Universidad de los Andes

Ingeniería de Sistemas y Computación
ISIS1206 – Estructura de Datos
Taller 1

Taller 1: Listas Encadenadas

Objetivos

- Crear una implementación de la Estructuras de Datos Lista Encadenada.
- Utilizar adecuadamente herramientas para el desarrollo de software en equipos

Fecha Límite de Entrega

3 de Febrero, 11:59 p.m.

Lectura Previa

Lista

Lista es un tipo abstracto de dato que sirve para representar una secuencia de elementos. Cada elemento tiene una posición dentro de la lista (a partir de la posición 0). Adicionalmente se puede definir una referencia a un elemento "actual" (que comienza en el primer elemento de la lista). 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. La lista ofrece las siguientes operaciones básicas:

- Añadir un elemento
- Borrar un elemento
- Consultar el tamaño (número de elementos)
- Consultar un elemento dada una posición (el elemento inicial de la lista tiene posición 0)
- Iniciar un recorrido (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. El nodo actual se convierte en null si no hay un elemento siguiente.)
- Retroceder un nodo en la lista (actualizar el nodo actual al anterior nodo. El nodo actual se convierte en null si no hay un elemento anterior.)

Lista Encadenada

Lista encadenada es una estructura de datos recursiva que puede ser usada para implementar una lista. En la lista encadenada cada uno de los elementos que pertenecen a la lista está contenido dentro de un nodo. La lista 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 (excepto el último) y en las listas doblemente encadenadas, cada nodo tiene además una referencia al nodo anterior en la lista (excepto el primero).

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

Descripción General

El taller se define bajo el contexto del proyecto 1 el cual está relacionado con el manejo de información de los comparendos o infracciones al Código Nacional de Tránsito, impuestas por la autoridad policial, generadas en el año 2018 en la ciudad de Bogotá D.C.
"PRUDENCIA es el Sistema integrado de información que facilita la comunicación y el intercambio de información entre los actores (peatón, ciclista, pasajeros y conductores) y los componentes de la movilidad (entidades del sector de movilidad, infraestructura, vehículos, empresas)" [http://www.simur.gov.co/portal-simur/sobre-el-simur/]. Este sistema integrado será la fuente de información que se utilizará para el desarrollo de nuestro proyecto; específicamente utilizaremos la sección de datos abiertos del sistema PRUDENCIA la cual se encuentra disponible en el siguiente enlace: http://www.simur.gov.co/portal-simur/datos-del-sector/datos-abiertos/
En particular, como ya se enunció se utilizarán los datos de los Comparendos del año 2018 en Bogotá, estos datos se pueden consultar y descargar en formato GeoJSON desde:
https://datosabiertos.bogota.gov.co/dataset/comparendos-dei-2018-bogota-d-c

Para fines de este taller, le será suministrado el archivo de datos obtenido del sistema PRUDENCIA.

Fuente de Información en Formato JSON

El formato JSON (JavaScript Object Notation) es un mecanismo de representación de Objetos Java en formato texto. Su contenido se define a partir de texto con la estructura de una clase Java. En general la estructura de un archivo en formato JSON es:

- Una colección de parejas **"propiedad": valor** separadas por "," y contendidas entre corchetes { y } o
- Un conjunto de valores separados por "," y contenidos entre llaves [y]

Por otro lado, el formato **GeoJSON** es un formato diseñado para representar elementos geográficos sencillos, junto con sus atributos no espaciales, basado en el formato JSON.

El archivo de comparendos es *comparendos_dei_2018.geojson* tiene la siguiente estructura:

```
{
"type": "FeatureCollection",
"name": "Comparendos_DEI_2018",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
"features": [
```

{ "type": "Feature", "properties": { "OBJECTID": 1, "FECHA_HORA": "2018\/12\/28", "MEDIO_DETE": "LAPIZ", "CLASE_VEHI": "AUTOMÓVIL", "TIPO_SERVI": "Particular", "INFRACCION": "D02", "DES_INFRAC": "CONDUCIR SIN PORTAR LOS SEGUROS ORDENADOS POR LA LEY. ADEMÁS, EL VEHÍCULO SERÁ INMOVILIZADO.", "LOCALIDAD": "SANTA FE" }, "geometry": { "type": "Point", "coordinates": [-74.080546, 4.600314940000032, 0.0] } },
{ "type": "Feature", "properties": { "OBJECTID": 2, "FECHA_HORA": "2018\/12\/28", "MEDIO_DETE": "LAPIZ", "CLASE_VEHI": "AUTOMÓVIL", "TIPO_SERVI": "Público", "INFRACCION": "H02", "DES_INFRAC": "EL CONDUCTOR QUE NO PORTE LA LICENCIA DE TRASNITO, ADEMAS EL VEHICULO SERA INMOVILIZADO", "LOCALIDAD": "FONTIBON" }, "geometry": { "type": "Point", "coordinates": [-74.141937, 4.697726200000034, 0.0] } },
…
]
}

Una descripción más completa de esta fuente de información la encuentra en el enunciado del proyecto 1 sección Fuente de Datos.

