XML Technologies: Exercise 2

DTDs, Namespaces, Encodings

This is a mandatory exercise and the result will be part of your final mark. The solution must be uploaded to OLAT by **March 11th at 15:59**. Late submissions will not be accepted.

Submit the following files in a zipped archive:

- distilleries.xml
- distilleries.dtd
- 2a.xml, 2b.xml, 2c.xml, 2d.xml, 2e.xml
- japanese_distilleries.xml

Name the archive [lastname]_[firstname]_2.zip (for example *mueller_mathias_2.zip*). Some parts of your submission may be automatically evaluated, so make sure to name your files *precisely* as prescribed, otherwise you might not get any points.

1 Document Type Definitions (DTDs) revisited

1.1 Whisky Distilleries (2 Points)

The DTD `distilleries.dtd` attached defines the structure of an XML file representing a collection of whisky distilleries. Read the definition carefully before moving on. Also notice how the definitions are arranged: Usually, DTDs start with the definition of the root element and are then written starting with the outermost elements. The arrangement of attributes is considered a matter of taste.

Write an XML file that is a *valid* instance of the DTD above. The content you may make up, but the structure has to conform to the DTD! You can check validity with `xmllint --noout --valid distilleries.xml`. Make sure you use the appropriate declaration in the header of the XML file:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE distilleries SYSTEM "distilleries.dtd">
```

Also, to facilitate correction, include the original `distilleries.dtd` in your submission.

1.2 Better Distilleries (0 Points)

The DTD above illustrates how limited DTDs are. Often, we would like to enforce more complex constraints, as in the DTD comment:

```
<!--The minimum age for Single Malt from Scotland must be 3 years -->
```

Whereas DTDs are incapable of capturing constraints like these, XML Schemas are. They will be introduced later in the course. So, in due time we will be able to rewrite a proper XML whisky database.

Do not submit anything for this purely informational subsection.

2 Namespaces (2 Points)

Namespaces are a powerful aspect of XML. Any element or attribute that is *prefixed* as follows

```
<p:element p:a="done" xmlns:p="www.p.com"/>
```

is said to be in a namespace. If a default namespace is set in one of the enclosing elements, elements without a prefix are also in that namespace. And then, of course, there is the *null* namespace.

Attached you find 5 documents called `2a.xml`, `2b.xml`, `2c.xml`, `2d.xml` and `2e.xml`. They all use namespaces, but not all of them use them correctly.

For every file, insert an XML comment at the beginning of each file, either explaining that namespaces were used correctly:

```
<!--Correct use of namespaces.-->
```

Or explaining why namespaces were misused:

```
<!--Incorrect use of namespaces because...-->
```

If there are mistakes, correct them so all the files are well-formed.

3 Encoding Japanese Whiskies (1 Points)

There are two aspects to character encoding (both in XML and programming in general). On one hand, there are different *character sets* or *repertoires*, which are finite sets of abstract characters.

On the other hand, there must be a mapping, called *encoding*, between the abstract characters and their physical representation on disk. The most widely used example of this is *Unicode*, which defines both a repertoire and an encoding.

XML is designed to be equally comfortable with any character set, and any kind of encoding.

Save a copy of your `distilleries.xml` as `japanese_distilleries.xml`. Add as an additional element the famed Yamazaki distillery, the information you which you'll find in the attached `yamazaki.txt`. Choose an appropriate encoding and save the file. (While XML deals very well with Unicode, Latex does not!).