EJERCICIO 1 – TIPO BASE (2 puntos)

Se desea implementar parte de un sistema relacionado con la gestión de líneas de autobuses. Para ello se dispone de un archivo `viajes.csv`, donde la información de cada línea se corresponde con: el precio del viaje (en euros); la distancia en km que se recorre en el viaje; la duración, en horas y minutos; si el viaje es directo (Directo) o tiene transbordos (Transbordo); y el trayecto especificado por las distintas paradas en el viaje mediante una lista de ciudades separadas por comas entre corchetes. Dos paradas consecutivas y repetidas indican transbordo (en negrilla):

```
Precio;Distancia;Duracion;Tipo;Trayecto
14.99; 507;7:20;Cancelado;[Sevilla, Huelva, Faro, Aeropuerto Faro, Albufeira, Lisboa]
64.99; 1153;16:00;Transbordo;[Madrid,Hub,Hub,Bilbao,San Sebastián, Irún, Orthez, Toulouse, Nimes]
99.99; 923;12:50;Directo;[Paris,Lyon,Chambéry,Oulx,Turín,Milán]
69.04; 1487;24:50;Transbordo;[Grenoble,Valence,Aviñón,Nimes,Toulouse,Orthez,Irún,Irún,San
Sebastian,Pamplona,Tudela,Lleida,Barcelona]
…
```

Teniendo en cuenta que en el proyecto tiene definido el tipo enumerado **TipoViaje**, que puede tomar los valores DIRECTO, TRANSBORDO, CANCELADO se pide implementar la clase **Viaje** teniendo en cuenta la siguiente descripción.

Tipo Viaje:

Propiedades:

- precio, de tipo Double, consultable y modificable.
- distancia (en km), de tipo Integer, consultable y modificable. Distancia total recorrida en el viaje.
- duracion (en horas y minutos), de tipo Duration, consultable y modificable.
- tipo, de TipoViaje, consultable y modificable. Indica si el viaje es directo o tiene transbordos.
- trayecto, de tipo List<String>, consultable y modificable. Representa la lista con los nombres de las paradas del viaje, ordenadas de origen a destino.
- velocidad media (en km/h), de tipo Double, derivada. Representa la velocidad media, que se calcula a partir de la distancia y la duración.
- número de paradas, de tipo Integer, derivada. Representa el número de paradas intermedias entre el origen y el destino. Por ejemplo, el viaje Sevilla-Lisboa que aparece en el csv tiene 4 paradas.
- paradas, de tipo List<String>, derivada. Representa las paradas intermedias del trayecto, pudiendo ser una lista vacía si no hay paradas intermedias.
- origen, de tipo String, derivada. Representa el primer punto del trayecto.
- destino, de tipo String, derivada. Representa el último punto del trayecto.
- numero transbordos, de tipo Integer, derivada. Un transbordo se identifica en un trayecto porque la ciudad o punto de transbordo aparece repetida de forma consecutiva en el trayecto. Por ejemplo, en el viaje de Madrid a Nimes del csv hay un transbordo en Hub, y en el viaje Grenoble-Barcelona hay un transbordo en Irún.

Criterio de igualdad: por trayecto.

Representación como cadena: la de la lista para los trayectos. Ejemplo: "[Sevilla, Madrid, Barcelona]"

Constructores:

- C1: un constructor que tiene un como parámetros el precio, la distancia, la duración (de tipo Duration), el tipo de viaje, la ciudad origen y la ciudad destino del trayecto.
- C2: un constructor que tiene como parámetros cada una de las propiedades básicas del tipo.

Restricciones:

- El trayecto debe tener al menos dos paradas (origen y destino).
- La distancia, la duración y el precio deben ser mayores de cero.
- El tipo de viaje de un viaje con solo un origen y un destino no puede ser TRANSBORDO.

Solo se pide:

a) Cabecera y atributos de la clase. (0,25 puntos)
b) Constructor C1 (0,25 puntos)
c) Métodos relacionados con las propiedades derivadas: velocidad media, número de paradas, origen, destino, número de transbordos (1 punto)
d) Métodos relacionados con la propiedad duración (0,5 puntos)

EJERCICIO 2 – FACTORÍA (1,5 puntos)

Suponiendo que existe la clase **FactoriaViajes** para poder cargar la información del fichero de datos proporcionado. Se pide **SOLO** implementar el método **parsearViaje** que toma una línea con el formato del fichero csv mostrado anteriormente, y devuelve un objeto de tipo Viaje. Defina las funciones auxiliares que crea convenientes.

EJERCICIO 3 – TIPO CONTENEDOR (6,5 puntos)

Implemente el tipo contenedor **AgenciaBus**. La descripción del tipo es la siguiente:

Propiedades:

- nombre, de tipo String. Consultable. Nombre de la agencia.
- viajes, de tipo List<Viaje>. Consultable. Lista con viajes.

Constructores:

- Un constructor que tiene como parámetros un nombre y un Stream<Viaje> y crea un objeto de tipo AgenciaBus a partir del nombre y el Stream dados. (se supone implementado)

Otras operaciones (estas son las que debe implementar. Debe resolver estos métodos usando métodos de Stream):

a) **`hayViajesCirculares`**: Devuelve un valor booleano que indica si existe algún viaje circular, es decir, algún viaje que empiece y termine en la misma parada. (1 punto)
b) **`getMaximaDuracion:`** Devuelve la duración de tipo Duration del viaje más largo, siendo el viaje más largo aquél que consta de un mayor número de paradas. En caso de que no exista el viaje más largo, el método debe devolver una duración de cero. (1 punto)
c) **`añadirTiempoDescanso`**: Dados una parada concreta, de tipo String, y un tiempo extra en minutos, de tipo Integer, añade ese tiempo extra a los viajes cuyo trayecto incluya dicha parada. (1,5 punto)
d) **`getParadas:`** Devuelve un conjunto ordenado (SorteSet<String>) en orden alfabético de todas las posibles paradas intermedias por las que pasan los viajes. Tenga en cuenta que las paradas NO deben incluir el origen y el destino del viaje. (1,5 puntos)
e) **`ViajeDuracionMinimaPorDestinoPorParadas`**: Devuelve un `Map<String, Viaje>` Map que relaciona los destinos con el viaje de menor duración con dicho destino. (1,5 puntos)