PARTE 3: Ejercicio de programación JAVA

Este ejercicio comprende **3** partes:

1. Desarrollo de la funcionalidad especificada más abajo
2. Desarrollo del programa ejecutable con datos de ejemplo provistos y entrega de todo el proyecto Netbeans en la tarea correspondiente de la webasignatura.
3. Desarrollo de los casos de test para las funcionalidades requeridas

ESCENARIO

Una institución de asistencia médica en la que trabajamos, registra a sus cientos de miles de afiliados en una lista, insertándolos en el orden en el que se afilian. El registro y la identificación del afiliado se hace a partir de la cédula de identidad (8 cifras numéricas, sin puntos ni guiones).

Cada afiliado, además de los datos personales básicos (cédula de identidad y nombre y apellido), tiene una lista con el histórico de consultas médicas realizadas y otra lista de consultas médicas a las que está anotado.

Los datos de una consulta son: fecha (de tipo fecha), especialidad (string), médico (número de cédula) y resultado. Resultado es un atributo de tipo entero que puede tener tres valores: 0 si todavía no se ha producido, 1 si el afiliado concurrió y 2 si el afiliado no concurrió. Al realizarse la anotación, el campo se inicializa en 0, y a la hora de la consulta, el personal de la policlínica registra en 1 si el afiliado concurre, o en 2 si cerrada la hora de la consulta el afiliado no concurrió.

Dado que el volumen de consultas diarias ha crecido y también la cantidad de pacientes, las recorridas sobre listas comienzan a ser lentas para la búsqueda de cierta información. Es por ello que el gerente de aplicaciones nos ha encargado el desarrollo de nuevas funcionalidades que permita optimizar el sistema actual.

Se desea contar con funcionalidades que permitan:

1. Migrar el padrón de afiliados a un Árbol binario de búsqueda que permita optimizar las consultas de afiliados por Cédula.
2. Recorrer todas las noches los afiliados y en particular las listas de consultas en las que se encuentra anotado y cargarlas en un árbol de consultas, que permitan buscar las consultas que se tienen en el día de una forma mejor que la que se tiene actualmente. En dicho árbol han de cargarse las consultas en estado 0, para la fecha del día. Podemos considerar que la clave es la concatenación de los campos ci_afiliado, fecha, especialidad (para lo cual podemos asumir que un afiliado no puede tener dos consultas a una misma especialidad un mismo día).

PARTE 1: Funcionalidad a desarrollar (vale 35%):

Dado un TArbolBBPadronAfiliados (que desciende de TArbolBB) implementar los siguientes métodos:

- TArbolBB<Afiliado> cargarDesdeLista(TLista listaAfiliados) // realiza la carga de los afiliados de la lista pasada por parámetro en un TArbolBB (con CI como clave).
- TArbolBB<Afiliado> obtenerConsultasDelDia(String fecha) // realiza la recorrida de los afiliados, las listas de consultas anotados, filtra por fecha (parámetro) y los carga en un TArbolBB para retornar.
- Integer cantidadConsultasdelDia() // obteniendo la cantidad de consultas del día de hoy.

PARTE 2: PROGRAMA (vale 35%)

La clase principal se denomina "**Parcial2**", y tiene su correspondiente método "**main**". En éste, implementa lo necesario para aplicar los TDA y métodos desarrollados.

1. Se provee una clase TArbolBB para representar el árbol de afiliados.
2. Se provee interfaz IArbolPadronAfiliados y clase TArbolPadronAfiliados para incluir los métodos solicitados.
3. Instanciar una TLista y cargar los datos del archivo "padron.txt", en donde la clave primaria será la cédula del afiliado.
4. Instanciar y cargar las consultas de los afiliados contenidas en los archivos "históricas.txt" y "agendadas.txt".
5. Instanciar un TArbolPadronAfiliados y ejecutar el método cargaDesdeLista(Lista<Afiliados> listaAfiliados) para cargar los afiliados que tenemos actualmente en una lista, al árbolBB. Imprimir por pantalla la recorrida in orden del árbol resultante.
6. Aplicar el método obtenerConsultasDelDía() para obtener las consultas del día de hoy. Descargar a un archivo llamado "consultasDelDia.txt" la recorrida del árbol resultante in orden.
7. Obtener la cantidad de consultas del día que se tienen, e imprimir esa cantidad por pantalla.

PARTE 3: TEST CASES (vale 30%).

Implementa el o los **Casos de Prueba** necesarios para verificar el correcto funcionamiento de los métodos desarrollados.

NOTA IMPORTANTE:

Se proveen las interfases y clases necesarias. Deben implementarse los métodos de TElementoAB<T> necesarios (insertar, buscar, inorden, etc.). NO SE DEBEN ALTERAR LAS INTERFASES DE TArbolBB ni TElementoAB provistas, ni agregar otros métodos que los requeridos en las interfases.

NO DE DEBEN CREAR NUEVAS CLASES Y NO SE DEBE INCLUIR NINGUN METODO NO SOLICITADO O INNECESARIO.

ENTREGA: Debes entregar TODO el proyecto Netbeans y los archivos de salida solicitados, en un archivo comprimido "**Parcial2.zip**" en la tarea "**PARCIAL2-PARTE3**" publicada en la webasignatura, hasta la hora indicada.