PARTE 3: Ejercicio de programación Java
Duración: 75 minutos

CONTEXTO

Contexto

Un sitio de películas almacena la información de las películas que comercializa en un árbol binario de búsqueda de tipo FILM. Entre otros muchos, la información de un FILM tiene los siguientes atributos:

id: de tipo string; es el identificador único por el cual se almacenan las películas en el árbol.

Título: de tipo string, es el nombre de la película.

Actores: lista de tipo ACTOR; es una lista que contiene a todos los actores que actúan en la película.

A su vez, mantiene un árbol binario de búsqueda de todos los actores de las películas que comercializa. Los atributos de un ACTOR, entre otros muchos, son de interés para este problema:

Nombre: de tipo string; es el identificador único del actor, por el cual se almacenan los actores en el árbol de actores.

Films: lista de tipo FILM; es una lista de películas en las que actúa.

Una consulta frecuente que debe ser implementada es que, dado un actor ("elActor"), se obtenga una lista de todos los otros co-actores con los actuó en cualquier película; cada uno de esos co-actores tiene a su vez una lista de las películas en las que participó con el actor ("elActor").

De esta forma, el encargado de desarrollo le solicita implementar el método que permita evacuar esa consulta.

NOTA IMPORTANTE: Las instancias de tipo ACTOR de los nodos de esta lista devuelta por el método, deben ser diferentes a la instancia de ACTOR del árbol binario en donde se guardan los actores, ya que la lista de películas en este caso está compuesta solamente por las que comparte con "elActor".

Ejemplo:

Supongamos que el tenemos solamente dos actores en nuestros archivos de datos: Gerard Depardieu y Jeremy Irons.

A su vez, el actor Gérard Depardieu tiene una lista de películas: "Cyrano de Bergerac", "Asterix y Obelix" y "El hombre de la máscara de Hierro".

Por su parte Jeremy Irons, tiene "El mercader de Venecia" y "El hombre de la máscara de Hierro".

La instancia de Actor en el árbol binario debería contener en Gerard Departieu sus 3 películas y en Jeremy Irons, las dos en las que actúa.

Por su parte, luego de ejecutar el método de co-actores con el parámetro Gerard Depardieu, la lista resultante tiene solamente un nodo Actor, el correspondiente a Jeremy Irons. Por su parte, la lista contenida en Jeremy Irons, también tiene solamente un nodo: "El hombre de la máscara de Hierro"; única película en la cual participan ambos actores.

Ejemplo de listas de películas de actores en árbol binario.

Gerard Depardieu
Lista de películas El hombre de la máscara de Hierro Ásterix y Óbelix Cyrano de Bergerac

Jeremy Irons
Lista de películas El hombre de la máscara de Hierro El mercader de Venecia

Ejemplo de listas de películas de actores en lista resultante del método coactores con parámetro: Gerard Depardieu

Jeremy Irons
Lista de películas El hombre de la máscara de Hierro

Además, como la búsqueda por actor es muy frecuente, se ha decidido que para optimizarla se construya un árbol binario de búsqueda óptimo a partir de un archivo de actores buscados que ha conseguido el sitio de películas. Cada línea de ese archivo contiene un nombre actor; algunos corresponden a actores que se encuentran en el árbol, pero otros no. Cada actor aparecerá en más de una línea.

PASO 1

Descarga los archivos de la webasignatura y

1. crea el proyecto de nombre PARCIAL3
2. estudia detenidamente las interfaces y clases provistas, las operaciones que deben ser desarrolladas y los archivos de datos provistos
3. completa las clases de árbol necesarias para implementar las interfaces provistas

PASO 2

Desarrolla

1. En la clase TArbolActores, el método Lista<Actor> coactores(String elActor)
2. Desarrolla los test cases necesarios necesarios para probar el funcionamiento del método anterior.
3. En la clase Main, los métodos estáticos indicados en el orden dado:
    a. public static void cargarDatos(TArbolBB<Film> pelis, TArbolBB<Actor> actores), que carga en los árboles las películas y los actores, a partir de los archivos de películas y actores.
    b. public static void cargarFrecuencias(TArbolBB<Actor> actoresBase), que cuenta las frecuencias de búsquedas exitosas e infructuosas sobre el árbol de actores, a partir del archivo de búsquedas.
    c. public static void armarABO(TArbolBB actoresBase, TArbolActores losActores), que arma el árbol óptimo de actores.
    d. public static void salidaCoActores(Lista<Actor> losActores, String actor), que genera el archivo de salida con el formato indicado en el archivo de muestra.

PASO 3

Ejecuta el programa, que debe:

1. Cargar en los árboles las películas y los actores, a partir de los archivos correspondientes de películas y actores provistos.
2. Contar las frecuencias de búsquedas exitosas e infructuosas sobre el árbol de actores, a partir del archivo de búsquedas provisto.
3. Armar un árbol óptimo de actores a partir de los vectores de frecuencias que se extraigan del árbol de actores usado como base para contar las frecuencias.
4. Ejecutar el método “coactores” del árbol óptimo de actores hallado, con el actor que se indique.
5. Generar el archivo de salida correspondiente con el formato dispuesto en el archivo de salida ejemplo provisto.

PASO 4

Entrega todo el proyecto y el archivo de salida solicitado, en un archivo comprimido “Parcial3.zip” en la tarea “PARCIAL3-PARTE3” publicada en la webasignatura, hasta la hora indicada.