OBJETIVOS

- Uso de colecciones.
- Implementación de tratamientos secuenciales.

IMPLEMENTACIÓN DE PROPIEDADES PENDIENTES

En los tipos desarrollados hasta la fecha se han dejado sin implementar una serie de propiedades que requerían para su cálculo el uso de tratamientos secuenciales. Ahora es el momento de implementar estas propiedades. Para ello, vaya al tipo correspondiente y realice las modificaciones oportunas tanto en la interfaz como en la clase.

Tipo **Asignatura**:

- Implemente el método `getAcronimo()` descrito en el boletín T1. Recuerde que el acrónimo se construye tomando las letras mayúsculas que aparecen en el nombre de una asignatura. Por ejemplo, el acrónimo de "Fundamentos de Programación" es "FP".

Tipo **Alumno**:

- Implemente la propiedad derivada **Curso** descrita en el boletín T5. Recuerde que el curso de un alumno es el mayor de los cursos de las asignaturas en que está matriculado, o 0 si no está matriculado en ninguna asignatura.
- Complete la representación como cadena añadiendo el curso del alumno.

Tipo **Expediente**:

- Implemente la propiedad derivada **Nota media** descrita en el boletín T7. Recuerde que la nota media de un expediente es la media de los valores numéricos de todas las notas del expediente que tienen un valor mayor o igual a 5. Si el expediente no tiene ninguna nota mayor o igual a 5, la nota media será 0.0.
- Añada a este tipo una nueva restricción: un expediente no puede contener notas para más de dos convocatorias de una misma asignatura y curso. Si no se cumple esta restricción, lance la excepción `ExcepcionExpedienteOperacionNoPermitida`.

Tipo **Profesor**:

- Añada la propiedad derivada **Dedicación total**, de tipo `Double`, que representa el número total de créditos impartidos por el profesor, y que se calcula sumando los créditos que imparte el profesor en cada una de sus asignaturas.
- Añada a este tipo una nueva restricción: un profesor puede impartir un máximo de 24 créditos (valor que puede cambiar en el futuro). Deberá tenerlo en cuenta cuando añada nuevas asignaturas a un profesor o cuando modifique la dedicación de éste a una asignatura. Si no se cumple esta restricción, lance la excepción `ExcepcionProfesorOperacionNoPermitida`.

NUEVAS OPERACIONES SOBRE EL TIPO CENTRO

Añada las siguientes operaciones al tipo Centro:

- `Integer[] getConteosEspacios()`. Devuelve un array de 5 elementos de tipo `Integer` que representan el número de espacios de tipo aula de teoría, laboratorio, seminario, aula de examen u otro tipo, respectivamente, que hay en el centro.
- `Set<Despacho> getDespachos()`. Devuelve un conjunto con todos los espacios de tipo Despacho que hay en el centro.
- `Set<Despacho> getDespachos(Departamento d)`. Devuelve un conjunto con todos los despachos del centro donde hay al menos un profesor del departamento `d`.
- `Set<Profesor> getProfesores()`. Devuelve el conjunto de todos los profesores que tienen un despacho en el centro.
- `Set<Profesor> getProfesores(Departamento d)`. Devuelve el conjunto de todos los profesores que pertenecen al departamento `d` y tienen un despacho en el centro.
- `Espacio getEspacioMayorCapacidad()`. Devuelve el espacio con mayor capacidad del centro. Si no hay espacios, lance la excepción `ExcepcionCentroOperacionNoPermitida`.

NUEVAS OPERACIONES SOBRE EL TIPO DEPARTAMENTO

Añada las siguientes operaciones al tipo Departamento:

- `void borraTutorias()`. Elimina las tutorías de todos los profesores del departamento.
- `void borraTutorias(Categoria c)`. Elimina las tutorías de todos los profesores del departamento cuya categoría es `c`.
- `Boolean existeProfesorAsignado(Asignatura a)`. Devuelve `true` si existe al menos un profesor asignado a la asignatura `a`, es decir, que imparte algún crédito en ella, y `false` en caso contrario.
- `Boolean estanTodasAsignaturasAsignadas()`. Devuelve `true` si todas las asignaturas tienen asignado al menos un profesor, y `false` en caso contrario.
- `void eliminaAsignacionProfesorado(Asignatura a)`. Elimina la asignatura `a` de todos los profesores del departamento que la están impartiendo.

TIPO GRADO

Un grado representa una titulación impartida por la Universidad. Un grado se imparte en un centro y consta de unas asignaturas obligatorias y otras optativas, que son impartidas por profesores de los departamentos con docencia en el grado.

Defina el tipo Grado, siguiendo la plantilla que se le proporciona.

Tipo **Grado**.

Propiedades:

- **Nombre**. Propiedad básica, de tipo `String`. Consultable. Sin restricciones.
- **Asignaturas obligatorias**. Propiedad básica, de tipo `Set<Asignatura>`. Consultable. Sin restricciones.
- **Asignaturas optativas**. Propiedad básica, de tipo `Set<Asignatura>`. Consultable.
- **Número mínimo de créditos de optativas**. Propiedad básica, de tipo `Double`. Representa el número de créditos de asignaturas optativas que debe cursar un alumno del grado como mínimo. Consultable.
- **Número total de créditos**. Propiedad derivada, de tipo `Double`. Representa el número total de créditos que debe cursar un alumno para obtener el título. Este número se calcula sumando

los créditos de todas las asignaturas obligatorias y el número mínimo de créditos de asignaturas optativas que debe cursar. Consultable.

- **Departamentos**. Propiedad derivada, de tipo `Set<Departamento>`. Representa los departamentos con docencia en el grado, que son aquellos que imparten alguna asignatura en el mismo. Consultable.
- **Profesores**. Propiedad derivada, de tipo `Set<Profesor>`. Representa los profesores que pertenecen a cualquiera de los departamentos con docencia en el grado. Consultable.

Restricciones:

- Todas las asignaturas optativas de un grado deben tener el mismo número de créditos. Si no se cumple esta restricción, lance la excepción `ExcepcionGradoNoValido`.
- El número mínimo de créditos de asignaturas optativas que debe cursar un alumno debe estar comprendido entre cero y el número total de créditos de asignaturas optativas del grado. Si no se cumple esta restricción, lance la excepción `ExcepcionGradoNoValido`.

Operaciones:

- `Set<Asignatura> getAsignaturas(Integer curso)`. Devuelve un conjunto con todas las asignaturas del grado, tanto obligatorias como optativas, que se imparten en el curso `curso`.
- `Asignatura getAsignatura(String codigo)`. Devuelve la asignatura del grado cuyo código es `codigo`, o `null` si no existe ninguna asignatura con dicho código.
- Constructor: recibe como parámetros todas las propiedades básicas del tipo.
- Criterio de igualdad: dos grados son iguales si tienen el mismo nombre.
- Criterio de orden natural: los grados se ordenan por nombre.
- Representación como cadena: el nombre del grado. Por ejemplo, "Grado en Ingeniería Informática – Ingeniería del Software".

TEST

Pruebe el tipo Grado y los nuevos métodos implementados siguiendo la metodología utilizada en boletines anteriores. Piense en los casos de prueba que debe emplear antes de implementar los test.