OBSERVACIONES DEL LA PRÁCTICA

Iván Camilo Ballén Méndez Cód. 202011440

María José Sáenz Rodríguez Cód. 202013542

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

El view.py permite la interacción del usuario con el menú de la aplicación. Este tiene funciones las cuales cumplen 3 objetivos en general:

1. Imprimir el menú de la aplicación en consola, así como recibir e interpretar las opciones que digita el usuario de acuerdo a la funcionalidad que quiere ejecutar.
2. Comunicar al controller qué función desea ejecutar el usuario. Para esto, se crean funciones que retornan el valor de funciones en el controller el cual va a interaccionar con el modelo dependiendo de la funcionalidad requerida.
3. Editar la información de output de una funcionalidad requerida. Se puede ver en el view.py que antes de comenzar el ciclo que mantiene corriendo el menú se definen funciones que reciben datos del controller y editan la forma en la que se van a exponer estos datos en la consola.
4. ¿Cómo se almacenan los datos de **GoodReads** en el **model.py**?

Los datos se almacenan en un diccionario nombrado ‘catalog’ el cual le asigna a la llave ‘books’ un TAD de lista encadenada mientras que a ‘authors’, ‘tags’ y ‘books-tags’ les asigna un TAD arreglo.

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

Las funciones que comunican el view.py y el model.py son por definición las funciones del controller. En las funciones del view.py siempre se hacen llamados a las funciones que están en el controller las cuales a su vez hacen llamados a funciones en el model dependiendo de la funcionalidad que se requiera, aunque las funciones son bastantes, se pueden clasificar en 4 tipos:

1. Funciones de inicialización de almacenamiento de datos
2. Funciones para cargar datos
3. Funciones de ordenamiento
4. Funciones de consulta
5. ¿Cómo se crea una lista?

Una lista se crea haciendo un llamado a la biblioteca DISClib, específicamente al TAD list.py. Se abrevia este llamado con ‘lt’ de tal forma que se puedan llamar los métodos de list.py de la forma lt.\*metodo\*(). Se invoca lt.newlist() y recibe como términos el tipo de estructura y la función de comparación. En caso de no recibir ninguno de los dos, la estructura se inicializa en lista encadenada y la función con un None, pero se debe introducir un parámetro para ‘key’.

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

Este parámetro es el encargado de definir la función de comparación dentro del TAD, en caso de no definirse, por defecto es None pero se define un parámetro para ‘key’; de lo contrario ‘key’ se le asigna el valor None.

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

La función addLast() se encarga de agregar un nuevo elemento a la lista, específicamente en la última posición, y además realiza un cambio en los apuntadores para el caso de las listas encadenadas. Por último, aumenta el tamaño de la lista en una unidad.

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

La función getElement() se encarga de retornar el elemento establecido por el parámetro ‘pos’, el cual se refiere a la posición en la que se encuentra dicho elemento dentro de la lista. Además, se tiene en cuenta que al retornarlo no se elimina, y que una de las condiciones es que el parámetro ‘pos’ debe estar entre los limites de la lista, es decir, mayor a cero, y menor o igual al tamaño de la lista.

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

Esta función retorna una lista nueva considerada una sub-lista, donde sus elementos comienzan o parten del parámetro ‘pos’ y la longitud de esta se basa en el parámetro ‘numelem’. Es decir, básicamente se crea una copia con ciertos elementos según los parámetros de inicio y longitud dados.

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

Al implementar el cambio en el uso de TAD lista y ejecutar el programa, no se observaron modificaciones en su comportamiento.