OBSERVACIONES DEL LA PRACTICA

Estudiante 1 Cod XXXX

Estudiante 2 Cod XXXX

1. ¿Cuáles son los mecanismos de interacción (I/O: Input/Output) que tiene el **view.py** con el usuario?

El primer mecanismo de interacción que encontramos en el view.py se trata del menú desplegable de la aplicación, donde a el usuario se le muestran las diferentes opciones que tiene para ejecutar, ligadas a un numero del 0 al 4 para que el usuario ingrese dicho número y la aplicación ejecute esa opción.

Cada una de estas opciones botara un output al usuario, donde le indique que la acción se ha ejecutado junto a detalles específicos de la acción (ej;cantidad de libros cargados) en el caso de la opción 1 (Cargar información en el catálogo), en las opciones del 2 al 4 (Consultar los top x libros por promedio, consultar los libros de un autor, libros por genero) la aplicación le pedirá al usuario más información para completar la acción y posteriormente hará algo similar a lo sucedido en la opción 1. La opción 0 servirá para salir de la aplicación.

1. ¿Cómo se almacenan los datos de **GoodReads** en el **model.py**?

Los datos del archivo GoodReads se almacenan en el modelo partir del siguiente proceso:

Para empezar, es necesario que el usuario ejecute la opción 1 que se encuentra en el menú que despliega la aplicación en el view.py, esto hará que se ejecute la función initCatalog en el View que a su vez la hará ejecutar en el Controller, que llamara a la función newCatalog en el Model. Esta última función crea un diccionario con las llaves “books”, “authors”, “tags” y “book tags”, donde cada una de estas tendrá como valor una lista vacía donde posteriormente se agregarán los datos de los archivos de GoodReads. La función devuelve este diccionario a initCatalog en el Controller y este se lo pasa a la función de mismo nombre en el View, posteriormente esta va a ser llamada como parámetro en la función loadData, que tomara el diccionario y lo pasara a la misma función en el Controller que ejecutara las funciones loadBooks, loadTags y loadBooksTags pasándoles el diccionario como parámetro a cada una. Cada una de estas 3 funciones abrirá el correspondiente archivo csv que se encuentra en GoodReads (books, tags y book\_tags, respectivamente) y se encargará de leer línea por línea para ir enviando cada una a las funciones add(Book, Tag, BookTag) en el Model. Estas funciones “add” del Model reciben también el diccionario del catálogo como parámetro. Las funciones van a ir agregando línea por línea la información a la lista vacía correspondiente a la categoría del diccionario y al archivo especifico que estén leyendo. De esta forma cada línea de información que se encuentra en cada uno de los archivos csv de GoodReads es agregado a el catálogo (estructura de datos) del Model, de manera categorizada.

1. ¿Cuáles son las funciones que comunican el el **view.py** y el **model.py**?

Las funciones que comunican la vista con el modelo se encuentran en el controlador(controller.py). Estas son:

initCatalog, loadData, loadBooks, loadTags, loadBooksTags, getBooksByAuthor, getBestBooks y countBooksByTag

Estas funciones se invocan desde la vista por funciones que también pasan por parámetro el input del usuario para que las funciones del controlador pueden pasarlo al modelo, cuyas funciones realizaran los algoritmos de funcionamiento de la aplicación.

1. ¿Cómo se crea una lista?

Con la función newList de list.py, para crearla se necesita importar el archivo list.py a el archivo en el que se este trabajando. A la hora de crear una lista esta recibe por parámetro los siguientes argumentos: datastructure (si no se especifica será single linked por defecto) que es la estructura de datos con la que se puede crear la lista, cmpfunction (si no se especifica será None por defecto) que es la función de comparación que se puede utilizar para comparar los elementos de la lista, key (si no se especifica será None por defecto) que es el identificador utilizado para comparar dos elementos de la lista cuando la función de comparación esta por defecto, filename (si no se especifica será None por defecto) que se utiliza para colocar el nombre de un archivo en caso de que se quiera crear una lista a partir de este, y delimiter (si no se especifica se utilizara una coma por defecto) que se utilizara en caso de que se haya dado un filename para separar los campos que se encuentran en el archivo a partir de un valor.

1. ¿Qué hace el parámetro **cmpfunction=None** en la función **newList()**?

Utiliza una función para comparar los elementos de la lista creada, por defecto ya trae una función, sin embargo el usuario puede utilizar una función específica que desee si la ingresa como parámetro.

1. ¿Qué hace la función **addLast()**?

La función agrega un elemento que se coloque como parámetro a una lista que también se ingrese, en la última posición de esta, también actualiza el apuntador a la nueva última posición e incrementa el tamaño de la lista en 1.

1. ¿Qué hace la función **getElement()**?

La función retorna el valor del elemento que se encuentra dentro de la posición que debe ser dada por parámetro dentro de una lista también especificada por parámetro.

1. ¿Qué hace la función **subList()**?

La función crea una sublista que se crea a partir de una lista dada por parámetro. Se deberá ingresar la posición del elemento de la lista original donde empezara la sub lista y también la cantidad de elementos siguientes que se tomaran (estos dos valores por parámetro).

1. ¿Observó algún cambio en el comportamiento del programa al cambiar la implementación del parámetro **“ARRAY\_LIST”** a **“SINGLE\_LINKED”**?

Si, al cambiar el código para que las listas de tags y book tags fueran single linked en lugar de arrays, el tiempo de carga para consultar el top de libros por promedio y para consultar los libros de un autor aumento exponencialmente.