OBSERVACIONES DEL LA PRÁCTICA

Santiago Sinisterra Arias Cod 202022177

Juan Felipe Serrano Cod 201921654

1. ¿Cuáles son los mecanismos de interacción (I/O: Input/Output) que tiene el **view.py** con el usuario?
2. ¿Cómo se almacenan los datos de **GoodReads** en el **model.py**?
3. ¿Cuáles son las funciones que comunican el el **view.py** y el **model.py**?
4. ¿Cómo se crea una lista?
5. ¿Qué hace el parámetro **cmpfunction=None** en la función **newList()**?
6. ¿Qué hace la funció **addLast()**?
7. ¿Qué hace la función **getElement()**?
8. ¿Qué hace la función **subList()**?
9. ¿Observó algún cambio en el comportamiento del programa al cambiar la implementación del parámetro **“ARRAY\_LIST”** a **“SINGLE\_LINKED”**?
10. El input que tiene view depende de la opción escogida principalmente pero en todas se comunica por medio de la consola. En el menú principal el usuario escoge un número del 0 al 4 para utilizar alguna de las funciones.
    1. Opción 0: Esta opción permite al usuario salir de forma segura del menu y parar de ejecutar el código.
    2. Opción 1: Al escoger la opción 1 el programa tiene un output de mencionar el número de libros cargados, autores cargados y asociación de géneros a libros.
    3. Opción 2: En esta opción el usuario debe ingresar un nuevo número mediante la consola cuando el programa le pregunta cuántos libros quiere ver del top de los más vendidos. El output del programa es una lista con el número de valores ingresados por el usuario donde se muestra el título de los mejores libros junto a su ISBN y su rating
    4. Opción 3: En la opción 3 el usuario puede ingresar un string del nombre del autor que está interesado en buscar en la base de datos. El programa luego devuelve el promedio, el total de libros y el título de ISBN de los libros relacionados al autor buscado. Nota: la búsqueda es flexible.
    5. Opción 4: Similar a la opción 3 el usuario nuevamente puede ingresar un string con la etiqueta que quiere buscar en la base de datos. Se le retorna el número de libros que cumplen las condiciones y una lista con toda la información de los libros marcados con la etiqueta buscada.
11. Como se almacena la info de goodreads?

Respecto a la forma en la que se almacena la información sin procesar, esto se encuentra evidentemente en una subcarpeta llamada Data enm la que se encuentran todos los archivos csv de Goodreads.

Revisando el archivo Controller, la función initCatalog llama al modelo a que cree un nuevo diccionario con las llaves Books, authors, tags y book\_tags, estas llaves luego se les atribuyen una lista y en el caso de los autores y tags se les especifica su función de comparación donde *compare authors* permite una búsqueda de si está presente el carácter sin importar capitalización y *compare tag names* retorna un booleano en vez del habitual 0 y 1. Luego se llama a la función LoadData la cual tiene como parámetro el catálogo que se acaba de crear y llama a otras 4 funciones, la primera, segunda y tercera función se encargan de ir a la dirección del archivo establecida en los datos del proyecto y cargar al catálogo la lista de libros, la lista de tags y la lista de BookTags. La lista de libros luego se organiza basado en sus ratings.

1. Funciones de comunicación de view.py con model.py?

Las funciones de comunicación entre el view y el model se podrían ver en el model y serial init catalog, load data, printAuthorData y printBestBooks.

1. ¿Cómo se crea una lista?

Como se hace en el laboratorio para crear una lista se llama al model y usando la librería lt se usa el método newlist(), este método se puede usar sin parámetros para crear una lista predeterminado pero si se quiere algo en específico de esta también se puede especificar aspectos como el orden que se utiliza con el ‘ARRAY\_LIST’ type, y la función de comparación.

1. ¿Qué hace el value cmpfunction=none en list?

Este valor es en este momento sosteniendo el lugar a la función de comparación que se quiere utilizar en la lista, si no se le es dado ninguna función especial de comparación, este asumirá la función como la predeterminada la cual compara los elementos con mayores y menores(si el primer valor es mayor devuelve 1 y si es menor devuelve -1) si los valores no son ni mas grandes ni más pequeños retorna 0(este sería considerado que encontró que los dos eran iguales).

1. ¿Qué hace AddLast()?

Add last es una función en el adt lista que intenta agregar un elemento a la lista suministrada, internamente la función determina el tipo de la lista(ArrayList o SingleLinkedList) y usa el método adecuado para agregar el elemento en la última posición de la lista. No tiene return.

1. ¿Qué hace la función getElement()?

Similarmente a la función add last, ésta requiere la lista y el número de la posición del elemento que se busca consultar de la lista, después internamente reconoce si es un ArrayList o una SingleLinkedList y usa los métodos adecuados para cada estructura. Retorna el elemento presente en la posición suministrada

1. ¿Qué hace la función Sublist()?

La función sublist tiene como parámetros la lista, la posición desde donde se quiere la sublista y el tamaño de la sublista. Con estos parámetros la función identifica si la lista es ArrayList o Single linked y usa sus métodos adecuados. La función idealmente busca el elemento en la posición dada en la lista y crea una nueva lista con los siguientes elementos de la lista original en orden hasta que la sublista tenga el mismo tamaño que el parámetro especificado. Retorna la nueva sublista está siendo el mismo tipo que su original.

1. Observó algún cambio en el comportamiento del programa con la implementación de SingleLinkedList en vez de ArrayList?

Utilizando la función process\_time() antes y después de las funciones y luego encontrando la diferencia de tiempos en InitCatalog(),LoadData(), printAuthorData(no), printBestBooks(catalog,5) se obtuvo los siguientes datos:

| Funcion | Diferencia de Tiempo | Tipo de Lista |
| --- | --- | --- |
| InitCatalog() | 0.0 | ARRAY\_LIST |
| InitCatalog() | 0.0 | SINGLE\_LINKED |
| LoadData() | 0.171875 | ARRAY\_LIST |
| LoadData() | 0.25 | SINGLE\_LINKED |
| PrintAuthorData(no) | 0.0 | ARRAY\_LIST |
| PrintAuthorData(no) | 0.0 | SINGLE\_LINKED |
| PrintBestBooks(catalog,5) | 0.0 | ARRAY\_LIST |
| PrintBestBooks(catalog,5) | 0.0 | SINGLE\_LINKED |

Como podemos ver en la tabla la única función que presentó un cambio en el tiempo de ejecución fue loadData() y demostró que utilizar Array\_list es más rápido que usar Single\_linked.

| Funcion | Tiempo ejecuión | Tipo de lista |
| --- | --- | --- |
| InitCatalog() | 0 | ARRAY\_LIST |
| InitCatalog() | 0 | SINGLE\_LINKED |
| LoadData() | 577.21875 | ARRAY\_LIST |
| LoadData() | 584.4375 | SINGLE\_LINKED |
| PrintAuthorData(no) | 0 | ARRAY\_LIST |
| PrintAuthorData(no) | 0 | SINGLE\_LINKED |
| PrintBestBooks(catalog,5) | 0 | ARRAY\_LIST |
| PrintBestBooks(catalog,5) | 0 | SINGLE\_LINKED |