3. Unit: XML Processing with Java (I)

The first four exercises are "typical" exercises to get into the new technologies:

Exercise 3.1 (DOM Basics)

Parse mondial.xml into a DOM instance and implement the following query based on the DOM operations (do not apply XPath in DOM):

For all organisations that have their headquarter in the capital of a member country, output the name of the organisation and the name of the headquarter (to System.out).

Exercise 3.2 (DOM: Creation of a Statistics Table)

Create an HTML table as a JDOM object (and write it finally into a file) that gives, in steps of 100,000, how many cities exist that have between n00.000 and (n+1)00.000 inhabitants, and how many inhabitants each of these groups has. Note: also is there is not city in such a step, return a line with a "0".

0 - 100000	275	16120153
100000 - 200000	1074	154777116
200000 - 300000	556	134546479
:	:	:
22200000-22300000	0	0
22300000-22400000	1	22315474

For computing the counts and numbers from Mondial, XPath can be used (e.g., read Mondial into a JDOM object and apply XPath to it; you can also use any other package for this if you want).

Where would be a problem to create the same table in XQuery or XSLT?

Exercise 3.3 (SAX: Queries against Mondial)

Write SAX Event Handlers in Java for the following tasks:

a) Output an HTML file that lists the names of all countries in mondial.xml.

b) Output the population of the capital of Germany via System.out.

c) use the previous part of the exercise to output all country names, the country's capital and the country capital's population – if available – into an HTML table.
Example:

country	**capital**	**capital population**
Albania	Tirane	192000
Greece	Athens	885737
:	:	:

- First, simply output the table values as text to System.out.
- Create an HTML table as a JDOM object with the result during the SAX run and output it into a file.

d) For each country in mondial.xml, output an HTML table containing the names and – if present – the most recent population count for each city in the country. Use a <ul> (unordered list) environment with one list item per country.
Example:

- ...
- **Germany**

Stuttgart	588482
Mannheim	316223
Karlsruhe	277011
...	...

- ...

e) Modify the event handler of the previous part of the exercise to output the following for each country with at least than 10 valid city population entries:
   - the country's name
   - the overall number of cities
   - each city with name and most recent population count
   - the average city population
   - inside the city table, mark (either by color, font etc) (1) the capital city, and (2) the city whose population is closest to the average city population of the country.

Exercise 3.4 (StAX: Queries against Mondial)

a) Implement parts a) and e) as given in the SAX exercise, now by using StAX.
Reuse the program code as far as possible, and adapt it only to the StAX frame.

b) Implement the following query in StAX: For all organisations that have their headquarter in the capital of a member country, output the name of the organisation and the name of the headquarter.

c) Playing with streams: implement a pipe such that (b) creates results of the form

```
<result>
 <organization name="European Union"/>
 <city name="Brussels"/>
</result>
```

that are written into another XMLOutputStream. Create a second thread that reads from this stream and filters (via StAX) only the <city> elements. Let both threads log to System.out to show the concurrent processing.

Usecase Scenario: Calexit.

The second group of exercises implement a "realistic" use case for an update to Mondial (recall that XQuery and XSLT cannot be used to *change* a document directly).

Consider the case that California leaves the U.S.A. Then, some updates must be executed on Mondial.

- plan first your strategy, before starting to program,
- consider especially, what kinds of data items must be changed, and how,
- write the programs as generic as possible (such that they can be applied also whenever some other province turns into a new, independent country).
- Update mondial.xml appropriately, especially:
- take as much data as possible from Mondial,
- California becomes a member of all organizations where the USA are a member, except G-5 and G-7.
- Ignore the airport elements at the end of Mondial.
- The below fragment contains all necessary new facts about California:

```
<population_growth>1.0</population_growth>
<infant_mortality>4.5</infant_mortality>
<gdp_total>1810000</gdp_total>
<gdp_agri>2</gdp_agri>
<gdp_ind>23</gdp_ind>
<gdp_serv>75</gdp_serv>
<inflation>1.8</inflation>
```

```
<unemployment>5.5</unemployment>
<ethnicgroup percentage="72.9">European</ethnicgroup>
<ethnicgroup percentage="6.5">African</ethnicgroup>
<ethnicgroup percentage="14.7">Asian</ethnicgroup>
<ethnicgroup percentage="1.7">Amerindian</ethnicgroup>
<religion percentage="32">Protestant</religion>
<religion percentage="28">Roman Catholic</religion>
<religion percentage="2">Jewish</religion>
<religion percentage="2">Buddhist</religion>
<religion percentage="1">Mormon</religion>
<religion percentage="1">Muslim</religion>
<language percentage="60.5">English</language>
<language percentage="25.8">Spanish</language>
<language percentage="2.6">Chinese</language>
<border country="MEX" length="226"/>
<border country="USA" length="1681"/>
```

Exercise 3.5 (Calexit in JDOM)

- read mondial.xml into a JDOM object,
- update it using the JDOM operations (you can use XPath in the JDOM for searching for values or nodes),
- write it out into a file,
- validate the result file against mondial.dtd.

Exercise 3.6 (Calexit in XSLT)

Solve the previous exercise with an XSLT transformation.

Exercise 3.7 (Calexit in SAX/StAX)

Solve the previous exercise with a SAX or StAX transformation, again writing the result out as XML into a file.

Hints:

- for the JDOM case, you must explicitly change some things; all others stay unchanged.
- For XSLT and SAX/StAX, you have also to deal with the unchanged portions. Handle this with as little effort as possible, using general rules (XSLT templates and SAX/StAX conditions). Note that these two strategies are closely related.
- Focus on the general, structural issues; don't spend too much time for the dirty details of the new values for the USA.
- My solution in XSLT just contains 9 templates (80 lines) for the general handling, and another 8 templates (80 lines) for dealing with a reasonable computation of the new USA values.