OBJETIVOS

- Uso de colecciones.
- Implementación de relaciones bidireccionales entre tipos.

DEFINICIÓN DEL TIPO CENTRO

En este boletín trabajaremos con los tipos Centro y Departamento. Para ello crearemos los nuevos tipos a partir de sus respectivas definiciones.

Defina en primer lugar el tipo Centro, siguiendo la plantilla que se le proporciona.

Tipo **Centro**.

Propiedades:

- **Nombre**. Propiedad básica, de tipo `String`. Consultable.
- **Dirección**. Propiedad básica, de tipo `String`. Consultable.
- **Número de plantas**. Propiedad básica, de tipo `Integer`. Representa el número de plantas sobre rasante que tiene el centro. Consultable.
- **Número de sótanos**. Propiedad básica, de tipo `Integer`. Representa el número de sótanos o plantas subterráneas que tiene el centro. Consultable.
- **Espacios**. Propiedad básica, de tipo `Set<Espacio>`. Consultable. Representa los espacios ubicados en un centro.

Restricciones:

- Un centro debe tener al menos una planta. Si no se cumple esta restricción, lance la excepción `ExcepcionCentroNoValido`.
- Un centro puede tener cero o más sótanos. Si no se cumple esta restricción, lance la excepción `ExcepcionCentroNoValido`.

Operaciones:

- `void nuevoEspacio(Espacio e)`. Añade el espacio `e` al centro, siempre que la planta del espacio sea un número de planta válido en el centro. Para ello debe estar comprendida en el intervalo [`-s`, `p-1`], siendo `p` el número de plantas del centro y `s` el número de sótanos. En caso contrario, lance la excepción `ExcepcionCentroOperacionNoPermitida`.
- `void eliminaEspacio(Espacio e)`. Elimina el espacio `e` del centro. Si el espacio no pertenece al centro, la operación no tiene efecto.
- Constructor: recibe como parámetros el nombre del centro, su dirección, el número de plantas y el número de sótanos. Los espacios se inicializan con un conjunto vacío.
- Criterio de igualdad: dos centros son iguales si tienen el mismo nombre.
- Criterio de orden natural: los centros se ordenan por nombre.
- Representación como cadena: el nombre del centro. Por ejemplo, “Escuela Técnica Superior de Ingeniería Informática”.

DEFINICIÓN DEL TIPO DEPARTAMENTO

Defina el tipo Departamento, siguiendo la plantilla que se le proporciona. Este tipo está relacionado con otros dos tipos vistos anteriormente, los profesores y las asignaturas. Así, un departamento está formado por un conjunto de profesores e imparte un conjunto de asignaturas. Por eso, trabajaremos en paralelo con los tres tipos.

Tipo **Departamento**.

Propiedades:

- **Nombre**. Propiedad básica, de tipo `String`. Consultable.
- **Profesores**. Propiedad básica, de tipo `Set<Profesor>`. Consultable. Representa los profesores que pertenecen a un departamento.
- **Asignaturas**. Propiedad básica, de tipo `Set<Asignatura>`. Consultable. Representa las asignaturas impartidas por un departamento.

Operaciones:

- `void nuevoProfesor(Profesor prof)`. Añade el profesor `prof` al departamento. Este método debe actualizar la propiedad departamento del profesor `prof`. De momento no implemente esta parte (hágalo más adelante cuando se le indique).
- `void eliminaProfesor(Profesor prof)`. Elimina el profesor `prof` del departamento. Si el profesor no pertenece al departamento, la operación no tiene efecto. Este método debe actualizar la propiedad departamento del profesor `prof`. De momento no implemente esta parte.
- `void nuevaAsignatura(Asignatura asig)`. Añade la asignatura `asig` al departamento. Este método debe actualizar la propiedad departamento de la asignatura `asig`. De momento no implemente esta parte.
- `void eliminaAsignatura(Asignatura asig)`. Elimina la asignatura `asig` del departamento. Si la asignatura no es impartida por el departamento, la operación no tiene efecto. Este método debe actualizar la propiedad departamento de la asignatura `asig`. De momento no implemente esta parte.
- Constructor: recibe como parámetro el nombre del departamento. Los profesores y las asignaturas se inicializan con sendos conjuntos vacíos.
- Criterio de igualdad: dos departamentos son iguales si tienen el mismo nombre.
- Criterio de orden natural: los departamentos se ordenan por nombre.
- Representación como cadena: el nombre del departamento. Por ejemplo, "Lenguajes y Sistemas Informáticos".

