El ambulatorio de Durango está inmerso en un proceso de informatización de los datos de las consultas que atiende. Para diseñar este proceso, el ambulatorio debe aportar la información sobre sus pacientes indicando para cada uno de ellos, su nombre y apellidos, edad, número de la Seguridad Social, provincia en la que vive y una lista con los síntomas que presentaba cuando ha acudido a la consulta. Un síntoma se caracteriza por su nombre y un código numérico.

Objetivo:

Determinar el número de pacientes que presentan un síntoma determinado.

GUIA DE BOOCH:

Extraer y agrupar las partes que hacen referencia a cada entidad involucrada en el proceso que se ha enunciado. Estas entidades (*sustantivos propios o comunes*) constituyen las clases que debéis definir en la aplicación. Una vez identificadas, determinar sus atributos, que representan las propiedades que describen a cualquier objeto de esa clase.

Definir las clases necesarias para recoger la información de todos los pacientes del centro.

Paciente nombre: String apellidos: String edad: int numSS: String provincia: String lSintomas: ListaSintomas	**ListaPacientes** (sería el Ambulatorio) lista: Contenedor<Paciente>
Sintoma nombre: String codigo: int	**ListaSintomas** lista: Contenedor<Sintoma>

Criterios a considerar:

- Si se tiene una clase con varios atributos, alguno de los cuales es una colección de objetos del mismo tipo (p.e., listaSintomas), se debe definir una nueva clase que represente dicha información (lSintomas: ListaSintomas).
- Cada clase que represente una colección de objetos debe incluir
    - Un sin parámetros, que inicializa la lista vacía. Es posible sobrecargar el constructor, siempre que no se comprometa la ocultación de información.
    - Métodos para añadir elementos a la lista.
    - Métodos para comprobar si un determinado objeto está en la lista
    - Un método que devuelva el iterador para recorrerla secuenciamente.

Además de estos métodos, puede resultar interesante definir otros para eliminar un elemento de la lista, etc.

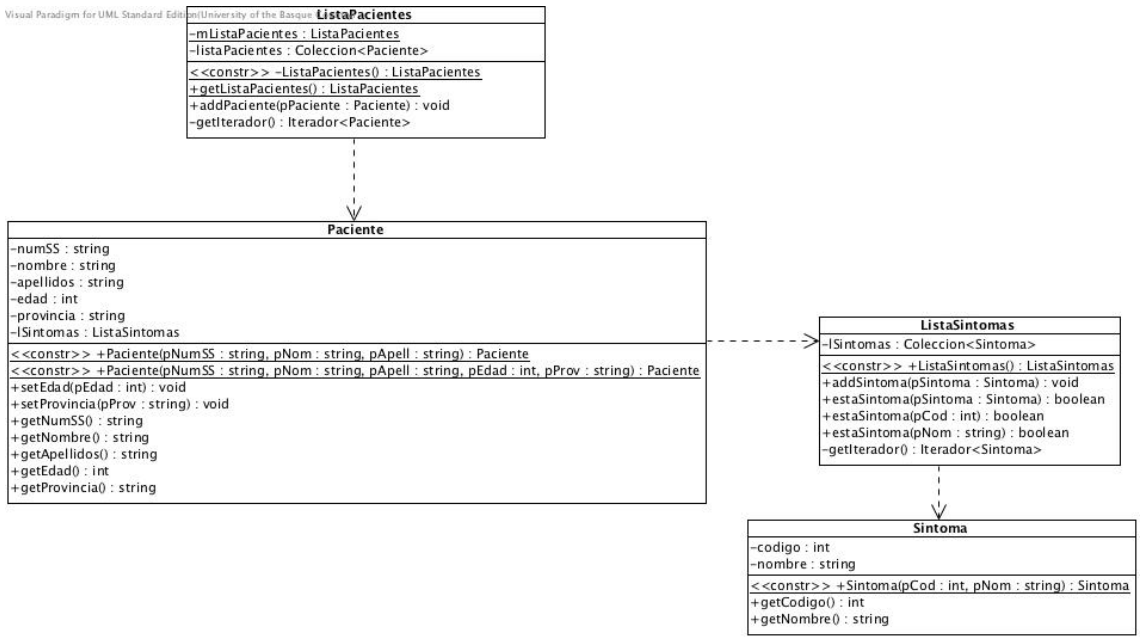
TODOS LOS ATRIBUTOS DE LAS CLASES SE DEFINEN INICIALMENTE PRIVADOS, MIENTRAS NO SEA NECESARIO QUE OTRAS CLASES ACCEDAN A ELLOS.

