Overview

Write a Java library to retrieve and transform an XML based traffic incident feed to a JSON version of that feed.

Input Format Example

Below is an example extract from the full feed representing the two types of locations from the data set. Your program must process the full data set from the URL given later in this document.

```
<?xml version="1.0" encoding="UTF-8"?>
<incident>
  <ti table='120' tv='2Q2016' fv='1.0' at='2019-03-31T21:12:15Z'>
    <ev>
      <id>23317403</id>
      <ec>702</ec>
      <se>3</se>
      <loc type='geo'>
        <geo lon='-74.00625' lat='40.717075'/>
        <addr>Worth St</addr>
      </loc>
      <valid start='2019-02-10T05:00:00Z'
             end='2019-04-13T03:58:59Z'/>
      <text lang='e'>In Manhattan, major road construction on Worth
St between W Broadway and Church St.</text>
      <delay>8</delay>
    </ev>
    <ev>
      <id>24049284</id>
      <ec>701</ec>
      <se>2</se>
      <loc type='tmc'>
        <start id='16078' dir='+' offset='86' extent='14'/>
      </loc>
      <valid start='2019-03-02T05:00:00Z'
             end='2019-04-02T03:58:59Z'/>
      <text lang='e'>In Bronx, road construction on Creston Ave
between Minerva Pl and E 198th St.</text>
      <delay>0</delay>
```

```
    </ev>
    .
    .
    .
  </ti>
</incident>
```

Field	Description
id	A unique ID for the incident.
ec	Event code.
se	Severity.
loc	Each traffic incident, <ev>, has one of two location types, “geo” or “tmc”. For “geo” the location is given in coordinates. For “tmc” the location is given by a reference to the road by a map ID (and other additional information).
valid	The start and end times of the incident.
text	A plain text description of the incident.

Data Set Source

Your program must retrieve and process the data from this URL:

https://sxm-dataservices-samples.s3.amazonaws.com/incidents.xml.gz

Do not download and include the file in you submission. Your program must read the file from this URL at runtime.

Output Format Example


```
{
  “locations”: [
    {
      "_id": "224155332",
      "description": "In Greenview No. 16, animals on roadway on AB-40 EB
etween  Forestry Trunk Rd and Pierre Greys Lake Rd.",
      "geo": {
        "type": "Point",
        "coordinates": [
          -118.67265,
          53.924755
        ]
      },
      "roadName": "AB-40",
      "eventCode": 922,
      "severity": 2,
      "validStart": "2018-10-26T13:39:58.000Z",
      "validEnd": "2019-07-20T00:51:19.000Z",
      "type": "TrafficIncident",
      "lastUpdated": "2019-07-20T00:23:12.748Z"
    },
    {
      "_id": "223155366",
      "description": "In Gibson, object on roadway on PA-120 EB between Church
t and Church St.",
      "tmc": {
        "table": 4,
        "id": 12915,
        "direction": "+",
      }
      "eventCode": 61,
      "severity": 2,
      "validStart": "2018-03-28T21:27:14.000Z",
      "validEnd": "2019-07-20T00:54:26.000Z",
      "type": "TrafficIncident",
      "lastUpdated": "2019-07-20T00:26:16.321Z"
    }
  ]
}
```

Field	Source Field	Description
_id	id	
description	text	
geo	loc.geo id “geo”	GeoJSON formatted location
tmc	loc.start if “tmc” table is determined by the value in the “table” attribute in the parent <ti> element	
eventCode	ec	
severity	se	
validStart	validStart	
validEnd	validEnd	
type		Always “TrafficIncident”
lastUpdated		The time the output file was generated formatted as ISO 8601.

Requirements

- Assume your library will be used in a production environment. It should be structured and written to be maintainable, flexible and extensible. Anything you wouldn’t normally do in such a situation please make note of in a comment (it’s fine, this is just an exercise and we won’t really be using it in a production environment).
- All code must be Java 8 compatible. Please use any Java 8 features where appropriate (Lambdas, Generics, concurrency, etc.). But, don’t use them just to use them.
- Use any 3rd party open source library you would like.
- Your library must retrieve the data from the given URL at runtime. Do not simply download the static file and package it with your library or assume it will exist locally.
- Provide a main() method that demonstrates the usage of your library. Only include code here to drive the library for the demonstration. The library itself should do all of work required.
- Provide a build script that produces a runnable JAR file (e.g. java -jar transform.jar).
- If you have any questions please ask.