2. Document Navigation with the Java DOM Interface and with XPath

The purpose of this lab is

1. to experiment with the DOM binding in Java
2. to write queries in XPath.

2.1 The DOM Interface in Java

The Java API for XML Processing (JAXP) contains a Java version of the DOM interface for XML. The goal of the first part of this lab is to try out DOM to

- navigate in XML documents,
- print documents,
- create documents.

2.1.1 Flat Navigation in an XML document

Write a class that reads an XML file and counts the number of children of the top level element. Follow the example of the class `DOMTest` on the course page.
Test your code with the document `recipes.xml`.

2.1.2 Printing Documents

Add a method `printDocument(Document doc)` to your class that takes as input a DOM document and prints it onto standard ouput. Print the document.
Use the Transformer interface. (You may rely on the code on the course page.)

2.1.3 Modifying Documents

Take an XML document of your choice. Write a class that

1. clones a top-level child and adds it to the root element,
2. changes an attribute value of that new child.

Test your code with the `printDocument` Method.

2.1.4 Generating XML Documents

A “family” document has the following structure:

- the top level element is tagged “family”
- the top level element has several children tagged “person”
- each person element has an attribute “role” and text content, standing for the name of the person.

Create an XML document with two persons whose roles are “father” and “mother”. Test your code with the `printDocument` Method.

2.1.5 Deep Navigation in an XML document

Write a class that reads an XML document and reports

1. the number of element nodes
2. the number of attribute nodes
3. the number of text nodes.

You may find some inspiration in the `DOMTest` example.

2.2 XPath Queries

Hint: You can run XPath queries in Eclipse in the XML perspective, using the XPath view. To open that view, go to

```
Window --> Show View --> XPath
```

If you do not see `XPath` as one of the options under `Show View`, go to `Other...` and open the XML folder, where you find an XPXath Symbol to choose.

Countries

Open the `countries.xml` file in Eclipse. Write XPath expressions that return answers to the queries below.[1] To formulate some of queries, you may need to consult the slides of the XPath lecture for built-in functions.[2]

1. Return the area of Mongolia.
2. Return the names of all countries with population greater than 100 million.
3. Return the names of all countries where over 50% of the population speak German. (Hint: Depending on your solution, you may want to use "`.`", which refers to the "current element node" within an XPath expression.)
4. Return the names of all countries where a city in that country contains more than one-third of the countrys population.
5. Return the population density of Qatar. Note: Since the "`/`" operator has its own meaning in XPath, the division operator is infix `div`. So to compute population density use "`(@population div @area)`".
6. Return the names of all countries whose population is less than one thousandth that of some city (in any country).
7. Return the names of all cities that have the same name as the country in which they are located.
8. Return all city names that appear more than once, i.e., there is more than one city with that name. You may return a city name multiple times if it simplifies your query. (Hint: You might want to use the "`preceding`" and/or "`following`" navigation axes for this query.)
9. Return the names of all countries which have a city such that some other country has a city of the same name.
10. Return the number of countries where Russian is spoken.

[1]These data have a tradition in XML teaching. They originate from the Mondial database, assembled by Wolfgang Mey at the University of Goöttingen. The current, simplified version has been created at the Stanford InfoLab, building upon a version created at the University of Washington, Seattle.

[2]The queries and the exercises are taken from classes at Stanford.

11. Return the names of all countries that have at least three cities with population greater than 3 million.

12. Return the names of all countries for which the data does not include any languages or cities, but the country has more than 10 million people.

The last four queries are four different variations on the same theme. Consider the built-in function `contains`.

13. Return the names of all countries whose name textually contains a language spoken in that country. For instance, Uzbek is spoken in Uzbekistan, so return Uzbekistan. (Hint: As in question 3, you may want to use "`.`", which refers to the "current element" within an XPath expression.)

14. Return the names of all countries in which people speak a language whose name textually contains the name of the country. For instance, Japanese is spoken in Japan, so return Japan.

15. Return all languages spoken in a country whose name textually contains the language name. For instance, German is spoken in Germany, so return German. (Hint: Depending on your solution, you may want to use a value type casting function like "`string(.)`", which returns the string value of the node set denoted by an XPath expression. Note that the function "`data(.)`" exists only in XPath 2.0.).

16. Return all languages whose name textually contains the name of a country in which the language is spoken. For instance, Icelandic is spoken in Iceland, so return Icelandic.