COIN-078B : Internet Programming with XML

Assignment 4

Objectives:

- Learn how to handle XML documents with Java using the Document Object Model (DOM) interfaces.
- Load, parse and transform XML documents to graphical format in Java.

Alternative: If you are familiar with SQL and JDBC, you may choose to skip this specs and work on converting the elements.xml to a SQL table using DOM and an additional 5 pts. extra credits using SAX also.

Note: Please follow the specifications below. Points will be knocked off if not followed.

Read Assignment Guidelines & Requirements

Sun J2SE 1.4 and later includes the Java API for XML Processing (JAXP) that is used for processing XML data using applications written in the Java programming language. JAXP leverages the parser standards SAX (Simple API for XML Parsing) and W3C DOM (Document Object Model) so that you can choose to parse your data as a stream of events or to build an object representation of it. In this assignment you are going to write a standalone application - a graphical browser in Java that reads an XML document and extracts its data to draw text and graphics figure such as circles. The browser will process and display the elements.xml document in a tabular format like in assignment 1 except that you don't use XSLT nor HTML. The presentation output is in graphical form using the Java AWT and Graphics APIs. Since the non-Microsoft browsers do not support VML, you are going to use the Java API to create the filled and hollow circles that represent the elements size and state. The textual data is also drawn graphically in a tabular fashion. See the screenshots. The units on the header must be extracted from the XML. Don't do the alternating colors.

This assignment contains **3 classes** all in one source file named **Assign4.java**:

class Assign4

- primary class containing the `main()` method and other static methods. It basically contains static arrays of the fields to be displayed.

- **Class variables:** (static variables)
  - **name** - `String[]` of atom names
  - **symbol** - `String[]` of atomic symbol
  - **atomic_number** -`int[]` of atomic number
  - **atomic_weight** - `double[]` of atomic weight
  - **melting_point** - `double[]` of melting point
  - **boiling_point** - `double[]` of boiling point