-

Universidad de Jaén
Departamento de Informática

Prácticas de Estructuras de Datos
Grado en Ingeniería en Informática
Curso 2021/22

Práctica 3. Árboles AVL

Sesiones de prácticas: 2

Objetivos

Implementar la clase `AVL<T>` utilizando **patrones de clase y excepciones**. Programa de prueba para comprobar su correcto funcionamiento.

Descripción de la EEDD

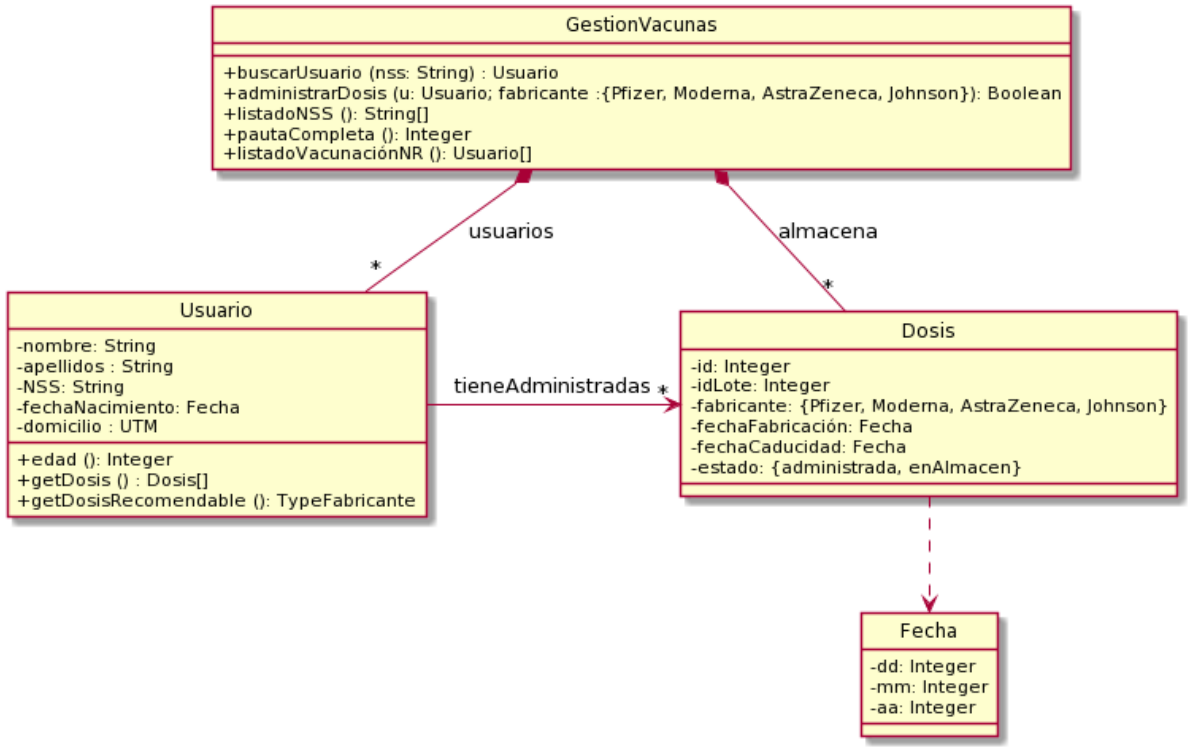
Implementar la clase `AVL<T>` para que tenga toda la funcionalidad de un árbol equilibrado AVL en memoria dinámica, tal y como se describe en la Lección 11, utilizando patrones de clase y excepciones. Los métodos a implementar serán los siguientes:

En concreto se usará:

- Constructor por defecto `AVL<T>`
- Constructor copia `AVL<T>(const AVL<T>& origen)`.
- Operador de asignación (=)
- Rotación a derechas dado un nodo `rotDer(Nodo<T>* &nodo)`
- Rotación a izquierdas dado un nodo `rotIzq(Nodo<T>* &nodo)`
- Operación de inserción `bool inserta(T& dato)`
- Operación de búsqueda recursiva `T* buscaRec(T& dato)`
- Operación de búsqueda iterativa `T* buscaIt(T& dato)`
- Recorrido en inorden[1] `VDinamico<T*> recorreInorden()`
- Número de elementos del AVL `unsigned int numElementos()`
- Altura del AVL, `unsigned int altura()`
- Destructor correspondiente

Tanto el constructor copia como el operador de asignación deben crear copias idénticas. El contador de números de elementos puede realizarse mediante un atributo contador o mediante un recorrido en O(n).

[1] Para poder mostrar por pantalla los datos del arbol es necesario que la clase T haya implementado el operador << http://en.cppreference.com/w/cpp/language/operators



NOTA: TypeFabricante: {Pfizer, Moderna, AstraZeneca, Johnson}

Programa de prueba: gestionar vacunación de usuarios

Siguiendo con la gestión del proceso de vacunación, en esta práctica añadimos la clase *GestionVacunas*, encargada de la gestión tanto de los objetos de tipo *Usuario* como a las *Dosis* que están siendo administradas.

