OBJETIVO

Este documento contiene material de apoyo para el boletín T12 del trabajo práctico. En concreto, el documento incluye una parte de las soluciones del enunciado. El profesor le explicará estas soluciones en la sesión de laboratorio y responderá sus dudas al respecto. El alumno, por su parte, debe añadir el código a su proyecto de curso, y realizar por su cuenta el resto de ejercicios del boletín.

DEFINICIÓN Y USO DE COMPARADORES

Clase `AlumnoImpl` (añada también la cabecera del método a la interfaz Alumno)

```
public SortedSet<Asignatura> getAsignaturasOrdenadasPorCurso() {
	Comparator<Asignatura> cmp = Comparator.comparing(Asignatura::getCurso)
		.reversed()
		.thenComparing(Comparator.naturalOrder());
	SortedSet<Asignatura> res = new TreeSet<Asignatura>(cmp);
	res.addAll(asignaturas);
	return res;
}
```

Clase `CentroImpl` (añada también la cabecera del método a la interfaz Centro)

```
public SortedSet<Espacio> getEspaciosOrdenadosPorCapacidad() {
	Comparator<Espacio> cmp = Comparator.comparing(Espacio::getCapacidad)
		.reversed()
		.thenComparing(Comparator.naturalOrder());
	SortedSet<Espacio> res = new TreeSet<Espacio>(cmp);
	res.addAll(espacios);
	return res;
}
```

Clase `GradoImpl` (añada también la cabecera del método a la interfaz Grado)

```
public SortedSet<Departamento> getDepartamentosOrdenadosPorAsignaturas() {
	Comparator<Departamento> cmp =
		Comparator.comparing((Departamento d)->d.getAsignaturas().size())
		.reversed()
		.thenComparing(Comparator.naturalOrder());
	SortedSet<Departamento> res = new TreeSet<Departamento>(cmp);
	res.addAll(getDepartamentos());
	return res;
}
```

TRATAMIENTOS SECUENCIALES CON *STREAMS*

Clase `CentroImpl2`

```
public class CentroImpl2 extends CentroImpl {

	public CentroImpl2(String nombre, String direccion, Integer numPlantas,
			Integer numSotanos) {
		super(nombre, direccion, numPlantas, numSotanos);
	}

	public Espacio getEspacioMayorCapacidad() {
		Optional<Espacio> res = getEspacios().stream()
			.max(Comparator.comparing(Espacio::getCapacidad));

		if (!res.isPresent()) {
			throw new ExcepcionCentroOperacionNoPermitida(
				"El centro no dispone aún de espacios.");
		}

		return res.get();
	}
}
```

Clase `DepartamentoImpl` (añada también la cabecera del método a la interfaz Departamento)

```
public Profesor getProfesorMaximaDedicacionMediaPorAsignatura() {

	Comparator<Profesor> cmp = Comparator
		.comparing(profesor ->
		profesor.getDedicacionTotal() / profesor.getAsignaturas().size());

	return getProfesores().stream()
		.filter(profesor -> !profesor.getAsignaturas().isEmpty())
		.max(cmp).get();
}
```

ELECCIÓN ENTRE DIFERENTES CLASES DE IMPLEMENTACIÓN

Clase `Grados`

```
private static Boolean usarJava8 = true;

public static void setUsarJava8(Boolean b) {
	usarJava8 = b;
}

public static Centro createCentro(String nombre, String direccion,
		Integer numPlantas, Integer numSotanos) {
	Centro res = null;
	if (usarJava8) {
		res = new CentroImpl2(nombre, direccion, numPlantas, numSotanos);
	} else {
		res = new CentroImpl(nombre, direccion, numPlantas, numSotanos);
	}
	actualizaPoblacionales(res);
	return res;
}

// El resto de métodos createAlumno y createCentro deben ser modificados de
forma similar
```