-

Práctica 4. Estructuras STL

Sesiones de prácticas: 2

***Importante**: esta práctica y las siguientes deben ser realizadas por los estudiantes que convalidaron las tres primeras prácticas.*

Objetivo

Aprender a utilizar estructuras de datos de la Standard Template Library (STL).

Descripción de la EEDD

En esta práctica reemplazamos las EEDD implementadas en sesiones anteriores por contenedores de STL.

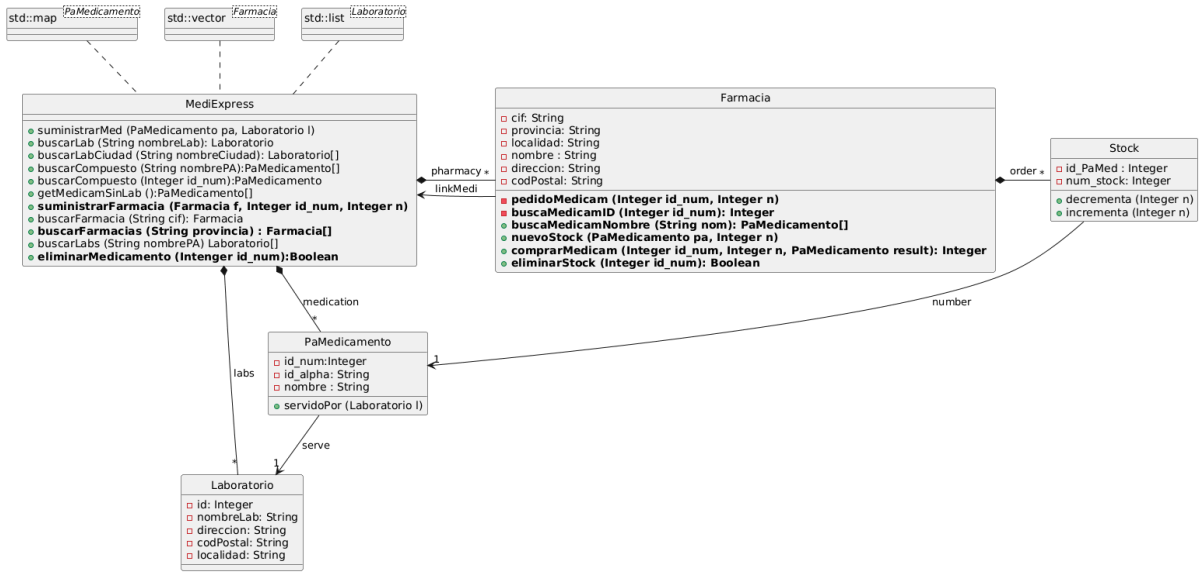
Descripción de la práctica

En esta nueva práctica vamos a sustituir las estructuras de datos iniciales por las de STL. También se va a mejorar el diseño para controlar el stock de medicamentos en las farmacias. La nueva funcionalidad viene de añadir la clase *Stock*, que sirve para conocer en todo momento el stock de medicamentos que tiene la farmacia. El atributo *id_PaMed* debe coincidir con *PaMedicamento::id_num* y el *num_stock* es el número de medicamentos o stock que tiene la farmacia en sus almacenes. Las funciones miembro aumentan o decrementan el número de unidades del medicamento. Cuando el stock queda a cero, no se elimina esta clase.

El resto de la funcionalidad es similar a la práctica anterior, aunque considerando los siguientes cambios. La función *Farmacia::buscarMedicamID()* ahora es privada y devuelve el stock que tiene la farmacia de un medicamento. Si existe un objeto *Stock* se devuelve *Stock::num_stock,* pero si no existe, entonces se devuelve 0**.** Por tanto, devolverá 0 en el caso de que se haya acabado el stock que había o porque la farmacia nunca ha tenido ese medicamento disponible. La función que tiene una funcionalidad clave ahora es *Farmacia::comprarMedicam()*, que se llama cada vez que alguien necesita comprar un medicamento en la farmacia (si quiere comprar diferentes medicamentos debe llamarse varias veces). Para ello, se debe indicar el *id* numérico del medicamento y el número de unidades que necesita. Lo que hace entonces la función es llamar a *Farmacia::buscaMedicamID(),* y si

