OBJETIVOS

- Realizar tratamientos secuenciales con *streams*.
- Trabajar con las interfaces funcionales Comparator, Predicate, Function y Consumer.

TRATAMIENTOS SECUENCIALES CON *STREAMS*

Vamos a redefinir métodos que realizan tratamientos secuenciales sustituyendo los esquemas que utilizan por operaciones con *streams*. En cada caso deberá definir los objetos funcionales que sean necesarios y combinar adecuadamente los métodos de `Stream` para conseguir el objetivo del método.

Para escribir las nuevas versiones de los métodos cree, si no existen ya, las nuevas clases `AlumnoImpl2`, `CentroImpl2`, `DepartamentoImpl2`, `ExpedienteImpl2` y `GradoImpl2`. Estas clases extenderán a las clases correspondientes, y en ellas sólo debe escribir, además de los métodos indicados, los constructores definidos en la clase padre, que únicamente contendrán una llamada al constructor correspondiente.

Antes de redefinir cada método se recomienda leer de nuevo el objetivo del mismo en el boletín correspondiente, recordar el tratamiento utilizado en su resolución y seleccionar las operaciones del tipo `Stream` que permiten realizar este tratamiento.

Recuerde, por último, que en operaciones más complejas es recomendable dividir el problema y crear un método privado que resuelva una parte del mismo.

A continuación se enumeran los métodos que debe redefinir en cada una de las clases, divididos en dos bloques correspondientes a las dos sesiones de laboratorio que se dedican a este boletín.

SESIÓN 1

CentroImpl2:

```
Integer[] getConteosEspacios();
Set<Despacho> getDespachos();
Set<Despacho> getDespachos(Departamento d);
Set<Profesor> getProfesores();
Set<Profesor> getProfesores(Departamento d);
```

DepartamentoImpl2:

```
void borraTutorias();
void borraTutorias(Categoria categoria);
Boolean existeProfesorAsignado(Asignatura a);
Boolean estanTodasAsignaturasAsignadas();
void eliminaAsignacionProfesorado(Asignatura a);
```

SESIÓN 2

AlumnoImpl2:

```
SortedMap<Asignatura, Calificacion> getCalificacionPorAsignatura();
```

CentroImpl2:

```
SortedMap<Profesor, Despacho> getDespachosPorProfesor();
```

DepartamentoImpl2:

```
SortedMap<Profesor, SortedSet<Tutoria>> getTutoriasPorProfesor();
```

ExpedienteImpl2:

```
Double getNotaMedia();
```

GradoImpl2:

```
Double getNumeroTotalCreditos();
Set<Asignatura> getAsignaturas(Integer curso);
Asignatura getAsignatura(String codigo);
Set<Departamento> getDepartamentos();
Set<Profesor> getProfesores();
SortedMap<Asignatura, Double> getCreditosPorAsignatura();
```

ELECCIÓN ENTRE DIFERENTES CLASES DE IMPLEMENTACIÓN

Modifique los métodos de factoría de aquellos tipos que disponen de dos clases de implementación para permitir el uso de ambas. Para ello, utilice el método `setUsarJava8` añadido en el boletín anterior. Por defecto se utilizará la implementación mediante Java 8.

TEST

Para probar los métodos redefinidos, utilice los métodos creacionales del tipo cambiando la clase de implementación de la forma indicada en boletines anteriores.