OBJETIVO

Este documento contiene material de apoyo para el boletín T10 del trabajo práctico. En concreto, el documento incluye algunos de los constructores que se piden en el enunciado, así como las modificaciones a realizar a NotaImpl para obtener NotaInmutableImpl. El profesor las explicará en la sesión de laboratorio y responderá a 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.

CONSTRUCTORES A PARTIR DE STRING

Clase `PersonaImpl`

```
	public PersonaImpl(String persona) {
		// Ejemplo de formato de la cadena de entrada:
		// 12345678H,Juan,López García,20/01/1998,juan@acmemail.com

		String[] campos = persona.split(",");
		if (campos.length != 5) {
			throw new IllegalArgumentException(
				"El formato de la cadena de entrada no es correcto.");
		}

		String dni = campos[0].trim();
		String email = campos[4].trim();
		LocalDate fechaNacimiento = LocalDate.parse(campos[3].trim(),
				DateTimeFormatter.ofPattern("d/M/y"));

		checkDni(dni);
		checkFechaNacimiento(fechaNacimiento);
		checkEmail(email);

		this.dni = dni;
		this.nombre = campos[1].trim();
		this.apellidos = campos[2].trim();
		this.fechaNacimiento = fechaNacimiento;
		this.email = email;
	}
```

Clase `AlumnoImpl`

```
	public AlumnoImpl(String alumno) {
		// Ejemplo de formato de la cadena de entrada:
		// 12345678H,Juan,López García,20/01/1998,juan@alum.us.es
		super(alumno);
		checkEmailUniversidad(getEmail());
		this.asignaturas = new HashSet<Asignatura>();
		this.expediente = new ExpedienteImpl();
	}
```

Clase `AsignaturaImpl`

```
	public AsignaturaImpl(String asignatura) {
		// Ejemplo de formato de la cadena de entrada:
		// Fundamentos de Programación#1234567# 12.0# ANUAL# 1

		String[] campos = asignatura.split("#");
		if (campos.length != 5) {
			throw new IllegalArgumentException(
				"El formato de la cadena de entrada no es correcto.");
		}

		String codigo = campos[1].trim();
		Double creditos = new Double(campos[2].trim());
		Integer curso = new Integer(campos[4].trim());

		checkCodigo(codigo);
		checkCreditos(creditos);
		checkCurso(curso);

		this.nombre = campos[0].trim();
		this.codigo = codigo;
		this.creditos = creditos;
		this.tipo = TipoAsignatura.valueOf(campos[3].trim());
		this.curso = curso;
		this.departamento = null;
	}
```

Clase `NotaImpl`

```
	public NotaImpl(String nota) {
		// Ejemplo de formato de la cadena de entrada:
		// Fund. de Prog. #1234567#12.0#ANUAL#1;2014;PRIMERA;10.0;true

		String[] campos = nota.split(";");
		if (campos.length != 5) {
			throw new IllegalArgumentException(
				"El formato de la cadena de entrada no es correcto.");
		}

		Double valor = new Double(campos[3].trim());
		Boolean mencionHonor = new Boolean(campos[4].trim());

		checkNota(valor);
		checkMencionHonor(mencionHonor, valor);

		this.asignatura = new AsignaturaImpl(campos[0].trim());
		this.curso = new Integer(campos[1].trim());
		this.convocatoria = Convocatoria.valueOf(campos[2].trim());
		this.valor = valor;
		this.mencionHonor = mencionHonor;
	}
```

TIPO NOTA INMUTABLE

Clase `NotaInmutableImpl`

```
// Se añade final a la cabecera de la clase
public final class NotaInmutableImpl implements Nota {
	// Se añade final en la declaración de atributos
	private final Asignatura asignatura;
	private final Integer curso;
	private final Convocatoria convocatoria;
	private final Double valor;
	private final Boolean mencionHonor;

	public NotaInmutableImpl(Asignatura asignatura, Integer curso,
		  Convocatoria convocatoria, Double valor, Boolean mencionHonor) {
		checkNota(valor);
		checkMencionHonor(mencionHonor, valor);
		// Se almacena una copia de asignatura, al ser un tipo mutable
		this.asignatura = copia(asignatura);
		this.curso = curso;
		this.convocatoria = convocatoria;
		this.valor = valor;
		this.mencionHonor = mencionHonor;
	}

	private Asignatura copia(Asignatura asignatura) {
		return new AsignaturaImpl(asignatura.getNombre(),
			asignatura.getCodigo(), asignatura.getCreditos(),
			asignatura.getTipo(), asignatura.getCurso(),
			asignatura.getDepartamento());
	}

	public Asignatura getAsignatura() {
	// Se devuelve una copia de asignatura, al ser un tipo mutable
		return copia(asignatura);
	}

	// El resto de la clase permanece inalterada

}
```

TESTS

Clase `TestPersona` (añada casos de prueba similares al resto de sus clases de test)

```
	// Hay que invocar este caso de prueba en el main
	private static void testConstructorString() {
		List<Persona> personas = Grados.leeFichero("res/personas.txt",
				s->new PersonaImpl(s));
		for (Persona p:personas) {
			mostrarPersona(p);
		}
	}
```