Universidad de los Andes

Ingeniería de Sistemas y Computación
ISIS1206 – Estructura de Datos
Taller 2

Taller 2: Listas Encadenadas

Objetivos

- Crear implementaciones de estructuras de datos del tipo de dato abstracto Lista.
- Utilizar adecuadamente herramientas para el desarrollo de software en equipos

Lectura Previa

Listas

Las listas son un tipo abstracto de dato que sirven para representar una secuencia de elementos. Cada elemento tiene una posición dentro de la lista. Las listas además pueden tener una posición "actual" de referencia (que comienza en el primer elemento de la lista, en la primera posición). Esta posición de referencia puede avanzar o retroceder sobre la lista y se puede obtener el elemento que está en la posición de referencia actual de la lista. Las listas pueden ofrecer las siguientes operaciones:

- Añadir elemento a una lista
- Quitar elemento de una lista
- Consultar el tamaño de una lista
- Recuperar un elemento de la lista dada una posición
- Iniciar recorrido de la lista (definir el nodo actual como el primer nodo)
- Consultar elemento actual (retornar el elemento actual del nodo actual)
- Avanzar un nodo en la lista (actualizar el nodo actual al próximo nodo)
- Retroceder un nodo en la lista (actualizar el nodo actual al anterior nodo)

Listas Encadenadas

Las listas encadenadas son estructuras de datos recursivas que pueden ser usadas para implementar una lista. En las listas encadenadas cada uno de los elementos que pertenecen a la lista está contenido dentro de un nodo. La estructura contenedora solo tiene una referencia al primer nodo de la lista llamada cabeza.

Existen listas sencillamente encadenadas y doblemente encadenadas, en las listas sencillamente encadenadas cada nodo tiene una referencia al siguiente nodo en la lista. En las listas doblemente encadenadas, cada nodo tiene además una referencia al nodo anterior en la lista.

En las listas encadenadas, la estructura contenedora puede guardar una referencia a un nodo "actual", esta referencia puede avanzar o retroceder sobre la lista.

CSV [1]

El formato de valores separados por comas (CSV, por sus siglas en inglés) se ha utilizado para intercambiar y convertir datos entre varios programas de hojas de cálculo. Si bien hay varias especificaciones e implementaciones para el formato CSV, no existe una especificación formal en existencia, lo que permite una gran variedad de interpretaciones en los archivos CSV. El formato que parece ser seguido por la mayoría de las implementaciones es el siguiente:

1. Cada registro está ubicado en una línea separada, delimitada por un salto de línea (CRLF).
2. El último registro en el archivo puede tener o no un salto de línea final.
3. Puede haber una línea de encabezado opcional que aparezca como la primera línea del archivo con el mismo formato que las líneas de registro normales. Este encabezado contendrá los nombres correspondientes a los campos en el archivo y debe contener el mismo número de campos que los registros en el resto del archivo.
4. Dentro del encabezado y cada registro, puede haber uno o más campos, separados por comas. Cada línea debe contener el mismo número de campos en todo el archivo. Los espacios se consideran parte de un campo y no deben ignorarse. El último campo en el registro no debe ir seguido de una coma.
5. Cada campo puede o no estar entre comillas dobles.

Descripción General

"La accidentalidad vial en Colombia se ha convertido en la segunda manera de muerte violenta y según la Organización Mundial de la Salud (OMS), en el mundo cerca de seis millones de personas mueren por este evento, convirtiéndose así el trafico automotor en la primera manera de muerte violenta en el nivel mundial.[2]" Medicina Legal, Lesiones no Intencionales.

Para prevenir los accidentes viales los gobiernos usan policías de tránsito, cámaras, etc. los cuales producen mucha información. Para que las políticas del gobierno sean más efectivas esta información debe ser tenida en cuenta en la toma de decisiones. Sin embargo, el problema es que la información no siempre es accesible al encargado de tomar las decisiones.

El tema del proyecto está relacionado con la información de infracciones en movimiento emitidas por el Departamento de Policía Metropolitana (MPD) del Distrito de Columbia y agencias asociadas con la autoridad. Esta ciudad está a la vanguardia en el registro de información en diferentes aspectos de su funcionamiento (portal oficial de datos *Open Data DC* http://opendata.dc.gov/). Para el análisis del

[1] Tomado de: https://www.ietf.org/rfc/rfc4180.txt

[2] Tomado de: http://www.medicinalegal.gov.co/documents/20143/49484/Muertes+Transito.pdf/ad2ae841-ed99-d524-66bf-1b70dec0b44a

sistema de información de infracciones en movimiento utilizaremos como fuente de información, el portal Web de consulta http://opendata.dc.gov/datasets?q=moving%20violations.

