Ejercicio 1

```
public record Parada(String nombre, LocalTime horaSalida) {
}


public class Viaje {

	private Double precio;
	private Integer distancia;
	private Duration duracion;
	private TipoViaje tipo;
	private List<Parada> trayecto;

	public Viaje (Double precio, Integer distancia, Duration duracion,
			TipoViaje tipo, Parada origen, Parada destino) {
		setPrecio(precio);
		setDistancia(distancia);
		setDuracion(duracion);
		this.tipo = tipo;
		this.trayecto = new ArrayList<Parada>();
		trayecto.add(origen);
		trayecto.add(destino);
	}

	public Viaje (Double precio, Integer distancia, Duration duracion,
			TipoViaje tipo, List<Parada>trayecto) {
		setPrecio(precio);
		setDistancia(distancia);
		setDuracion(duracion);
		Checkers.check("Trayecto incorrecto", trayecto.size() >= 2);
		this.tipo = tipo;
		this.trayecto = new ArrayList<Parada>(trayecto);
	}

	public Double getPrecio() {
		return precio;
	}

	public void setPrecio(Double precio) {
		Checkers.check("Precion incorrecto", precio > 0);
		this.precio = precio;
	}

	public Integer getDistancia() {
		return distancia;
	}

	public void setDistancia(Integer distancia) {
		Checkers.check("distancia incorrecta", distancia > 0);
		this.distancia = distancia;
	}

	public Duration getDuracion() {
		return duracion;
	}

	public void setDuracion(Duration duracion) {
```

```
		Checkers.check("Duración incorrecta",
				duracion.compareTo(Duration.ZERO) > 0);
		this.duracion = duracion;
	}

	public TipoViaje getTipo() {
		return tipo;
	}

	public void setTipo(TipoViaje tipo) {
		this.tipo = tipo;
	}

	public List<Parada> getTrayecto() {
		return new ArrayList<>(trayecto);
	}

	public Double getVelocidadMedia() {
		Double duracion = getDuracion().getSeconds() / 3600.0;
		return getDistancia() / duracion;
	}

	public Integer getNumeroParadas() {
		return getTrayecto().size() – 2 - getNumeroTransbordos();
	}

	public String getOrigen() {
		return getTrayecto().get(0).nombre();
	}

	public String getDestino() {
		return getTrayecto().get(getTrayecto().size() - 1).nombre();
	}

	public List<String> getParadas() {
		List<Parada> inter = getTrayecto().subList(1, getNumeroParadas() + 1);
		List<String> res = new ArrayList<>();
		for (Parada p: inter) {
			res.add(p.nombre());
		}
		return res;
	}

	public Integer getNumeroTransbordos() {
		Integer numTransbordos = 0;
			if (getTipo().equals(TipoViaje.TRANSBORDO)) {
				numTransbordos =
					calcularRepetidosConsecutivos(getTrayecto());
			}
		return numTransbordos;
	}

	private Integer calcularRepetidosConsecutivos(List<Parada> trayecto) {
		Integer cont = 0;
		for (int i=0; i < trayecto.size() - 2; i++) {
			if (trayecto.get(i).nombre().equals(
					trayecto.get(I + 1).nombre())) {
				cont++;
			}
```

```
		}
		return cont;
	}

	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((distancia == null) ? 0 : distancia.hashCode());
		result = prime * result + ((duracion == null) ? 0 : duracion.hashCode());
		result = prime * result + ((precio == null) ? 0 : precio.hashCode());
		result = prime * result + ((tipo == null) ? 0 : tipo.hashCode());
		result = prime * result + ((trayecto == null) ? 0 : trayecto.hashCode());
		return result;
	}

	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (!(obj instanceof Viaje))
			return false;
		Viaje other = (Viaje) obj;
		if (distancia == null) {
			if (other.distancia != null)
				return false;
		} else if (!distancia.equals(other.distancia))
			return false;
		if (duracion == null) {
			if (other.duracion != null)
				return false;
		} else if (!duracion.equals(other.duracion))
			return false;
		if (precio == null) {
			if (other.precio != null)
				return false;
		} else if (!precio.equals(other.precio))
			return false;
		if (tipo != other.tipo)
			return false;
		if (trayecto == null) {
			if (other.trayecto != null)
				return false;
		} else if (!trayecto.equals(other.trayecto))
			return false;
		return true;
	}

	public String toString() {
		return "Viaje [precio=" + precio + ", distancia=" + distancia
			+ ", duracion=" + duracion + ", tipo=" + tipo
			+ ", trayecto=" + trayecto + "]";
	}

}
```

