1. Implementar el método **imprimirPosiblesDestinos(String pOrigen):void** de la clase CatalogoOfertas, que imprime en pantalla a que destinos se puede viajar desde el origen indicado.

```
public void imprimirPosiblesDestinos(String pOrigen)
{
	lista.stream().
	filter(o->o.esOrigen(pOrigen)).
	map(Oferta::getDestino).
	forEach(System.out::println);
}

public boolean esOrigen(String pOri) //Clase OFERTA
{
	return this.origen.equals(pOri);
}
public boolean esDestino(String pDest)
{
	return this.destino.equals(pDest);
}
```

2. Implementar el método **imprimirPosiblesDestinos2(String pOrigen):void** igual que el anterior pero sin destinos repetidos.

```
public void imprimirPosiblesDestinos2(String pOrigen)
{
	lista.stream().
	filter(o->o.esOrigen(pOrigen)).
	map(Oferta::getDestino).
	distinct().
	forEach(System.out::println);
}
```

3. Implementar el método ***getOfertasOrdenadasOrigen(): Colección<Oferta>*** de la clase CatalogoOfertas, que devuelve una colección ordenada alfabéticamente por el origen.

```
public List<Oferta> getOfertasOrdenadasOrigen()
{
	return lista.stream().
	sorted(comparing(Oferta::getOrigen)).
	collect(toList());
}
```

4. Implementar el método ***getOfertasOrdenadasOrigenDestino(): Colección<Oferta>*** de la clase CatalogoOfertas, que devuelve una colección ordenada alfabéticamente por el origen y las ofertas con el mismo origen ordenadas por el destino.

```
public List<Oferta> getOfertasOrdenadasOrigenDestino()
{
	return lista.stream().
	sorted(comparing(Oferta::getOrigen).
	thenComparing(Oferta::getDestino)).
	collect(toList());
}
```

5. Implementar el método ***ofertasConEstacion*(String pCiudad): Colección<Oferta>** de la clase CatalogoOfertas que obtiene la colección de ofertas que tiene una estación en la ciudad indicada.

```
public List<Oferta> ofertasConEstacion(String pCiudad)
{
	return lista.stream().
```

```
            filter(o->o.tieneEstacion(pCiudad)).
            collect(toList());
}
public boolean tieneEstacion(String pCiudad) //Clase OFERTA
{
            return listaEstaciones.stream().anyMatch(e->e.esCiudad(pCiudad));
}
public boolean esCiudad(String pC) //Clase ESTACION
{
            return ciudad.equals(pC);
}
```

6. Implementar el método **buscarOfertas(String pOrigen, String pDestino): Colección<Estacion>** de la clase CatalogoOfertas, que obtiene la Colección de ofertas de viaje para ir desde la ciudad origen hasta la ciudad destino ordenadas por precio, y las muestra por pantalla.

```
public void buscarOfertas(String pOri, String pDes)
{
            lista.stream().
            filter(o-> o.esOrigenYDestino(pOri, pDes)).
            sorted(comparing(Oferta::calcularPrecio)).
            forEach(System.out::println);
}
public float calcularPrecio() //Clase OFERTA
{
            return (float) listaEstaciones.stream().
            mapToDouble(e->e.getPrecio()).sum();
}
public float getPrecio() //Clase ESTACION
{
            return costeParada;
}
```

7. Implementar el método **getOfertaMinEstacionesPorOrigen ():Map<String,Oferta>** que devuelve un mapa que almacena para cada origen la oferta que menos estaciones tiene.

```
public Map<String,Oferta> getOfertaMinEstacionesPorOrigen()
{
```

```
        return lista.stream().
        collect(groupingBy(Oferta::getOrigen,
        collectingAndThen(
        minBy(comparing(Oferta::contarEstaciones)),
        p->p.get())));
}
public int contarEstaciones() //Clase OFERTA
{
        return (int)listaEstaciones.stream().count();
}
```

8. Implementar el método **getMinEstacionesPorOrigen():Map<String,int>** igual que el anterior pero con la cantidad de estaciones.

```
public Map<String,Integer> getMinEstacionesPorOrigen()
{
        return lista.stream().
        collect(groupingBy(Oferta::getOrigen,
        collectingAndThen(
        minBy(comparing(Oferta::contarEstaciones)),
        p->p.get().contarEstaciones())));
}
```