5. XML Schema (2)

In the exercise of Sheet 4, you have created an XML schema for XML documents about juicers. This sheet contains further exercises that build upon that first one.[1]

Hint: In all the exercises, use the Eclipse "design" view as far as possible to create your schema.

1. Schema in the Russian Doll Style

You designed the first juicers schema by translating a DTD. By this approach, you created a schema in the Salami Slice style, where each element is declared by a *global* declaration.
As an alternative, create another equivalent schema in the Russian Doll Style, where you nest all elements as deeply as possible (use again `juicers.dtd`).
Validate (1) your new schema and (2) validate the `juicers.xml` against the new schema.

2. Use More Meaningful Built-In Types

Using a DTD, one can only specifcy the elements with atomic content as containing `#PCDATA`. By translating the DTD into XML schema, we have inserted the built-in type string instead of `#PCDATA`.
XML Schema, however, offers a large number of different built-in types. In this exercise, you are to change the datatypes from string to another built-in datatype, wherever it makes sense. For instance, `weight`, `cost`, and `retailer` can be declared with a much more relevant datatype than string. (Check the document to see how to do this.)
As before, validate the instance document against your schema.

[1] The exercises are based on the labs in the XML Schema tutorial by Roger L. Costello, `http://www.xfront.com/xml-schema.html`

3. Creating a New Datatype by Restriction

Modify the schema that you created in the previous exercise. Create a new datatype called `money` and declare the juicer `cost` element to be of that type.
Hint: Here are the facets for the built-in datatype `decimal`:

`totalDigits`: the total number of digits allowable in the number (including the digits to the right of the decimal point)

`fractionDigits`: the number of digits allowed to the right of the decimal point

`pattern`: regular expression specifying the possible values

`enumeration`: one of the possible values

`whitespace`: instructs the XML processor how to deal with white space; possible values are `preserve`, `replace`, or `collapse`

`maxInclusive`, `maxExclusive`, `minInclusive`, `minExclusive`: self-explanatory

4. Restricting a Datatype Using Regular Expressions

Modify the schema further that you created in Exercise 3. Note that the `image` element contains a string that specifies an image file in the local file system.
Create a new datatype called `imageFile` and declare the `image` element to be of that type. Define a regular expression for `imageFile`. (Consult the lecture slides to understand how to write regular expression in XML Schema.)
Again, validate the instance document against your schema.

5. Extending Datatypes

Modify the schema that you created in Exercise 4.
Create a type called `appliance`. Define `appliance` to contain declarations for `description` and `warranty`. Create a type called `juiceAppliance`, which is derived from `appliance` (by extension). Make `juicer` of type `juiceAppliance`.
Validate the instance document against your schema.

Note: You will need to rearrange the contents of each `juicer` in the instance document!

6. Schemas with Attributes

Consider now an extension of our original DTD, where the element `juicer` has attributes.

```
<!ELEMENT juicers (juicer)*>
<!ELEMENT juicer (name, image, description, warranty?,
                  weight?, cost+, retailer)>
<!ATTLIST juicer id ID #REQUIRED
                 electric (true|false) #REQUIRED
                 type (press | gear | centrifugal) #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT image (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT warranty (#PCDATA)>
<!ELEMENT weight (#PCDATA)>
<!ELEMENT cost (#PCDATA)>
<!ELEMENT retailer (#PCDATA)>
```

Create corresponding attributes in the XML schema. Validate the document `juicers6.xml` against the new schema.

7. Extending Simple Types with Attributes

Consider a further extension of the DTD, where the element `cost` has an attribute `currency`.

```
...
<!ELEMENT cost (#PCDATA)>
<!ATTLIST cost currency (USD | CAD) #REQUIRED>
...
```

Create a corresponding attribute in the XML schema. Note that in this case, you have to extend a simple type (`money`) with an attribute.
Validate the document `juicers7.xml` against the new schema.