Ejercicio 2

```
private static Viaje parsearViaje(String linea) {
	Checkers.checkNoNull(linea);

	String [] trozos = linea.split(";");
	Checkers.check("Formato no válido->" + trozos.length, trozos.length == 5);
	Double precio = Double.parseDouble(trozos[0].trim());
	Integer distancia = Integer.parseInt(trozos[1].trim());
	Duration duracion = parsearDuracion(trozos[2].trim());
	TipoViaje tipo = TipoViaje.valueOf(trozos[3].trim().toUpperCase());
	List<Parada> trayecto = parsearTrayecto(trozos[4].trim());
	return new Viaje(precio, distancia, duracion, tipo, trayecto);
}

private static Duration parsearDuracion(String strDuracion) {
	Checkers.checkNoNull(strDuracion);
	String [] trozos = strDuracion.split(":");
	Checkers.check("Formato no válido", trozos.length == 2);
	Integer horas = Integer.parseInt(trozos[0].trim());
	Integer minutos = Integer.parseInt(trozos[1].trim());
	return Duration.ofHours(horas).plusMinutes(minutos);
}

private static List<Parada> parsearTrayecto(String strTrayecto) {
	String limpia = strTrayecto.replace("[", "").replace("]","");
	String [] trozos = limpia.split(",");
	Checkers.check("Formato no válido->" + trozos.length + "-->" +
		Arrays.toString(trozos), trozos.length >= 2);
	List<Parada> res = Arrays.stream(trozos)
				.map(FactoriaViajes::parseaParada)
				.collect(Collectors.toList());
	return res;
}

private static Parada parseaParada(String strParada) {
	String [] trozos = strParada.split("-");
	String parada = trozos[0].trim();
	String strHora =trozos[1].trim();
	LocalTime hora = null;
	if (!strHora.toUpperCase().equals("FIN")) {
		hora = parseaHora(strHora);
	}
	return new Parada(parada, hora);
}

private static LocalTime parseaHora(String strHora) {
	return LocalTime.parse(strHora, DateTimeFormatter.ofPattern("HH:mm"));
}
```

Ejercicio 3

Apartado a)

```
public Duration getMaximaDuracion() {
	Duration res = viajes.stream()
			.max(Comparator.comparing(Viaje::getNumeroParadas))
			.map(Viaje::getDuracion)
			.orElse(Duration.ZERO);
	return res;
}
```

Test

```
private static void testGetMaximaDuracion(AgenciaBus agencia) {
	try {
		Duration res = agencia.getMaximaDuracion();
		String msg = String.format("La duración máxima es %s", res);
		System.out.println(msg);
	} catch (Exception e) {
		System.out.println("Capturada excepcion inesperada " +
		  e.getMessage());
	}
}

public static void main(String[] args) {
  AgenciaBus agencia = FactoriaViajes.leerViajes("Flixbus", "data/viajes.csv");
  System.out.println("Test 4.1 ============================================");
  testGetMaximaDuracion(agencia);
  …
}
```

Apartado b)

```
public void añadirTiempoDescanso(String parada, Integer minutos) {
	viajes.stream()
		.filter(viaje -> viaje.getParadas().contains(parada))
		.forEach(viaje ->
		   viaje.setDuracion(viaje.getDuracion().plusMinutes(minutos)));
}
```

Test

```
private static void testAñadirTiempoDescanso(AgenciaBus agencia, String parada,
		Integer minutos) {
	try {
		String msg = String.format("Se van a añadir %d minutos a los viajes
				con la parada %s, que son", minutos, parada);
		System.out.println(msg);
		mostrar_viajes_con_parada(agencia, parada);
		agencia.añadirTiempoDescanso(parada, minutos);
		System.out.println("Los viajes tras la modificación son");
		mostrar_viajes_con_parada(agencia, parada);
	} catch (Exception e) {
		System.out.println("Capturada excepcion inesperada " +
				e.getMessage());
	}
```

```
}

public static void main(String[] args) {
  AgenciaBus agencia = FactoriaViajes.leerViajes("Flixbus", "data/viajes.csv");
  System.out.println("Test 4.2 ==========================================");
  String parada = "Bilbao";
  Integer minutos = 30;
  testAñadirTiempoDescanso(agencia, parada, minutos);
  …
}
```