existe el número suficiente de unidades de este medicamento, se decrementa el *Stock::num_stock* y se termina el proceso. Si el objeto Stock de ese medicamento no existe o es insuficiente, entonces se llama a *Farmacia::pedidoMedicam()* indicando el número de stock que quiere incrementar. La función también devuelve el número de medicamentos que tiene disponibles inicialmente (el devuelto en *Farmacia::buscarMedicam())* y devuelve el medicamento que se compra por parámetro (result). La función *Farmacia::pedidoMedicam()* debe llamar a *MediExpress::suministarFarmacia()* para que le suministre el medicamento. La función *MediExpress::suministarFarmacia()* hace algo similar a la versión anterior localizando el objeto *PaMedicamento* vía *MediExpress::medication*, pero luego llama a *Farmacia::nuevoStock()*. Esta función mira si el objeto stock ya existe, y en ese caso incrementa su stock, pero si nunca antes se vendió el medicamento, entonces debe crear un nuevo objeto Stock y enlazarlo correctamente con *PaMedicamento*. La función *Farmacia::eliminarStock()* elimina definitivamente el stock asociado a un medicamento. También se pueden buscar medicamentos por nombre parcial con la función *Farmacia::buscarMedicamNombre()*.

La nueva función *MediExpress::buscarFarmacias()* devuelve todas las farmacias de una provincia dada. La función *MediExpress::eliminarMedicamento()* elimina dicho medicamento de la relación *MediExpress::medication*, pero también debe eliminar todos los objetos Stock que se relacionan con dicho medicamento a través de *Farmacia::eliminarStock()*. Esta función devuelve V/F según sea exitoso o no el borrado.



Proyecto de prueba: Gestión eficiente de medicamentos Express

A continuación se describen los pasos que debe seguir el **constructor de la clase MediExpress**, que tomará los nombres de los 3 ficheros como parámetros:

1. Leer los ficheros de las prácticas anteriores y cargar la información en los contenedores de STL que indica el nuevo UML. La relación *Farmacia::order* se implementará con un *std::set*, para lo cual la clase *Stock* necesita implementar el *operator<*. Se recuerda que para hacer búsquedas/borrados con set, se debe crear un objeto vacío que contenga la clave a buscar.
2. Al igual que en la práctica anterior, enlazar de forma automatizada los medicamentos (un total de 3310) con los laboratorios (un total de 1579), incluyendo a los 152 ubicados en Madrid. Enlazar también las farmacias con los medicamentos de forma consecutiva cada 100 medicamentos, como se hizo en la práctica anterior, pero esta vez a través de la clase *Stock* como se ha comentado anteriormente. Hacer que haya inicialmente en stock 10 medicamentos de cada tipo.

Una vez que la estructura está inicializada realizar las siguientes acciones:

1. El magnesio ha sido muy recomendado últimamente por especialistas en redes sociales. Hacer que en todas las farmacias de la provincia de Sevilla vayan 12 personas a comprar "ÓXIDO DE MAGNESIO" con ID=3640 y si no hay en stock que compren "CARBONATO DE MAGNESIO" con ID=3632 y, si no, que compren "CLORURO DE MAGNESIO" con ID=3633. Mostrar por pantalla el stock de cada medicamento mencionado en cada farmacia.
2. Localizar todas las farmacias de Madrid que proporcionen algún medicamento con "VIRUS" en su nombre. Contarlas y listarlas en pantalla.
3. La Organización Mundial de la Salud ha prohibido cualquier medicamento con Cianuro, y las farmacias que lo suministran deben deshacerse de él inmediatamente. Eliminar el medicamento con ID=9355 de todo el sistema para que desaparezca también del stock de todas las farmacias. Buscarlo para que se compruebe que no está. Probar a eliminar el medicamento con ID=3244.

Para los que trabajan en parejas:

1. Una alerta de gripe en la Comunidad de Madrid ha determinado que todas las farmacias deben incrementar en 20 unidades su suministro de medicamentos para el virus de la gripe, en concreto el medicamento ID=997. Hacer este incremento a todas ellas y luego listar la/s farmacia/s que tenga 30 unidades de este medicamento (10 + 20). Mostrarlas en pantalla.

IMPORTANTE:

- **Cuidado con realizar copias de objetos**
- **Añadid *getters* y *setters* cuando se necesiten**