características como mínimo:
    - Mantiene una lista de aerolíneas.
    - Carga la información de archivos de texto para mantener los vuelos en las aerolíneas.
    - Permite obtener el itinerario óptimo de un destino a otro, si se consideran escalas y considerando varias aerolíneas.
    - Permite realizar búsquedas.
    - Maneja la interacción con la persona que ingresa los datos, mostrando también los resultados. (Esto puede ir en esta clase, o separado en una clase Interacción).
    - Podría implementar operaciones de uso interno de la clase.

- Diseñe una clase **Main** que contiene el método principal del programa y hace un llamado inicial.

Limitaciones técnicas

Debe resolver todo utilizando únicamente los temas cubiertos en clase, por lo que únicamente puede incluir librerías contempladas en el curso, variables de tipo primitivas y tipo String, uso de arreglos uni o multidimensionales, pero no memoria dinámica ni plantillas. Tampoco se permite el uso de la clase importada Arrays.

La interacción de usuario(a) puede realizarla por entrada y salida estándar. El método principal únicamente debe contener un instanciado de otra clase, y un llamado inicial a otro método.

No use métodos ni variables estáticas, a menos de que lo considere indispensable para la solución. Si implementa algo estático, deberá justificarlo en el archivo de readme.

Debe resolver el problema utilizando programación orientada a objetos. Esto incluye restricciones como declarar los atributos privados, hacer constructores correspondientes, usar nombres significativos, usar encapsulamiento, entre otros.

Si es necesario, use manejo de excepciones para que, en caso de error, no se detenga abruptamente.

Suposiciones de programación

Realice un archivo README.txt dentro de los archivos, donde debe especificar suposiciones técnicas que hizo a la hora de desarrollar su programa. Por ejemplo, una suposición válida podría ser la cantidad máxima de vuelos que va a manejar la aerolínea a la vez, o la cantidad de dígitos del ID de un vuelo.

Describa en este archivo el formato que utilizó para los ficheros.

En este mismo archivo, especifique la forma en que se debe compilar todo el proyecto.

Formato de entrega

Entregue el reporte de laboratorio antes de la fecha y hora descrita en la plataforma virtual. Debe entregar los archivos **de tipo .java** con la solución, así como el README y al menos 5 archivos de información de vuelos que usó como pruebas. La entrega debe ser un archivo comprimido en .zip, .rar o .7z y el título debe ser: número de carné, guion bajo, "TP2". Por ejemplo, si el carné es *A12345*, el archivo debe llamarse "*A12345_TP2*". Si trabajó en pareja o trío, debe colocar todos los carnés en el nombre del archivo, y entregar además un documento de coevaluación.

Rúbrica de evaluación

Solo se obtiene calificación si cumple **todas** las restricciones técnicas establecidas. Caso contrario, se evalúa como cero. Si cumple las restricciones, entonces se sigue la siguiente rúbrica de evaluación.

Rúbrica	
Vuelo: Constructor(es), atributos y definición general de clase	5%
Aerolínea: Constructor(es), atributos y definición general de clase	5%
Aerolínea: Itinierario óptimo directo	10%
Aerolínea: Itinierario óptimo indirecto	15%
Búsqueda: Constructor(es), atributos y definición general de clase	5%
Búsqueda: Itinierario óptimo indirecto con varias aerolíneas	20%
Búsqueda: Carga de archivos de texto	15%
Búsqueda: Interacción y realización de búsquedas	10%
Validaciones de entradas y manejo de excepciones	5%
Orden del código, main y uso de buenas prácticas de programación	5%
Entrega correcta: incluye README, archivos de ejemplo, coevaluación	5%
Opcional: mostrar resultados ordenados de barato a caro	+5%
Opcional: mostrar ranking, para búsqueda avanzada	+5%

Ejemplo de ejecución[1]

```
=== Bienvenid@ al sistema de búsqueda de
vuelos baratos ===

Primero se va a cargar la información.

Ingrese el nombre de una Aerolínea, o bien un
punto para indicar que ha terminado: Delta
Ingrese el nombre de archivo que contiene la
información de los vuelos de Delta: Delta.txt
Se ha cargado la información.

Ingrese el nombre de una Aerolínea, o bien un
punto para indicar que ha terminado: Avianca
Ingrese el nombre de archivo que contiene la
información de los vuelos de Delta: Av.txt
Se ha cargado la información.

Ingrese el nombre de una Aerolínea, o bien un
punto para indicar que ha terminado: Copa
Ingrese el nombre de archivo que contiene la
información de los vuelos de Delta:
CopaAir.txt
Se ha cargado la información.

Ingrese el nombre de una Aerolínea, o bien un
punto para indicar que ha terminado: AA
Ingrese el nombre de archivo que contiene la
información de los vuelos de Delta: AA.txt
Ocurrió un error al cargar la información, el
archivo no respeta el formato debido.

Ingrese el nombre de una Aerolínea, o bien un
punto para indicar que ha terminado: .

--- INICIO DE BÚSQUEDA ---

Ingrese la ciudad de partida: SJO
Ingrese la ciudad de destino: EZE
Ingrese la fecha de inicio (DD-MM-AAAA):
27 oct 2023
Debe seguir el formato establecido.
Ingrese la fecha de inicio (DD-MM-AAAA):
27-10-2023
Cuántas escalas permite como máximo: 0

--- RESULTADOS POR AEROLINEA ---

Delta - Vuelo A205
Despegue en SJO: 27-10-2023 05:00
Arribo en EZE: 27-10-2023 09:03
Precio: 953.45 USD
85 asientos disponibles

Avianca - Vuelo AV2046
Despegue en SJO: 27-10-2023 08:25
Arribo en EZE: 27-10-2023 12:23
Precio: 740.90 USD
45 asientos disponibles

Copa - Vuelo CO-AV5120
Despegue en SJO: 27-10-2023 23:55
Arribo en EZE: 28-10-2023 04:24
Precio: 810.21 USD
12 asientos disponibles

Ingrese B para hacer otra búsqueda o S para
Salir: B

--- INICIO DE BÚSQUEDA ---

Ingrese la ciudad de partida: SJO
Ingrese la ciudad de destino: EZE
Ingrese la fecha de inicio (DD-MM-AAAA):
27-10-2023
Cuántas escalas permite como máximo: UNA
Ingrese un valor numerico valido. Cuántas
escalas permite como máximo: 1
Permite cruce de aerolíneas? (S/N): S

--- MEJOR RESULTADO ---

Delta
Vuelo A245
Despegue en SJO: 27-10-2023 07:14
Arribo en LIM: 27-10-2023 09:23
Precio: 370.2 USD
40 asientos disponibles

Avianca
Vuelo AV3205
Despegue en LIM: 27-10-2023 12:40
Arribo en EZE: 27-10-2023 14:31
Precio: 351.0 USD
25 asientos disponibles

Total 721.2 USD

Ingrese B para hacer otra búsqueda o S para
Salir: S

== Fin del programa ==
```

[1] Los valores en negrita representan entradas del teclado.