Ejercicios Set.

Ejercicio 1. Supongamos un tipo Vuelo con las siguientes propiedades:

```
String destino;
Double precio;
Integer numPlazas; // debe ser positivo
Integer numPasajeros; // debe ser positivo y menor de numPlazas
String codigo;
Fecha fecha; // posterior al año 2000
```

Construya una interfaz Vuelo y una clase VueloImpl para implementar este tipo. Todos los atributos serán consultables y modificables sólo precio y numPasajeros. El orden natural será por fecha y a igualdad de ésta por código. Proporcione un constructor para Vuelo a partir de una cadena de caracteres que contenga todos los valores de sus atributos separados por comas.

Ejercicio 2. Un Aeropuerto tiene un nombre y guarda la información de los vuelos que salen del aeropuerto mediante un atributo de tipo Set<Vuelo> con todos los vuelos programados. Proporcione un constructor a partir de un String con el nombre del aeropuerto y el conjunto de vuelos vacío. La funcionalidad que el tipo Aeropuerto debe tener es la siguiente:

1. Añadir un vuelo
2. Quitar un vuelo
3. Dada una fecha dar el número de vuelos de ese día
4. ¿Cuántos vuelos completos (sin plazas libres) hay?
5. Dado un destino, obtenga la recaudación de todos los vuelos que van a ese destino (suma de número de pasajeros por precio)
6. Dado un destino, cuál es el vuelo más barato para ese destino.
7. Dado un destino, cuál es el primer vuelo con plazas libres para ese destino.
8. Dada una fecha, cuál es el vuelo de ese día con mayor porcentaje de plazas ocupadas (cociente entre número de pasajeros y número de plazas).
9. Dado un porcentaje p y una fecha f, incremente el precio de todos los vuelos a partir de f en el porcentaje p.

Ejercicio 3. Implemente una clase AeropuertoImpl2 para la interfaz Aeropuerto que cambie el atributo de tipo Set por uno te tipo SortedSet. Primero con el orden natural y después por un SortedSet definido a partir de un Comparator por Destino. Observe como "desaparecen" vuelos. Solucione el problema redefiniendo convenientemente el Comparator.

Ejercicio 4. Capture la excepción NoSuchElementException cuando se invoca Collections.sort con una colección vacía. Por ejemplo, si buscamos un vuelo barato a un destino para el que no hay aviones en el aeropuerto. Cree una excepción ExcepcionColeccionVacia no comprobada que informe convenientemente de que se ha hecho una consulta sin resultado. Trátela mediante una sentencia throws en el método y mediante try-catch en la clase Test. Cambie la excepción al tipo comprobada y haga los cambios necesarios para que funcione.

Ejercicio 5. Proporcione un constructor a Aeropuerto que reciba el nombre de un fichero de texto con una estructura como la del recuadro y, con la lectura de esa información, dé valor a la colección de objetos de tipo Vuelo que constituye el atributo privado de Aeropuerto. El nombre del aeropuerto coincidirá con el nombre del fichero sin la extensión .txt.

Madrid, 12.37, 155, 100, IB1123, 22, 11, 2007
Barcelona, 19.56, 200, 150, VLG256, 22, 11, 2007
Valencia, 2.1, 150, 150, RYA803, 22, 11, 2007
Paris, 10.0, 85, 85, UA894, 23, 11, 2007
Madrid, 22.37, 155, 154, IB2365, 23, 11, 2007
Bilbao, 29.56, 200, 150, EAS286, 23, 11, 2007
Valencia, 22.4, 100, 100, VLG127, 24, 11, 2007
Paris, 70.0, 75, 70, EAS348, 24, 11, 2007
Madrid, 32.37, 250, 250, AIF389, 24, 11, 2007
Barcelona, 39.56, 200, 150, UA7810, 24, 11, 2007
Londres, 28.4, 100, 90, IB6511, 25, 11, 2007
Paris, 80.0, 75, 75, RYA212, 25, 11, 2007