Para la lectura de archivos GeoJSON se recomienda usar el API (librería) GSON.
Consultar https://github.com/google/gson.
Para descargas del API GSON (librería extensión .jar) consulte:
https://search.maven.org/artifact/com.google.code.gson/gson/
Para documentación del API GSON consulte:
http://www.javadoc.io/doc/com.google.code.gson/gson
Como material de tutorial consulte:
https://www.studytrails.com/java/json/java-google-json-introduction/
Como ejemplo de lectura de un archivo JSON propiedad por propiedad consulte:
https://www.studytrails.com/java/json/java-google-json-parse-json-token-by-token/

Lo que usted debe hacer

Parte 1 – Trabajo en casa

1. Haga un *fork* del taller base (taller 0) disponible en el URL Github https://github.com/isis-1206/esqueleto_T0_202010.git. Su taller debe llamarse **T1_202010**. Si tiene dudas de cómo realizar este paso consulte la guía del taller 0.
2. El taller se debe trabajar **individualmente**. Su primer `commit` al taller debe ser un archivo README.txt con su nombre y código.
3. Descargue la información del archivo de comparendos 2018 de la ciudad de Bogotá y cópielo en la carpeta `data` de su proyecto Eclipse. Archivo: *comparendos_dei_2018.geojson*
4. Verifique la estructura del archivo. En este archivo los comparendos están contenidos en la propiedad "**features**" cuyo valor contiene un arreglo de comparendos. El primer comparendo se define por:

{"type": "Feature", "**properties**": {"OBJECTID": 1, "FECHA_HORA": "2018\/12\/28", "MEDIO_DETE": "LAPIZ", "CLASE_VEHI": "AUTOMÓVIL", "TIPO_SERVI": "Particular", "INFRACCION": "D02", "DES_INFRAC": "CONDUCIR SIN PORTAR LOS SEGUROS ORDENADOS POR LA LEY. ADEMÁS, EL VEHÍCULO SERÁ INMOVILIZADO.", "LOCALIDAD": "SANTA FE" }, "**geometry**": { "type": "Point", "**coordinates**": [-74.080546, 4.600314940000032, 0.0] } },

El detalle de un comparendo se define en su propiedad "**properties**". Esta propiedad contiene un subconjunto de propiedades, de las cuales las principales son:

- OBJECTID: Identificador único del comparendo
- FECHA_HORA: Fecha del comparendo en formato Año\/Mes\/Día
- CLASE_VEHI: tipo de vehículo (“AUTOMOVIL”, "BICICLETA", "BUS", "BUSETA", "CAMIONETA", "CAMPERO", "MOTOCICLETA", …)
- TIPO_SERVI: tipo de servicio ("Particular", "Público", “Oficial”)
- INFRACCION: Código de la infracción cometida
- DES_INFRAC: Descripción de la infracción
- LOCALIDAD: Localidad en la ciudad del comparendo

La información de su localización geográfica se define en su propiedad "**geometry**" la cual contiene a su vez la propiedad "**coordinates**" con su longitud y latitud geográficas.

5. Consulte cómo resolver conflictos en Git en el siguiente enlace https://githowto.com/resolving_conflicts

Parte 2 – Trabajo en clase

El propósito es construir una aplicación Eclipse/Java que nos permita realizar el proceso de lectura (carga) de un archivo con formato JSON, y posteriormente realizar análisis y búsquedas de información sobre el mismo. En ese sentido usted debe:

1. Diseñar e implementar el API de lista encadenada propuesto con las operaciones básicas. Ud. debe decidir si hace una implementación de lista sencillamente encadenada o doblemente encadenada. Esta implementación debe realizarse en el paquete **model.data_structures**.
   - Realice el diseño de la estructura de datos (imagen UML del diagrama de clases) y agreguelo en la carpeta **docs**.
   - Tenga en cuenta que esta estructura puede ser utilizada con cualquier tipo de datos (i.e., debe ser una **estructura genérica**).
   - Recuerde implementar una clase auxiliar **`Node.java`** donde se almacenará la información de los elementos (genéricos) de la lista.
2. Implemente las pruebas unitarias de la estructura de datos en el paquete **test.data_structures**; pruebe cada uno de sus servicios (métodos).
3. Modifique el menú de servicios que ofrece la clase **Controller** del paquete **controller**. Los servicios que debe ofrecer la aplicación son:
   - Opción 1: Realizar la carga de los comparendos de la ciudad de Bogotá para el periodo 2018.

La carga debe implementarse en un método en la clase **Modelo** del paquete **model.logic**. Los comparendos deben quedar almacenados en una lista encadenada en el orden en el que se leen del archivo.

Como información al usuario debe mostrarse la información básica del primer comparendo y del último comparendo en la lista y el total de comparendos en la lista.

- Opción 2: Consultar la información básica de un comparendo dado su OBJECTID. Esta consulta debe implementarse en un método en la clase **Modelo** del paquete **model.logic** y debe retornar la información del comparendo resultante. El OBJECTID de consulta debe ser un parámetro del método.

  Como información al usuario debe mostrarse la información del comparendo: OBJECTID, FECHA_HORA, INFRACCION, CLASE_VEHI, TIPO_SERVI, LOCALIDAD.

  Reportar el caso especial en que No exista información del comparendo.

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 Github.** Recuerde, si su repositorio del taller NO queda accesible en su cuenta Github a los monitores y profesor de su sección, su taller no podrá ser calificado.