En este proyecto, vamos a construir una aplicación que le permita entender a los administradores y autoridades de la ciudad de Washington D.C. un conjunto de consultas importantes sobre la información de infracciones en movimiento.

Lo que usted debe hacer

Parte 1 – Trabajo en casa

1. Haga un fork del taller, disponible en el sitio bitbucket (https://isis1206@bitbucket.org/talleres/esqueleto_t2_201910.git). Su taller debe llamarse T2_201910. Si tiene dudas de cómo realizar este paso consulte la guía del Taller1.
2. El taller se debe trabajar **individualmente**. Su primer `commit` al proyecto debe ser un archivo README.txt con su nombre y código.
3. Descargue la información del archivo de infracciones para el mes de enero de 2018 (formato CSV) que se encuentra en el siguiente enlace: http://opendata.dc.gov/datasets/moving-violations-issued-in-january-2018
4. Verifique la estructura del archivo *`Moving_Violations_Issued_in_January_2018.csv`*. En este archivo cada línea tiene la estructura:
    - OBJECTID: Identificador único de la infracción.
    - ROW_:
    - LOCATION: Dirección en formato de texto.
    - ADDRESS_ID: ID de la dirección.
    - STREETSEGID: ID del segmento de la calle.
    - XCOORD: Coordenada X donde ocurrió (No corresponde a longitud geográfica).
    - YCOORD: Coordenada Y donde ocurrió (No corresponde a latitud geográfica).
    - TICKETTYPE:
    - FINEAMT: Cantidad a pagar por la infracción USD.
    - TOTALPAID: Cuanto dinero efectivamente pagó el que recibió la infracción en USD.
    - PENALTY1: Dinero extra que debe pagar el conductor.
    - PENALTY2: Dinero extra que debe pagar el conductor.
    - ACCIDENTINDICATOR: Si hubo un accidente o no.
    - AGENCYID:
    - TICKETISSUEDATE: Fecha cuando se puso la infracción.
    - VIOLATIONCODE: código de la infracción.
    - VIOLATIONDESC: descripción textual de la infracción.
    - ROW_ID:
5. Consulte cómo leer archivos de tipo csv. Una opción es el opencsv.
   Puede encontrar la documentación en http://opencsv.sourceforge.net/

6. Consulte cómo resolver conflictos en git en el siguiente enlace https://githowto.com/resolving_conflicts

Parte 2 – Trabajo en clase

Se desea crear una aplicación que permita entender a los administradores y autoridades de la ciudad de Washington un conjunto de consultas importantes sobre las infracciones en movimiento. En ese sentido usted debe:

1. Seleccionar e Implementar el API de listas encadenadas a utilizar en el desarrollo de este Taller. Esta implementación debe realizarse en el paquete **model.data_structures**.
   - Tenga en cuenta que estas estructuras deben ser utilizadas con cualquier tipo de datos (i.e., deben ser **estructuras genéricas**).
   - Añada en la carpeta **docs** el diseño de la estructura.
   - Añada en la carpeta **data** el archivo *Moving_Violations_Issued_in_January_2018.csv*.
   - Recuerde implementar una clase auxiliar **`Node.java`** donde se almacenará la información de los elementos (genericos) de la lista.
2. Cree el diagrama de Clases que presente el diseño de las Estructuras de Datos a utilizar.
3. Cree (en la carpeta test/src) las pruebas unitarias de las implementaciones de las estructuras de datos, pruebe cada uno de los servicios para las listas.
4. Implemente el método `loadMovingViolations( )` en la clase `MovingViolationsManager` del paquete **model.logic**. Este método debe cargar en una lista encadenada la información proporcionada en el archivo *Moving_Violations_Issued_in_January_2018.csv*.
5. Implemente el método `getMovingViolationsByViolationCode (String violationCode)`, que retorna la lista de todas las infracciones en movimiento dado un código de infracción determinado.
6. Implemente el método `getMovingViolationsWithAccident()`, que retorna el listado de todas las infracciones en movimiento que presentaron un accidente.

Entrega

1. Para hacer la entrega del taller usted debe agregar a su repositorio los usuarios de los monitores y su profesor, siguiendo las instrucciones del documento “Guía Creación de Repositorios para Talleres y Proyectos”.
2. Entregue su taller en su cuenta BitBucket. Recuerde, si su cuenta Bitbucket no contiene el taller o está vacío, su taller no será calificado por más de que lo haya desarrollado.