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?
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”**?

Respuestas:

1. Al iniciar la aplicación se despliega el menú de opciones, que le da al usuario la opción de ingresar el número de la opción que desea ejecutar. Dependiendo de la opción escogida por el usuario (un número del 1 al 4 o 0 para salir) se ejecuta la función correspondiente y dependiendo de la opción la aplicación le pedirá algunos más datos para funcionar, por ejemplo, si se escoge libros por autor, el usuario deberá ingresar el autor. Después se muestra el resultado en la terminal, por ejemplo, la opción 1 muestra cuantos libros fueron cargados exitosamente. En general, el usuario interactúa con la aplicación por medio de inputs y la aplicación imprime lo que le debe mostrar al usuario. Las funciones del view.py son o para imprimir algo o para comunicarse con el controlador.
2. Los datos de GoodReads se almacenan en el modelo en la variable catalog, que es un diccionario con llaves “books”, “authors”, “books”, “tags” y “booktags” y cada valor es una lista que contiene diccionarios con la información respectiva de cada objeto. Por ejemplo, cada diccionario en la lista del valor de catalog[“authors”] contiene el nombre del autor, sus libros y su calificación promedio. Estás listas están estructuradas como arreglos.
3. Hay varias funciones que comunican al modelo y a la vista en el controlador. Estas son invocadas en la vista y son en el controlador una invocación de la función correspondiente en el modelo. Estas son: controller.initCatalog(), controller.loadData(catalog), que carga los datos al catalogo invocando varias funciones del modelo ( model.addBook(catalog, book) y las otras semejantes ), la función controller.getBestBooks() que invoca a la función model.getBestBooks() para conseguir los mejores libros del catálogo, la función controller.getBooksByAuthor() que invoca model.getBooksByAuthor para encontrar los autores de un libro y por último controller.countBooksByTag() que invoca model.countBooksByTag(). Estas funciones son utilizadas en la vista y son las que comunicanpor medio del controlador a la vista y al modelo.
4. Para crear una nueva lista se utiliza la función newList, la cual crea una nueva lista. Se pueden especificar parámetros como la estructura de la lista, la función de comparación, el delimitador y el nombre del archivo, si se da un archivo se cargan los datos del archivo en la lista. Si se invoca solo newList() se crea por defecto una lista vacía, simplemente encadenada, separada por coma y con la función comparación por defecto.
5. La función cmpfunction en cada lista es la forma de comparar elementos en una lista y decidir cual es mayor, menor y cuando son iguales. Ya que los elementos de la lista no son siempre números comparables de manera trivial (suelen ser diccionarios). Cuando el parámetro es None se utiliza la función comparadora por defecto.
6. Agrega el elemento (element) en la última posición de la lista (lst) addLast(lst, element). Cambia la lista lst
7. Retorna el elemento que se encuentra en la posición pos de la lista lst cuando se escribe getElement(lst, pos). Pos debe ser menor o igual al tamaño de la lista y mayor a 0. La lista no puede ser vacía.
8. Cuando se invoca subList(lst,pos,numelem) retorna una nueva lista que tiene como primer elemento el elemento de la lista original lst en la posición pos, y tiene en total numelem elementos. Es una copia de los numelem elementos a partir de la posición pos de la lista original lst, manteniendo el orden y la estructura. Por ejemplo subList([1,2,3], 2, 2)=[2,3].
9. Vimos que cuando se corrió con SINGLE\_LINKED los datos se cargaron en 5:30 min mientras que con ARRAY corrió en 8:10 minutos, una diferencia significativa que se puede deber a que los datos ocupaban mucha memoria, que las listas encadenadas aprovechan mejor.