OBSERVACIONES DEL LA PRACTICA

OBSERVACIONES DEL LA PRACTICA

Juan Camilo Falla – C.E. 201922219

Nicolás Klopstock – C.E. 202021352

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

El primero es la opción de que se le dé la opción al usuario de ingresar la opción que desee. Además de esto, hay otro tipo de Input cuando se le pide al usuario ingresar el autor del que quiere saber la información, el top X deseado de libros por promedio, o alguna etiqueta que desee buscar el usuario.

Sobre le Output, el programa le devuelve al usuario la información que desea, dependiendo de la opción que ingresó (el Input).

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

Se almacenan en varios pasos. El primero es crear un catálogo, es decir, un diccionario en el que las llaves son (en este caso) books, authors, tags y book\_tags, y los valores son arreglos donde se van a guardar los nombres de los libros (en la llave books), los nombres de los autores (en la llave authors), los tags (en la llave tags), y los tags de los libros (en la llave book\_tags). Los datos se encuentran relacionados entre sí, ya que comparten la misma posición en los distintos arreglos creados para cada llave del catálogo.

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

Las funciones que comunican el view.py con el model.py son:

En view.py está la función **initCatalog()**, la cual llama a una función de controller.py con el mismo nombre. Esta esta última llama a la función **newCatalog()** de model.py.

1. ¿Cómo se crea una lista?

Una lista se crea, primero definiendo la función. Los parámetros de esta función son muy importantes, ya que son los que dan los detalles de la creación de la lista. Entre estos parámetros está el llamado “datastructure”, el cual es el que define el tipo de estructura de datos que va a ser esta lista. Hay dos tipos, el primero es un array\_list el segundo un single\_linked. Con esto ya definido, se hace un **try** para crear la lista. Esto se hace para que la función tenga un “plan B” y no muera por si sucede algún tipo de error durante la creación de la lista.

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

El parámetro **cmpfunction=None** es una función que compara elementos de la lista. En este caso, el parámetro es **None**, esto hace que se use la función de comparación por defecto. Para eso existe un siguiente parámetro llamado **Key**, el cual es un identificador que compara dos elementos de una lista con la función de comparación.

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

La función **addLast()** agrega un elemento al final de la lista que entra como parámetro de la función. Al agregar un elemento a la lista, se debe, además, aumentar el tamaño de la lista por uno. Sucede lo mismo que con la función **newList()**, es decir, se usa un **try** para que la función tenga un colchón donde caer si hay algún error en el proceso de agregar un elemento al final.

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

La función **getElement()** recorre la lista hasta la posición indicada como parámetro, recoge ese elemento y lo retorna. Claramente, la lista no debe estar vacía y la posición debe ser un número entero entre 0 y el tamaño de la lista. En esta función también se usa el **try** para que la función no colapse en caso de que suceda algún error en el proceso de encontrar y retornar el elemento.

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

La función **subList()** crea una nueva lista a partir de una ya existente a partir de la posición que entra como parámetro hasta el número de elementos ingresado. En esta función, como en muchas otras ya mencionadas, usa el **try** para que el programa pueda hacer algo en caso de que haya un error en el proceso de crear una sub-lista a partir de una lista ya creada y de este modo, evitar el colapso del código.

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

Al realizar el cambio de los parámetros se observó un aumento en el tiempo transcurrido para cargar los datos de los archivos, el cual era relativamente bajo para los archivos de datos pequeños, en los cuales se presentó una diferencia de 0.1s entre los dos. La observación realizada se encuentra de acuerdo a lo aprendido ya que las listas encadenadas requieren de la creación de nodos y sus apuntadores.