OBSERVACIONES DEL LA PRACTICA

Helena Vegalara Correa 201823328

Juan Esteban Rodriguez 202011178

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

Los mecanismos de interacción entre el **view.py** y el usuario son a través de inputs que genera el **view.py** y que el usuario llena con la información que desea obtener. Los outputs serían entonces los resultados que obtiene el **view.py** (a través de llamar al y **controller.py** al **model.py**), y que le muestra al usuario.

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

En primer lugar, en el **model.py** se crea un catálogo con la utilización de un diccionario. Luego, para cada llave del diccionario, se crea una lista que luego se le va a agregar la información específica para cada dato (ej. Books, authors, tags, book\_tags). En este caso, las listas se crean como arreglos.

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

Las funciones que comunican el **view.py** y el **model.py** son **controller.initCatalog()** y también **controller.loadData(catalog)**, las cuales interactúan directamente con el **controller.py**, y a su vez, este interactúa con el **model.py** a través de las funciones **model.newCatalog()** y demás funciones como **model.addBook(catalog, book)**, **model.addTag(catalog, tag)**, **model.addBookTag(catalog, booktag)**, **model.sortBooks(catalog)**, **model.GetBooksByAuthor(catalog, authorname)**, entre otras funciones. Es decir, la vista y el modelo interactúan a través del controlador, llamando funciones de cada una de los módulos.

1. ¿Cómo se crea una lista?

Se crea una función llamada **newList(datastructure, cmpfunction, key, filename, delimiter)**, en donde se pasan por parámetro todas estas especificaciones que determinan el tipo de estructura de datos (arreglo o lista encadenada simple), la funcion de comparación entre los elementos de la lista, la llave para comparar dos elementos de la lista, el nombre del archivo donde se encuentra la información que se desea agregar a la lista, y el delimitador o separador de campos (caso específico). Con esto, el modelo crea una variable que contiene la lista que se está creando, y utiliza la función **liststructure as lt** gracias al hecho que se importó lo siguiente:

from DISClib.DataStructures import liststructure as lt

Luego de esto, la función retorna la lista creada. En caso de que este proceso falle, la función opta por usar **Exception** y arroja el error correspondiente.

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

En este caso, el parámetro es utilizado como una función de comparación entre los elementos que están en la lista. Como es el caso, no se desea comparar los elementos de la lista creada, entonces se inicializa esta función en None, de manera que no asuma la función por default.

1. ¿Qué hace la funció **addLast()**?

La función **addLast()** adiciona un elemento a la lista en la última posición. Es claro que, para su funcionamiento, necesita que se especifique la lista a donde se quiere agregar el elemento, así como el valor del elemento que se quiere agregar. Nuevamente, se utiliza **liststructure** o **lt** en este caso, para agregar el elemento en la última posición de la lista. Además, tal como en el caso pasado, se hace uso de **Exception**, para el caso en que no sirva la función y no se dañe el código.

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

La función **getElement()** retorna el elemento que se quiera ver, que esté en una posición determinada de la lista. Es claro que esta función necesita dos parámetros para que pueda servir: la lista y la posición del elemento en cuestión que se quiere obtener. Asimismo, esta funcion utiliza **liststructure** o **lt** para obtener el elemento que se quiere de la lista especificada. De igual manera, se hace uso de **Exception**, para el caso en que no sirva la función y no se dañe el código.

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

Esta función retorna una lista dentro de la lista principal. Es decir, crea una lista a partir de una posición de la lista original y hasta un número de posiciones que se quiera obtener. En este caso, la función recibe por parámetros la lista original, la posición inicial de donde se quieren extraer los elementos para la lista nueva, y el número de elementos que se quieren extraer a partir de esa primera posición en adelante para la creación de la sub-lista.

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

No se observa ningún cambio específico en el comportamiento del programa con este cambio, lo cual es lo deseable: los TAD lists son creados de manera que sean fácilmente mutables