GestiónVacunas tiene la siguiente funcionalidad: es capaz de localizar a un *Usuario* por su número de la seguridad social *(buscarUsuario)*. Es importante que esta función no devuelva una copia del objeto, mejor devolver un *Usuario**. De igual modo gestiona la administración de una dosis indicando únicamente el fabricante *(administrarDosis)*. Con esta función se indica que a un usuario se le decide administrar, por ejemplo, una dosis de AstraZeneca. En dicho caso se debe buscar alguna de las dosis disponibles en el almacén (*Dosis::enAlmacen*). Además se deben tener en cuenta las restricciones de edad según la Tabla 1. Si no hubieran dosis compatibles, se puede poner cualquier vacuna, pero la función debe dar un valor falso. Finalmente se puede conocer en todo momento el porcentaje de personas que tienen la pauta completa con *pautaCompleta()*. También podemos conocer quienes se han puesto alguna de las vacunas no recomendadas, sacando una lista de usuarios que no han recibido la pauta recomendada por las autoridades (*ListadoVacunaciónNR()*). La función *listadoCompletoNSS()* devuelve un vector de tipo string (*Vdinamico<String>*) con todos los números de seguridad social ordenados.

La clase Usuario tiene ahora dos funciones más, la primera getDosis() devuelve un vector de las dosis ya suministradas al usuario y getDosisRecomendable que devuelve el tipo de dosis recomendada según la Tabla 1.

Las estructura de datos en *GestiónVacunas* serán un AVL instanciado a Usuario, *(AVL<Usuario>)* usuarios; y el vector de Dosis (*VDinamico<Dosis>*). Para la relación de Usuario con Dosis se usará una lista simplemente enlazada *ListaEnlazada<Dosis*>*, considerando que se trata de una asociación. La función *GestionVacunas::ListadoVacunacionNR()* y *Usuario::getDosis()* devolverá en ambos casos un vector de punteros.

Rango de edad (años)	**Vacuna recomendada**
[0,12]	-
]12, 30]	Johnson
]30, 50]	Moderna
]50,65]	AstraZeneca
]65,-]	Pfizer

Tabla 1: Rango de edades y vacuna recomendada

Programa de prueba: Proceso de vacunación

Crear un programa de prueba con las siguientes indicaciones:

- Crear un árbol AVL instanciado a 1000 enteros aleatorios y comprobar que todas las funciones trabajan correctamente.
- Indicar la altura del árbol AVL.
- Implementar las clases del diagrama UML.
- Instanciar el árbol con objetos de tipo *Usuario* con el fichero adjunto. La clase *GestionVacunas* toma el nombre del fichero en el constructor.
- Instanciar el vector de dosis al fichero de dosis adjunto con 30.000 dosis. La clase *GestionVacunas* toma el nombre del fichero en el constructor.
- Obtener los NSS de los usuarios en un vector dinámico *Vdinamico<String> vecAuxilia* obtenido con la función GesionVacunas::*listadoCompletoNSS()*. Este vector se utilizará para las funciones que aparecen a continuación.
- Vacunar de la primera dosis de forma automática a todos los usuarios menos a los que su NSS acabe en 0. Para ello se recorre *vecAuxiliar*, que tiene los NSS de todos los usuarios y se buscan dichas claves en el *AVL<Usuario>*. Diseñar un algoritmo para realizar esta asignación. Tened especial cuidado de no suministrar más de una vez una dosis (cuando se administra una dosis el estado cambia a *administrada*). Imprimir el número de dosis administradas.
- Ahora, usando el mismo proceso anterior, vacunar de la segunda dosis a todos los usuarios menos a aquellos cuyo NSS acabe en 3 (y tampoco obviamente a los que acaben en 0). Imprimir el número de dosis administradas.

- De igual forma, vacunar a todos los usuarios mayores de 75 años con la tercera dosis. Para saber la edad, buscar al usuario (*GestionVacunas::buscarUsuario()*). Si no estaban antes vacunados, no están vacunados de las dos anteriores no se vacunan de la tercera. Imprimir el número de dosis administradas.
- Mientras haya dosis de las recomendadas se ponen éstas, sino se ponen de cualquier otro tipo pero se devuelve un falso en la función: *GestionVacunas::administrarDosis(),* a juicio del facultativo de turno (vuestro propio algoritmo).
- Indicad el porcentaje de personas con pauta completa, es decir, todos los menores de 75 con 2 dosis y los de 75 con 3 dosis con la función *GestionVacunas::pautacompleta()*. Para ello es conveniente hacer uso de la función AVL::recorreInorden()
- Listar quienes no han sido vacunados con la pauta recomendada *GestionVacunas::listadoVacunacionNR();* Para ello es conveniente hacer uso de la función AVL::recorreInorden()
- Forzar la vacunación de la primera dosis de estos NSS que no han sido aún vacunados con una dosis de Moderna: 1622650940; 1941046560; 1756824615; 1625692780; 1855345010.

Nota: Para los que trabajan en parejas:

- Medir los tiempos de búsqueda total para localizar los 5 NSS anteriores (conjuntamente, no por separado) con respecto a tenerlos en un vector metidos con el orden del propio fichero.
- Añadir la función: *GestionVacunas::vacunasAlmacen(): Integer* que devuelva el número de vacunas sin utilizar tras realizar todo el proceso de vacunación anterior.

Estilo y requerimientos del código:

1. El código debe ser claro, tener un estilo definido y estar perfectamente indentado, para ello se pueden seguir algunos de los estilos preestablecidos para el lenguaje C++ (http://geosoft.no/development/cppstyle.html).frdd

2. Deben comprobarse todas los posibles errores y situaciones de riesgo que puedan ocurrir (desbordamientos de memoria, parámetros con valores no válidos, etc.) y lanzar las excepciones correspondientes, siempre que tenga sentido. Leer el tutorial de excepciones disponible en el repositorio de la asignatura en docencia virtual.

3. Se valorará positivamente la calidad general del código: claridad, estilo, ausencia de redundancias, etc.