OBJETIVO

Este documento contiene material de apoyo para el boletín T13 del trabajo práctico. En concreto, el documento incluye las soluciones para algunos de los métodos pedidos por el 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.

SESIÓN 1

Clase `CentroImpl2`

```
public Integer[] getConteosEspacios() {
	Integer[] conteos = {0,0,0,0,0};
	getEspacios().stream().forEach(e -> conteos[e.getTipo().ordinal()]++);
	return conteos;
}

public Set<Profesor> getProfesores() {
	return getDespachos().stream()
		.flatMap(d -> d.getProfesores().stream())
		.collect(Collectors.toSet());
}

public Set<Profesor> getProfesores(Departamento dep) {
	return getDespachos().stream()
		.flatMap(d -> d.getProfesores().stream())
		.filter(p -> p.getDepartamento().equals(dep))
		.collect(Collectors.toSet());

	/* Solución alternativa:
	return getProfesores().stream()
		.filter(p -> p.getDepartamento().equals(dep))
		.collect(Collectors.toSet());
	*/
}
```

Clase `DepartamentoImpl2`

```
public void borraTutorias() {
	getProfesores().stream()
		.forEach(Profesor::borraTutorias);
}

public void borraTutorias(Categoria categoria) {
	getProfesores().stream()
		.filter(p->p.getCategoria().equals(categoria))
		.forEach(Profesor::borraTutorias);
}
```

SESIÓN 2

Clase `CentroImpl2`

```
public SortedMap<String, Despacho> getDespachosPorProfesor() {
	return new TreeMap<String, Despacho>(getProfesores().stream()
		.collect(Collectors.toMap(p->p.toString(), p->buscaDespacho(p))));
	}

private Despacho buscaDespacho(Profesor p) {
	return getDespachos().stream()
		.filter(d -> d.getProfesores().contains(p))
		.findFirst().get();
}
```

Clase `DepartamentoImpl2`

```
public SortedMap<String, SortedSet<Tutoria>> getTutoriasPorProfesor() {
	return new TreeMap<String, SortedSet<Tutoria>>(
		getProfesores().stream()
			.collect(Collectors.toMap(p->p.toString(),
						p->p.getTutorias()))
	);
}
```

Clase `ExpedienteImpl2`

```
public Double getNotaMedia() {
	return getNotas().stream()
		.filter(n->n.getValor()>=5)
		.mapToDouble(n->n.getValor())
		.average()
		.orElse(0.0);
}
```

Clase `GradoImpl2`

```
public Double getNumeroTotalCreditos() {
	return getAsignaturasObligatorias().stream()
		.mapToDouble(a->a.getCreditos())
		.sum()+getNumeroMinimoCreditosOptativas();
}

public Set<Asignatura> getAsignaturas(Integer curso) {
	return Stream.concat(getAsignaturasObligatorias().stream(),
				getAsignaturasOptativas().stream())
		.filter(a->a.getCurso().equals(curso))
		.collect(Collectors.toSet());
}
```