Observaciones – Lab3

Grupo 2

Nicolas Rey Lopez - 202321258

Andres Felipe Santana Callejas – 202311502

Santiago Matijasevic Guell - 202225380

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

R/ En el view.py se evidencia que los mecanismos de interacción son el **Input** donde se ingresan los datos y **Output** muestra los resultados por medio del menú y comentarios con print() como se muestra en la siguiente imagen.

```
while working:
    printMenu()
    inputs = input('Seleccione una opción para continuar\n')
    if int(inputs[0]) == 1:
        print("Cargando información de los archivos ....")
        bk, at, tg, bktg = loadData(control)
        print('Libros cargados: ' + str(bk))
        print('Autores cargados: ' + str(at))
        print('Géneros cargados: ' + str(tg))
        print('Asociación de Géneros a Libros cargados: ' +
              str(bktg))

    elif int(inputs[0]) == 2:
        number = input("Buscando los TOP ?: ")
        books = controller.getBestBooks(control, int(number))
        printBestBooks(books)
```

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

R/ El model.py almacena los datos por medio de las funciones: addBook, addBookAuthor, addTag, addBookTag. También, se puede evidenciar por el llamado “add” en todas las funciones para agregar la información depende de los parámetros que se necesiten.

```
def addBook(catalog, book):
```

```
def addBookAuthor(catalog, authorname, book):
    """
```

```
def addTag(catalog, tag):
    """
```

```
def addBookTag(catalog, booktag):
    """
```

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

R/ La funcion loadData y newController porque llaman al controlador para que cargue tanto los datos en el modelo como el view por medio de las funciones mencionadas y mostradas a través de las imágenes.

```
def newController():
    """
    Se crea una instancia del controlador
    """
    control = controller.newController()
    return control
```

```
def loadData(control):
    """
    Solicita al controlador que cargue los datos en el modelo
    """
    books, authors, tags, book_tags = controller.loadData(control)
    return books, authors, tags, book_tags
```

4. ¿Cuál es la función que permite crear una lista?, ¿Qué datos son necesarios?

R/ La funcion newCatalog() es la que crea las listas con los datos de "books" "authors", "tags" y "book_tags". Se crea un diccionario al principio donde se muestra llave: valor. La llave son todos los datos necesarios y en este caso None van a ser los valores que se actualizarán. Además, se crea una lista con cada uno de los datos con su mejor estructura de datos.

```
def newCatalog():
    """
    Inicializa el catálogo de libros. Crea una lista vacia para guardar
    todos los libros, adicionalmente, crea una lista vacia para los autores,
    una lista vacia para los generos y una lista vacia para la asociación
    generos y libros. Retorna el catalogo inicializado.
    """
    catalog = {'books': None,
               'authors': None,
               'tags': None,
               'book_tags': None}

    catalog['books'] = lt.newList('ARRAY_LIST')
    catalog['authors'] = lt.newList('SINGLE_LINKED',
                                    cmpfunction=compareauthors)
    catalog['tags'] = lt.newList('SINGLE_LINKED',
                                 cmpfunction=comparetagnames)
    catalog['book_tags'] = lt.newList('ARRAY_LIST')
    #TODO 4.5 Modificar el uso del TAD lista (p.30.)

    return catalog
```

5. ¿Para qué sirve el parámetro datastructure en la función newList()?, ¿Cuáles son los posibles valores para este parámetro?

R/ El parámetro datastructure determina si la lista será un "ARRAY_LIST" o "SINGLE_LINKED". Esto es para mejorar eficiencia en el programa tanto para tiempo como para memoria. Cada una de estas estructuras de datos tiene sus ventajas, desventajas y su complejidad.

6. Para qué sirve el parámetro cmpfunction en la función newList()?

R/ Esta función básicamente sirve para comparar los elementos de la lista, podemos ver que solo se usa en una lista enlazada simple, se utiliza una función de comparación llamada “compareauthors” como su nombre lo indica compara los autores en la lista.

7. ¿Qué hace la función addLast()?

R/ La función agrega un elemento en la última posición de una lista enlazada, donde se actualiza el puntador a la última posición. Estos son los pasos que hace la funcion addLast:

- Recibe dos argumentos: la lista enlazada a la que se desea agregar el elemento y el elemento que se desea agregar.
- Crea un nuevo nodo que contiene el elemento que se desea agregar.
- Verifica si la lista está vacía. Si lo está, establece el nuevo nodo como el primer y último elemento de la lista.
- Si la lista no está vacía, encuentra el último nodo de la lista.
- Establece el siguiente nodo del último nodo encontrado como el nuevo nodo creado.
- Actualiza el último nodo de la lista para que sea el nuevo nodo creado.
- Devuelve la lista actualizada con el nuevo elemento agregado al final.

8. ¿Qué hace la función getElement()?

R/ La función getElement obtiene el elemento de una lista enlazada simple en una posición especifica.

9. ¿Qué hace la función subList()?

R/ Crea una nueva lista a partir de una que ya existe. También se puede ver de otra forma, permite seleccionar y extraer alguna parte especifica de una lista y devolverla como una lista independiente.

10. Revise el uso de la función iterator() en las funciones printAuthorData(author) y printBestBooks(books) en la Vista que aplican a una lista de libros. ¿Qué hace la función iterator()?

R/ La función iterator retorna un iterador para la lista, que permite recorrer de una lista y filtrara los elementos buscados o pedidos. En este caso, proporciona una forma de recorrer los elementos de la lista de manera eficiente.

```
def printAuthorData(author):
    """
    Recorre la lista de libros de un autor, imprimiendo
    la informacin solicitada.
    """
    if author:
        print('Autor encontrado: ' + author['name'])
        print('Promedio: ' + str(author['average_rating']))
        print('Total de libros: ' + str(lt.size(author['books'])))
        for book in lt.iterator(author['books']):
            print('Titulo: ' + book['title'] + '  ISBN: ' + book['isbn'])
    else:
        print('No se encontro el autor')
```

11. ¿Observó algún cambio en el comportamiento del programa al cambiar el valor del parámetro 'datastructure' en la creación de las listas?, explique con sus propias palabras los cambios que notó.

R/ Al cambiar la estructura de datos en la creación de listas, la carga de la información en el catálogo se demoró bastante y ocupa más memoria que con los anteriores parámetros por lo que ya lo hace ser una solución ineficiente. También, al buscar una cantidad x de libros mejora en la búsqueda al ser más rápido y eficiente. En este caso, en la opción 1 se encarga de la carga de información y la opción 2 mejoro en la búsqueda rápida y efectiva y en la opción 3 no se identifico ningún cambio ya sea por velocidad o memoria.

```
def printMenu():
    """
    Menu de usuario
    """
    print("Bienvenido")
    print("1- Cargar información en el catálogo")
    print("2- Consultar los Top x libros por promedio")
    print("3- Consultar los libros de un autor")
    print("4- Libros por género")
    print("0- Salir")
```