Ejercicio: Aeropuerto

Autor: Mariano González. **Revisores**: -. **Última modificación**: 26/5/2020

En este ejercicio vamos a trabajar con información sobre los vuelos operados en un aeropuerto. Para ello disponemos de datos obtenidos de la web FlightRadar24 (por ejemplo, https://www.flightradar24.com/data/airports/svq/departures).

Para cada vuelo se tiene un código que lo identifica, una fecha, una hora planificada y una hora efectiva (el vuelo puede tener retraso o adelanto), una ciudad y un código de aeropuerto (que corresponderán al destino u origen del vuelo, según sea de salida o llegada, respectivamente), una compañía, un modelo de avión, un id de avión, un estado y una dirección (salida o llegada).

Los datos se encuentran almacenados en ficheros en formato CSV codificados en UTF-8. Cada registro de un fichero ocupa una línea y contiene los datos correspondientes a una operación de vuelo. Por ejemplo, estas son las primeras líneas de un fichero con los vuelos del aeropuerto de Sevilla (exceptuando la línea de cabecera):

```
05/02/2018,6:45,FR1204,Verona,VRN,Ryanair,B738,EI-EKK,Departed,6:49,out
05/02/2018,6:50,VY2223,Barcelona,BCN,Vueling,A320,EC-MDZ,Departed,8:36,out
05/02/2018,7:00,FR1188,Memmingen,FMM,Ryanair,B738,EI-FIS,Departed,7:10,out
05/02/2018,7:00,TP1103,Lisbon,LIS,TAPExpress,AT76,CS-DJF,Departed,7:00,out
05/02/2018,7:10,FR3041,Santiago,SCQ,Ryanair,B738,EI-EVT,Departed,7:16,out
```

Los tipos que forman el modelo de datos son los siguientes:

- **EstadoVuelo**: tipo enumerado con los posibles estados de un vuelo.
- **DireccionVuelo**: tipo enumerado con las posibles direcciones de un vuelo.
- **Vuelo**: tipo que representa un vuelo de salida o llegada en un aeropuerto.
- **Aeropuerto**: tipo contenedor que representa todas las operaciones de vuelo en un aeropuerto.

Tipo Vuelo

Propiedades:

- **fecha**, de tipo LocalDate. Consultable. Fecha de salida o llegada del vuelo.
- **hora planificada**, de tipo LocalTime. Consultable. Hora prevista de salida o llegada del vuelo.
- **código**, de tipo String. Consultable. Código que identifica al vuelo.
- **ciudad**, de tipo String. Consultable y modificable. Ciudad de destino del vuelo, si es de salida, o de origen, si es de llegada.
- **código de aeropuerto**, de tipo String. Consultable. Código del aeropuerto de destino del vuelo, si es de salida, o de origen, si es de llegada.
- **compañía**, de tipo String. Consultable. Compañía que opera el vuelo.
- **modelo de avión**, de tipo String. Consultable. Modelo del avión que realiza el vuelo.
- **id de avión**, de tipo String. Consultable. Identificador del avión.

- **estado**, del tipo enumerado EstadoVuelo, que puede tomar los valores CANCELED, DEPARTED, ESTIMATED, LANDED, SCHEDULED y UNKNOWN. Consultable. Estado del vuelo.
- **hora efectiva**, de tipo LocalTime. Consultable. Hora real de salida o llegada del vuelo. En el fichero, la hora efectiva de un vuelo cancelado tiene el valor null.
- **dirección**, del tipo enumerado DireccionVuelo, que puede tomar los valores IN y OUT. Consultable. Dirección del vuelo.
- **diferencia en minutos**, de tipo Long. Consultable. Son los minutos de diferencia entre la hora efectiva y la hora planificada (positiva si el vuelo tiene adelanto, negativa si tiene retraso y 0 si es puntual). Si el vuelo está cancelado, su valor será null.

Constructores:

- Un constructor que recibe un parámetro por cada propiedad básica del tipo, en el mismo orden en el que están definidas.

Criterio de igualdad: dos vuelos son iguales si tienen el mismo código, la misma fecha y la misma hora planificada.

Criterio de ordenación: por fecha, a igualdad de esta por hora planificada, y a igualdad de esta por código.

Representación como cadena: incluirá el código, la fecha y la hora planificada.

Tipo Aeropuerto

Propiedades:

- **vuelos**, de tipo SortedSet<Vuelo>. Consultable. Operaciones de vuelo registradas en el aeropuerto, tanto de salida como de llegada.

Constructores:

- Un constructor sin parámetros.
- Un constructor a partir de un Stream<Vuelo>.

Representación como cadena: generada automáticamente con todas las propiedades básicas del tipo.

Otras operaciones:

- *void añadirVuelo(Vuelo v)*: añade un vuelo al aeropuerto.

Tratamientos secuenciales:

- *Boolean existeVueloDestino(String ciudad)*: indica si existe algún vuelo que tiene como destino la ciudad dada.
- *Boolean todosVuelosCompañiaSinRetraso(String compañia)*: indica si todos los vuelos de la compañía dada son puntuales (su hora efectiva es igual o anterior a su hora planificada).
- *Long getNumeroVuelosSalida()*: obtiene el número de vuelos de salida.
- *Long getNumeroVuelosDestino(String ciudad)*: obtiene el número de vuelos que tienen como destino la ciudad dada.
- *Long getNumeroVuelosCancelados()*: obtiene el número de vuelos cancelados.