Apartado c)

```
public SortedMap<String, Duration> getDuracionMinimaPorDestino(TipoViaje tipo) {
      return viajes.stream()
            .filter(v -> v.getTipo().equals(tipo))
            .collect(Collectors.groupingBy(v -> v.getDestino(),
                  TreeMap::new,
                  Collectors.collectingAndThen(
                        Collectors.minBy(Comparator.comparing(Viaje::getDuracion)),
                        o -> o.get().getDuracion())));
}
```

Test

```
private static void testGetDuracionMinimaPorDestino(AgenciaBus agencia, TipoViaje tipo){
      try {
            String msg = String.format("La duración minima por cada destino para
                                        viajes de tipo %s es ", tipo);
            System.out.println(msg);
            SortedMap<String, Duration> res =
                                       agencia.getDuracionMinimaPorDestino(tipo);
            res.entrySet().stream()
                        .forEach(System.out::println);
      } catch (Exception e) {
            System.out.println("Capturada excepcion inesperada " +
                              e.getMessage());
      }
}

public static void main(String[] args) {
  AgenciaBus agencia = FactoriaViajes.leerViajes("Flixbus", "data/viajes.csv");
  System.out.println("Test 4.3 ==========================================");
  testGetDuracionMinimaPorDestino(agencia, TipoViaje.TRANSBORDO);
  testGetDuracionMinimaPorDestino(agencia, TipoViaje.DIRECTO);
  …
}
```

Apartado d)

```
public Map<String, Set<Viaje>> getViajesPorParada(Double precio) {
      Map<String, Set<Viaje>> res = new TreeMap<>();
      for (Viaje v: viajes) {
            if (precio == null || v.getPrecio()<precio) {
                  for (Parada p: v.getTrayecto()) {
                        String clave = p.nombre();
```

```
                        if (res.containsKey(clave)) {
                                res.get(clave).add(v);
                        } else {
                                Set<Viaje> s = new HashSet<>();
                                s.add(v);
                                res.put(clave, s);
                        }
                }
        }
        return res;
}
```

Test

```
private static void testGestViajesPorParadas(AgenciaBus agencia, Double precio) {
        try {
                Map<String, Set<Viaje>> res =
                        agencia.getViajesPorParada(precio);
                String msg = String.format(
                        "Hay %d paradas con viajes con precio inferior a %.2f Euros son ",
                        res.size(), precio);
                System.out.println(msg);
                res.entrySet().stream()
                                .forEach(System.out::println);
        } catch (Exception e) {
                System.out.println("Capturada excepcion inesperada " + e.getMessage());
        }
}

public static void main(String[] args) {
  AgenciaBus agencia = FactoriaViajes.leerViajes("Flixbus", "data/viajes.csv");
  System.out.println("Test 4.d ============================================");
  testGestViajesPorParadas(agencia, 20.);
  testGestViajesPorParadas(agencia, null);
  …
}
```

Apartado e)

```
public SortedMap<String, Double> getPrecioMedioViajesPorParada() {
        Map<String, Set<Viaje>> m = getViajesPorParada(null);
        Function<Entry<String, Set<Viaje>>, Double> fMedia =
                        e -> e.getValue().stream()
                                .mapToDouble(Viaje::getPrecio)
                                .average()
                                .getAsDouble();

        return m.entrySet().stream()
                .filter(e -> e.getValue().size() > 1)
                .collect(Collectors.toMap(
                                Entry::getKey,
                                fMedia,
                                (e1, e2) -> e1,
                                TreeMap::new));
}
```

Test

```
private static void testGetPrecioMedioViajesPorParada(AgenciaBus agencia) {
  try {
       SortedMap<String, Double> res = agencia.getPrecioMedioViajesPorParada();
       String msg = String.format("El precio medio de los viajes por parada es ");
       System.out.println(msg);
       res.entrySet().stream()
                     .forEach(System.out::println);
  } catch (Exception e) {
       System.out.println("Capturada excepcion inesperada " + e.getMessage());
  }
}


public static void main(String[] args) {
  AgenciaBus agencia = FactoriaViajes.leerViajes("Flixbus", "data/viajes.csv");
  System.out.println("Test 4.e==========================================");
  testGetPrecioMedioViajesPorParada(agencia);
  …
}
```