Assignment 1

Report

# Pre-documentation

## XML schema description

For the 1994Feb XML File:

1. The root element is named sales.
2. Within this element there are many receipts elements
3. These receipt elements have an id attribute which is unique.
4. They also have a when element and format attribute to tell the date of the sale with the correct format.
5. Receipt element also has a where element which specifies location of the sale, and the id attribute has a unique identifier.
6. Where element has a name, address, and phone element. This describes the name, address and phone of the location.
7. The receipt element also has an items element with an id attribute.
8. Item element contains a name, price, currency, quantity and tax percentage element. To specify the name of item, price of item, currency used to purchase the item, the quantity of items bought, and the tax applied to the item.

The above constraints ensure that each "receipt" element contains all the necessary information about a sale, including the date, seller, location, and items sold, with unique identifiers for each element to avoid duplication or confusion. The constraints also ensure that all necessary information is included for each "item" element, including the name, price, quantity, and tax percentage.

## JSON structure recommendation

I would use a collection of name/value pairs for JSON.

In the JSON representation, each "receipt" element is within an object, which includes all the necessary information about the sale, such as the ID, date, seller, location, and items sold. Within each "receipt" object, the "when," "seller," "where," and "items" elements are encapsulated within their own objects, with attributes and values as appropriate.

Each "receipt" element in the JSON representation is contained within an object that contains all the required details about the sale, including the ID, date, seller, location, and the items sold. The "when," "seller," "where," and "items" elements are contained within separate "receipt" objects, each with the proper attributes and values.

Overall, this structure streamlines the data without omitting any crucial details and presents it in a more readable, manageable manner.

# Post-documentation

The task given was to parse 3 XML files using SAX or DOM and store it as JSON objects.  
After a lot of trial and error I was able to do the required task. The SimpleParser java file has 4 methods.

The main method takes 2 arguments. Since, the task I was given was to use 2 arguments: the XML file path and JSON file path. The code is in such a way that if the user inputs 1 argument, then it gives and output: ‘2 arguments needed.’ However, if 2 arguments are given it parses XML file into JSON objects using the next method ‘parseXMLtoJSON’.   
  
The parseXMLtoJSON method is responsible for parsing the XML file and creating a JSON object. It uses DOM. I used DOM because it parses XML into a tree structure, this makes it easier to navigate since I use recursion too. I could’ve uses SAX too since the XML files have a lot of repeating elements. DOM was personally easier for me to use[[1]](#footnote-1). Furthermore, the method does is obtains root elements and stores it in a JsonObject. The attributes of the root element are then added to the JsonObject, and the child nodes are processed using recursion.

The processNode method is called in the parseXMLtoJSON method. It recursively traverses the nodes of the XML document and builds a JSON object[[2]](#footnote-2). It adds the values of child nodes to a StringBuilder and creates a new JsonObject for each element node encountered. If the child node has the same name as an existing child node, the method adds the new object to an existing JsonArray[[3]](#footnote-3). Otherwise, it adds the new object to the JsonObject.  
  
This whole thing is done using GSON which stores the object and stores them as strings.[[4]](#footnote-4)

The saveJSON method uses the Files class to write the JSON string to the specified JSON file path.

I tried to make the Java code as concise, simple, and readable. Since, I used DOM it may not be able to handle all complex and very large XML data.

1. https://www.youtube.com/watch?v=2JH5YeQ68H8 [↑](#footnote-ref-1)
2. https://stackoverflow.com/questions/48189766/java-xml-parsing-recursively-find-element [↑](#footnote-ref-2)
3. https://stackoverflow.com/questions/35148704/java-reading-nodes-recursively-from-xml-returns-just-text-nodes [↑](#footnote-ref-3)
4. https://www.youtube.com/watch?v=cclaCWf2i-M [↑](#footnote-ref-4)