IMPLEMENTACIÓN DE LA RELACIÓN ENTRE DEPARTAMENTO Y ASIGNATURA

En la definición del tipo Asignatura aparecía una propiedad *departamento* que no se implementó en su momento. Para hacerlo hay que tener en cuenta que un cambio en el departamento de una asignatura debe reflejarse en la propiedad asignaturas del departamento, y viceversa, un cambio en las asignaturas de un departamento debe reflejarse en la propiedad departamento de la asignatura.

Una vez que tenemos clara esta relación de dependencia entre departamento y asignatura, vamos a implementarla. Para ello realice los siguientes pasos[1]:

[1] Puede ver un vídeo con la explicación en https://www.youtube.com/watch?v=wLVAjSHA_8A

1. Añada la propiedad departamento en la interfaz.
2. Añada la propiedad departamento en la clase.
3. Inicialice la nueva propiedad con un valor null en el constructor existente.
4. Añada a la clase un segundo constructor con un parámetro adicional para el departamento. Utilice el método `setDepartamento()` para inicializar la propiedad departamento, en lugar de hacerlo con una asignación.
5. Implemente el método `getDepartamento()`.
6. Implemente el método `setDepartamento()`. Tenga en cuenta que una asignatura sólo puede ser impartida por un departamento. Por tanto, antes de asignar un nuevo departamento a la asignatura hay que comprobar si la asignatura pertenece a otro departamento, y en ese caso eliminarla del departamento antiguo antes de añadirla al nuevo.

Si en su momento colocó los `TODO`, le resultará fácil localizar los lugares del código donde debe realizar los cambios.

Ahora que ya tiene el método `setDepartamento()`, vuelva sobre los métodos `nuevaAsignatura()` y `eliminaAsignatura()` del tipo Departamento. Añada a ambos métodos el código necesario para actualizar el departamento de la asignatura que se ha añadido o se ha eliminado, respectivamente.

IMPLEMENTACIÓN DE LA RELACIÓN ENTRE DEPARTAMENTO Y PROFESOR

La relación entre departamento y profesor es similar a la que hemos visto entre departamento y asignatura. Un cambio en el departamento de un profesor debe reflejarse en la propiedad profesores del departamento, y viceversa, un cambio en los profesores de un departamento debe reflejarse en la propiedad departamento del profesor.

Para implementar esta relación, añada al tipo Profesor una propiedad *departamento*, de tipo `Departamento`. Esta propiedad es consultable y modificable, y no tiene restricciones. Realice los cambios adecuados en la interfaz y en la clase. Inicialice el departamento a null en el constructor existente, y añada un segundo constructor con un parámetro más para el departamento.

En el método `setDepartamento()` tenga en cuenta que un profesor sólo puede pertenecer a un departamento. Por tanto, antes de asignar un nuevo departamento al profesor hay que comprobar si el profesor pertenece a otro departamento, y en ese caso eliminarlo del departamento antiguo antes de añadirlo al nuevo.

Una vez creado el método `setDepartamento()`, vuelva sobre los métodos `nuevoProfesor()` y `eliminaProfesor()` del tipo Departamento. Añada a ambos métodos el código necesario para actualizar el departamento del profesor que se ha añadido o se ha eliminado, respectivamente.

TEST

Pruebe todos los métodos de los tipos Centro y Departamento, siguiendo la metodología empleada con los tipos anteriores. Preste especial atención a las relaciones bidireccionales entre departamento y asignatura y entre departamento y profesor. Realice pruebas en las que se añadan y eliminen asignaturas y profesores a un departamento, y compruebe que se actualizan correctamente las propiedades relacionadas de los tres tipos.