- *Long getNumeroVuelosCompañia(String compañia)*: obtiene el número de vuelos operados por la compañía dada.
- *Long getNumeroVuelosConRetraso()*: obtiene el número de vuelos con retraso, entendiendo por tales aquellos cuya hora efectiva es posterior a su hora planificada. No se incluyen los vuelos cancelados.
- *Set<String> getCompañiasVuelos()*: obtiene un conjunto con las compañías aéreas diferentes que operan en el aeropuerto.
- *Long getNumeroCiudadesDestino()*: obtiene el número de ciudades diferentes a las que se puede volar desde el aeropuerto.
- *Set<String> getModelosAvionesCompañia(String compañia)*: obtiene un conjunto con los modelos de aviones utilizados por una compañía dada.
- *Long getMinutosRetrasoAcumuladosCompañia(String compañia)*: obtiene el retraso acumulado de todos los vuelos operados por la compañía dada, en minutos. No se incluyen los vuelos cancelados.
- *Long getMediaMinutosRetrasoCiudad(String ciudad)*: obtiene el retraso medio de todos los vuelos que tienen por destino una ciudad. No se incluyen los vuelos cancelados.
- *Set<DayOfWeek> getDiasSemanaConVuelosDestino(String ciudad)*: obtiene los días de la semana en los que salen vuelos hacia un destino.
- *Vuelo getVueloMayorRetraso()*: obtiene el vuelo con más minutos de retraso. En caso de igualdad, desempatar por el orden natural.
- *Vuelo getVueloMasTempranoDestino(String ciudad)*: obtiene el vuelo con destino la ciudad dada que tiene una hora planificada anterior. En caso de igualdad, desempatar por el orden natural.
- *Vuelo getPrimerVueloCompañia(String compañia)*: obtiene el vuelo operado por la compañía dada que tiene una hora planificada anterior. En caso de igualdad, desempatar por el orden natural.
- *List<String> getPrimerosDestinos(Integer n)*: obtiene los n primeros destinos, ordenados por fecha y hora de salida.
- *void desviaVuelosCiudad(String ciudad, String nuevaCiudad)*: desvía a *nuevaCiudad* todos los vuelos de salida que tienen como destino *ciudad* (por ejemplo, debido a condiciones meteorológicas adversas).
- *Map<String, List<Vuelo>> getVuelosPorCiudad()*: obtiene un Map que relaciona las ciudades con los vuelos que tienen como origen o destino dicha ciudad.
- *Map<LocalDate, List<Vuelo>> getVuelosPorFecha()*: obtiene un Map que relaciona las fechas con los vuelos que salen o llegan en esa fecha.
- *Map<String, Set<String>> getModelosPorCompañia()*: obtiene un Map que relaciona las compañías con el conjunto de modelos de avión de la compañía.
- *Map<String, Long> getNumeroVuelosPorCiudad()*: obtiene un Map que relaciona las ciudades con el número de vuelos que tienen por origen o destino dicha ciudad.
- *Map<String, Integer> getNumeroVuelosConRetrasoPorCompañia()*:obtiene un Map que relaciona las compañías con el número de vuelos con retraso de la compañía.

- *SortedMap<Integer, Long> getNumeroSalidasPorHora()*: obtiene un SortedMap que relaciona cada hora del día con el número de salidas que se producen en dicha hora, en orden creciente de hora.
- *Map<String, Long> getRetrasoAcumuladoPorCompañia()*: obtiene un Map que relaciona las compañías con el retraso acumulado de todos los vuelos de la compañía.
- *Map<String, Double> getRetrasoMedioPorCompañia()*:obtiene un Map que relaciona las compañías con el retraso medio de los vuelos de la compañía.
- *Map<String, Vuelo> getVueloMasRetrasoPorCompañia()*: obtiene el vuelo con más minutos de retraso de cada compañía.
- *Map<String, Vuelo> getVueloMasTempranoPorCiudad()*: obtiene el vuelo que sale a una hora más temprana con destino a cada ciudad.
- *Map<String, Double> getPorcentajeVuelosConRetrasoPorCompañia()*: obtiene el porcentaje de vuelos con retraso de cada compañía respecto al número total de vuelos con retraso.
- *Map<LocalDate, List<String>> getNPrimerasCiudadesPorFecha(Integer n)*: obtiene las n primeras ciudades de los vuelos de cada fecha, ordenadas por la hora planificada de salida.
- *Map<Month, Integer> getNumeroDestinosDiferentesPorMes()*: obtiene el número de destinos diferentes a los que hay vuelos en cada mes del año.
- *String getModeloMasUtilizadoCompañia(String compañia)*: obtiene el modelo de avión más utilizado por una compañía.
- *String getDestinoMasVuelos()*: obtiene la ciudad a la que se dirigen más vuelos.
- *String getCompañiaMasOperaciones()*: obtiene la compañía con más operaciones de vuelo del aeropuerto.
- *Set<String> getCompañiasConAlMenosNVuelos(Integer n)*: obtiene las compañías que operan al menos n vuelos.
- *List<String> getCompañiasMasRetraso(Integer n)*: obtiene las n compañías con más vuelos con retraso, de mayor a menor.
- *Vuelo getVueloConModelo(String modelo)*: obtiene algún vuelo que use un modelo de avión dado.
- *String getCompañiaTodosVuelosConRetraso()*: obtiene alguna compañía que tenga todos sus vuelos con retraso.
- *void escribeDestinos(String nombreFichero, String compañia)*: crea un fichero con las ciudades de destino diferentes de una compañía.
- *void escribeCompañias(String nombreFichero)*: escribe un fichero con las compañías que operan en el aeropuerto.