PROHIBIDO!! definir un método que devuelva el atributo que contiene la colección de objetos, ya que esto compromete la ocultación de información

- Los atributos que identifican de forma unívoca una instacia, por ejemplo el nombre y el código de un síntoma o el nombre, apellidos y código de la seguridad del paciente, deben inicializarse al construir la instancia. Es decir deben ser parámetros de la constructora de la clase. No se definirán métodos para modificar dichos valores. Las clases con este tipo de atributos no pueden tener un constructor sin parámetros.
- Pueden definirse métodos get, de los atributos que sean de tipo primitivo (int, float, …) y cadenas de caracteres (string). PROHIBIDO definir get en atributos para los que se comprometa la ocultación. Este tipo de atributos sólo podrán manipularse a través de los métodos definidos en su clase.
- En las clases que sólo van a tener una instancia, se aplicará el patrón SINGLETON.

Paciente ~~Paciente()~~ Paciente(pNombre: String, pApellidos: String,pNumSS: String) Paciente(pNombre: String, pApellidos: String, pEdad: int, pNumSS: String, pProvincia: String)	**ListaPacientes (Singleton)** ListaPacientes() getListaPacientes():ListaPacientes addPaciente(pPac: Paciente) getIterador(): Iterador
Sintoma ~~Sintoma()~~ Sintoma(pNombre:String,pCodigo:int)	**ListaSintomas** ListaSintomas() addSintoma(pSint: Sintoma) getIterador(): Iterador

