OBJETIVOS

- Uso de colecciones.
- Trabajo con propiedades modeladas mediante agregados de datos.

GESTIÓN DE LAS ASIGNATURAS DE UN PROFESOR

Vamos a implementar la gestión de las asignaturas impartidas por un profesor. Para ello, añada al tipo Profesor las siguientes propiedades, restricciones y operaciones:

Propiedades:

- **Asignaturas**. Propiedad básica, de tipo `List<Asignatura>`. Consultable. Representa las asignaturas que imparte un profesor.
- **Créditos**. Propiedad básica, de tipo `List<Double>`. Consultable. Representa los créditos impartidos por un profesor en cada asignatura. Cada elemento de la lista representa el número de créditos que imparte el profesor en la asignatura que ocupa la misma posición en la lista de asignaturas.
- **Dedicación total**. Propiedad derivada, de tipo `Double`. Consultable. Representa el número total de créditos que imparte un profesor. **Esta propiedad no se implementa de momento**.

Restricciones:

- Un profesor no puede impartir una asignatura de otro departamento.
- El número de créditos impartidos por un profesor en una asignatura debe ser mayor que 0 y menor o igual que el número de créditos de dicha asignatura.

Operaciones:

- `void imparteAsignatura(Asignatura asig, Double dedicacion)`. Añade la asignatura `asig` a las asignaturas que imparte el profesor, siendo `dedicacion` el número de créditos que imparte el profesor en dicha asignatura. Si la asignatura ya era impartida por el profesor, se actualiza la dedicación. Vigile que se respeten las restricciones, y en caso contrario lance la excepción `ExcepcionProfesorOperacionNoPermitida`.
- `Double dedicacionAsignatura(Asignatura asig)`. Devuelve el número de créditos que imparte el profesor en la asignatura `asig`. Si el profesor no imparte la asignatura, devuelve 0.0.
- `void eliminaAsignatura(Asignatura asig)`. Elimina la asignatura `asig` de las asignaturas que imparte el profesor. Si el profesor no imparte la asignatura, la operación no tiene efecto.
- El constructor debe inicializar las asignaturas y los créditos con sendas listas vacías.

DEFINICIÓN DEL TIPO EXPEDIENTE

Un expediente académico contiene una colección de notas. Defina el tipo Expediente, siguiendo la plantilla que se le proporciona.

Tipo **Expediente**.

Propiedades:

- **Notas**. Propiedad básica, de tipo `List<Nota>`. Consultable. Representa las notas que forman un expediente.
- **Nota media**. Propiedad derivada, de tipo `Double`. Consultable. 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 está vacío, la nota media será 0.0. **Esta propiedad no se implementa de momento**.

Operaciones:

- `void nuevaNota(Nota n)`. Añade la nota `n` al expediente. Si en el expediente ya existe una nota para la misma asignatura, convocatoria y curso académico (es decir, otra nota igual según el criterio de igualdad del tipo Nota), se elimina la nota antigua y se añade la nueva. Las notas se añaden siempre al final de la lista.
- Constructor: no recibe parámetros. Las notas se inicializan con una lista vacía.
- Criterio de igualdad: dos expedientes son iguales si tienen las mismas notas y están en el mismo orden.
- Criterio de orden natural: no existe criterio de orden natural para los expedientes.
- Representación como cadena: las notas que forman el expediente (utilice la representación como cadena de la propiedad 'Notas'). Por ejemplo, "[(0000230) Fundamentos de Programación, 2014-15, PRIMERA, 7.5, NOTABLE, (2050009) Estructura de Computadores, 2014-15, PRIMERA, 6.3, APROBADO]".

GESTIÓN DEL EXPEDIENTE DE UN ALUMNO

Una vez que se ha definido el tipo Expediente, se puede implementar la relación entre un alumno y su expediente. Para ello añada al tipo Alumno una nueva propiedad:

- **Expediente**. Propiedad básica, de tipo `Expediente`. Consultable. Sin restricciones.

Realice los cambios oportunos en la interfaz `Alumno` y en la clase `AlumnoImpl` para incorporar esta nueva propiedad. En la clase, tenga en cuenta que el expediente estará vacío cuando se construya un alumno.

Para añadir notas al expediente de un alumno escriba los siguientes métodos:

- `void evaluaAlumno(Asignatura a, Integer curso, Convocatoria convocatoria, Double nota, Boolean mencionHonor)`. Añade al expediente la nota cuyas propiedades recibe como parámetros. Si el alumno no está matriculado en la asignatura `a`, el método lanzará la excepción `ExcepcionAlumnoOperacionNoPermitida`.
- `void evaluaAlumno(Asignatura a, Integer curso, Convocatoria convocatoria, Double nota)`. Añade al expediente la nota cuyas propiedades recibe como parámetros. Si el alumno no está matriculado en la asignatura `a`, el método lanzará la excepción `ExcepcionAlumnoOperacionNoPermitida`.

TEST

Pruebe los métodos de gestión de las asignaturas de un profesor. Realice pruebas en las que se añadan y eliminen asignaturas a un profesor, y compruebe que se actualizan correctamente las propiedades del profesor.

Pruebe todos los métodos del tipo Expediente, siguiendo la metodología empleada con los tipos anteriores. Pruebe los métodos de gestión del expediente de un alumno. Realice pruebas en las que se añadan notas al expediente de un alumno, y compruebe que se actualizan correctamente las propiedades del alumno.