OBSERVACIONES DEL LA PRÁCTICA

Juan David Salguero - 201923136
David Molina - 202125176

Laboratorio SO3, Estructura de Datos y Algoritmos.
Juan David Salguero, David Molina
11 de febrero, 2022

PASO 3

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

La función printMenu() se encarga de mostrar una lista numerada de opciones, dándole al usuario, a manera de output, todas las funciones que el programa ofrece y un número respectivo a cada una.

```
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")
```

print Menu() se muestra al principio de un loop, el cual se mantiene en espera de un input por parte del usuario; se espera que este sea el número correspondiente a la función requerida.

```
while True:
    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()
        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)

    elif int(inputs[0]) == 3:
        authorname = input("Nombre del autor a buscar: ")
```

De acuerdo a la opción seleccionada por el usuario, el view se encarga de llamar una función del controller.view de acuerdo a la función del sistema deseada. Después de conseguir la información, el programa muestra en consola los resultados obtenidos.

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

En primer lugar, el controller crea un diccionario con la función newCatalog(), en el model.py. Este diccionario contiene toda la información del catálogo, en este caso: books, authors, tags y booktags. Y en cada una de estas llaves del diccionario se crean listas de la librería DISCLib ya sean de tipo Array o Single linked.

```
def newCatalog():
    """
    Inicializa el catálogo de libros. Crea una lista vacia para gu
    todos los libros, adicionalmente, crea una lista vacia para lo
    una lista vacia para los generos y una lista vacia para la aso
    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')

    return catalog
```

Luego, usando diferentes funciones del model.py, programadas para la adición de cada tipo de dato, por cada dato encontrado en un archivo csv, se adiciona la información al catálogo correspondiente. En este caso se ejemplifica con loadBooks() que haciendo uso de model.addBook, añade cada libro en el archivo csv al catálogo.

```
def loadBooks(catalog):
    """
    Carga los libros del archivo.  Por cada libro se toman sus aut
    cada uno de ellos, se crea en la lista de autores, a dicho aut
    referencia al libro que se esta procesando.
    """
    booksfile = cf.data_dir + 'GoodReads/books-small.csv'
    input_file = csv.DictReader(open(booksfile, encoding='utf-8'))
    for book in input_file:
        model.addBook(catalog, book)
    return model.bookSize(catalog), model.authorSize(catalog)
```

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

La primera función y tal vez la más importante es loadData(), contenida en el view.py

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

Esta guarda un una tupla los resultados obtenidos de controller.loadData() una función mensajera en controller.py con el mismo nombre.

```
def loadData(control):
    """
    Carga los datos de los archivos y cargar los datos en la
    estructura de datos
    """
    catalog = control['model']
    books, authors = loadBooks(catalog)
    tags = loadTags(catalog)
    booktags = loadBooksTags(catalog)
    sortBooks(catalog)
    return books, authors, tags, booktags
```

controller.loadData(), por medio de otras funciones menores, se encarga de llamar diferentes funciones en model.py, para así permitir a model.py establecer la informacion de la tupla que recibira view.py

```
for book in input_file:
    model.addBook(catalog, book)
return model.bookSize(catalog), model.authorSize(catalog)
```

```
def addBook(catalog, book):
    # Se adiciona el libro a la lista de libros
    lt.addLast(catalog['books'], book)
    # Se obtienen los autores del libro
    authors = book['authors'].split(",")
    # Cada autor, se crea en la lista de libros del catalogo, y se
    # crea un libro en la lista de dicho autor (apuntador al libro)
    for author in authors:
        addBookAuthor(catalog, author.strip(), book)
    return catalog
```

PASO 4

4) ¿Cómo se crea una lista?

En list.py, se llama una funcion newList() que toma como parámetros: el tipo de estructura de la lista, una función que compare los elementos de la lista, un identificador utilizado por la función comparativa descrita anteriormente, un archivo csv del cual se podrían leer información que forme la lista, un símbolo delimitador usado para separar la información en el archivo anteriormente mencionado.

```
def newList(datastructure='SINGLE_LINKED',
            cmpfunction=None,
            key=None,
            filename=None,
            delimiter=","):
    """Crea una lista vacia
```

Usando liststructure.newlist, la función hace un intento de crear la lista.

```
try:
    lst = lt.newList(datastructure, cmpfunction, key, filename,
    return lst
except Exception as exp:
    error.reraise(exp, 'TADList->newList: ')
```

Dependiendo de si se trata de un Array or SingleLinkedList, se crea una de estas dos tipos de lista, a manera de diccionario.

```
 newlist = {'elements': [],
            'size': 0,
            'type': 'ARRAY_LIST',
            'cmpfunction': cmpfunction,
            'key': key
            }
```

5) ¿Qué hace el parámetro cmpfunction=None en la función newList()?

Como se mencionó anteriormente:

```
cmpfunction: Función de comparación para los elementos de la lista.
Si no se provee función de comparación se utiliza la función
por defecto pero se debe proveer un valor para key.
Si se provee una función de comparación el valor de Key debe ser None.
```

6) ¿Qué hace la función addLast()?

Expande el tamaño de la lista en 1, al añadir un elemento tomado como segundo parámetro, a una lista, tomada como primer parámetro. Además se actualiza el apuntador a la última posición de la lista.

```
try:
    lt.addLast(lst, element)
except Exception as exp:
    error.reraise(exp, 'TADList->addLast: ')
```

7) ¿Qué hace la función getElement()?

Recorre una lista hasta la posición 'pos', donde 'pos' > 0 y 'pos' < al tamaño de la lista, 'lst'. Encontrado el elemento correspondiente a la posición dada, lo devuelve al programa.

```
try:
    return lt.getElement(lst, pos)
except Exception as exp:
    error.reraise(exp, 'List->getElement: ')
```

8) ¿Qué hace la función subList()?

Dadas las posiciones 'pos' dentro de una lista, y 'numelem' un número de elementos, la función retorna una lista compuesta por 'numelem' elementos desde la posición 'pos', copiando esta información a la lista retornada.

```
try:
    return lt.subList(lst, pos, numelem)
except Exception as exp:
    error.reraise(exp, 'List->subList: ')
```

9) ¿Observó algún cambio en el comportamiento del programa al cambiar la implementación del parámetro "ARRAY_LIST" a "SINGLE_LINKED"?

Gracias al sistema MVC que usa la aplicación, un cambio en la estructura de los datos no interrumpe el funcionamiento del programa. En este caso todos los componentes de la misma funcionan con regularidad. La diferencia notable es la siguiente:

Prueba con Array_List:

La información toma bastante tiempo en cargar, pero las tres funciones probadas son casi que inmediatas.

Prueba con Single_Linked:

La información se demora demasiado tiempo en cargar, una diferencia significativamente alta comparada con el Array-List.