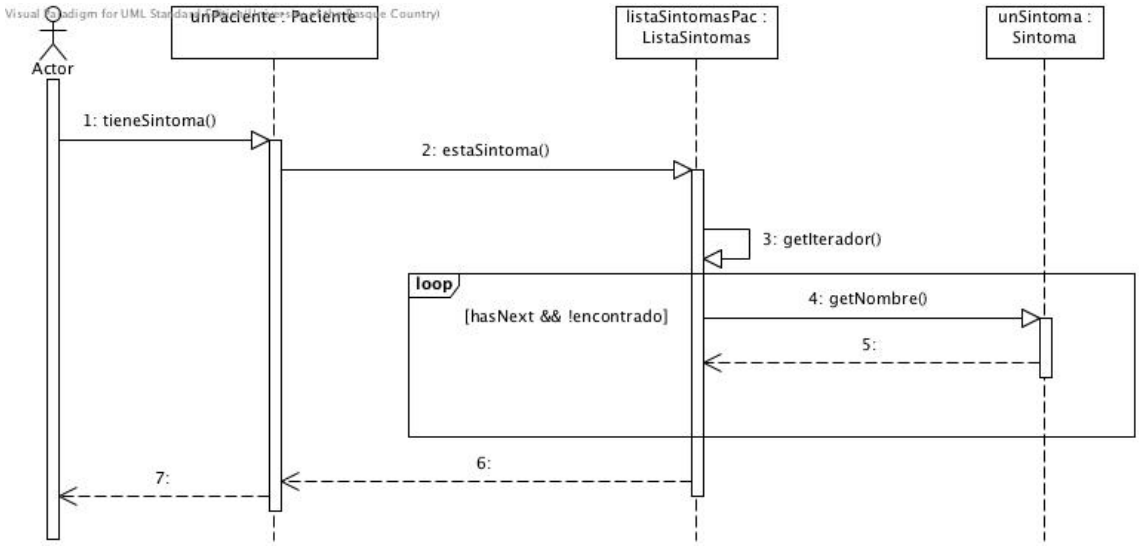
Patrón SINGLETON



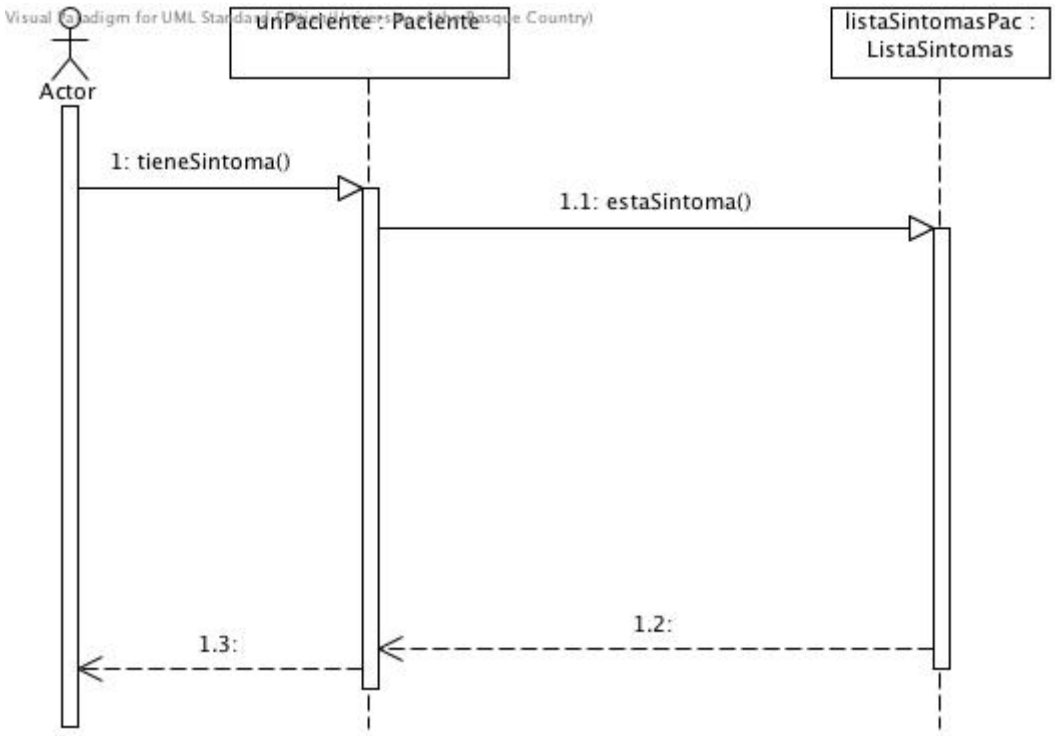
GUIA DE ELLIS:

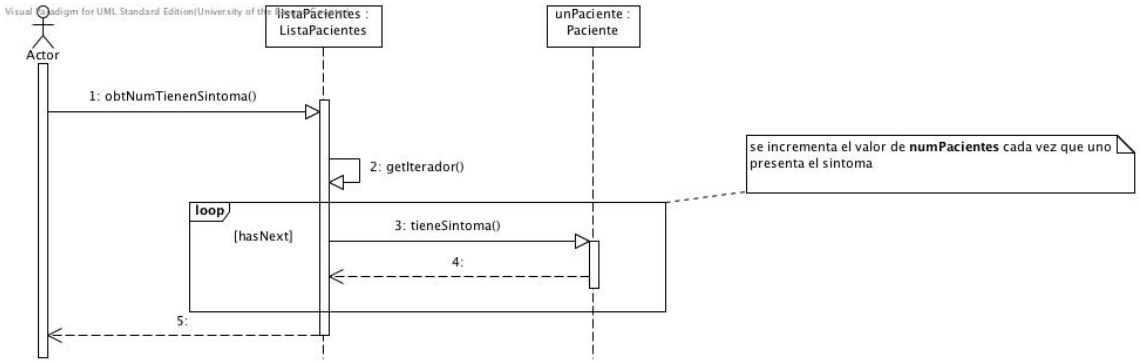
- Finalmente, es necesario entender bien el proceso que se pide implementar, para refinarlo y descomponerlo en acciones más simples que permitan determinar qué partes son responsabilidad de cada clase y, así poder definir las operaciones (métodos) que deben implementarse en cada una.

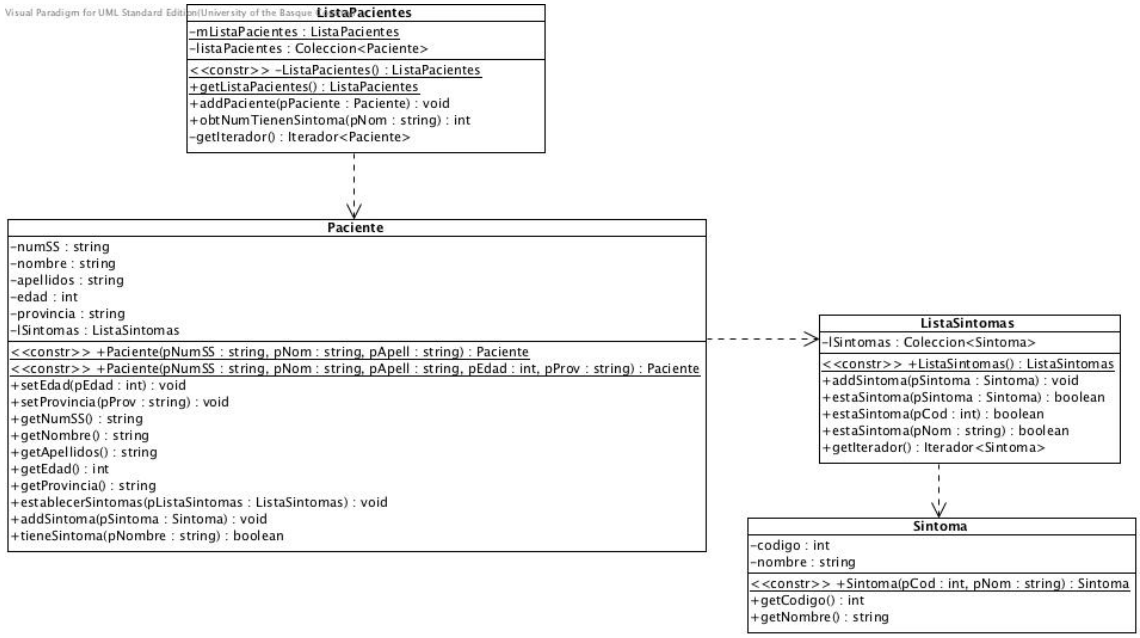
Para determinar cuántos pacientes presentan un síntoma determinado, es necesario saber si un paciente lo tiene o no.



Si predominan las operaciones de consulta sobre los tratamientos de la lista completa y no importa el orden, puede ser más interesante usar colecciones tipo MAP o SET que simplifican el diseño.







Debe haber una relación de dependencia entre dos clases si:

- Una contiene como atributo una o más instancias de la otra
- Una utiliza a la otra en la implementación de alguno de sus métodos.

NOTA DE IMPLEMENTACIÓN: En algunos casos las Listas, tales como ListaSintomas pueden presentar sólo la funcionalidad propia de las colecciones de JAVA. En estos casos, no será necesario implementar una nueva clase, siendo suficiente instanciar la clase genérica correspondiente

AMPLIACIÓN PARA TERMINAR EN CASA:

Además de la información sobre los enfermos, el ambulatorio también dispone de un catálogo con todas las enfermedades o pandemias existentes, almacenando por cada una su nombre y la lista de síntomas que presenta la enfermedad.
Teniendo en cuenta que es posible registrar varias pandemias en el mismo momento, el ambulatorio desea almacenar la información sobre las distintas enfermedades activas.
Se quiere ampliar la aplicación para que, por cada paciente, cruce sus síntomas con las pandemias que activas, obteniendo la lista de enfermedades que podría estar sufriendo.