CS 548—Fall 2018
Enterprise Software Architecture and Design
Assignment Two—XML Schemas and JAXB

Design the XML Schema for a medical information system. You should provide the schema definition itself, structured as a collection of namespaces for the different form of documents, as well as instance documents that exhaustively demonstrate the different forms of documents that may be in such a system.

A clinic database consists of a collection of patient records, one for each patient. Each patient has:
1. a patient identifier (that may be assigned for example by a national government),
2. a patient name.

Furthermore each patient is related by a one-to-many relationship to a collection of treatment records.

Every treatment record includes a diagnosis of the condition for which the treatment is prescribed (e.g. throat cancer, HIV/AIDS, hepatitis, etc). Currently there are three specific forms of treatment records:
1. A drug treatment record includes a drug and a dosage.
2. A surgery treatment record includes a date of surgery.
3. A radiology treatment record includes a list of dates of radiology treatments.

The diagnosis, patient and provider administering the treatment can be common fields for all types of treatment records. Patients and providers are identified by their database identifiers.

Every healthcare provider has a record that includes their provider identifier (NPI), their name, and an indication of their specialization (surgery, radiology, oncology, etc). Each provider may be related to several treatment instances. We will represent the patient treatment and provider-treatment relationships using foreign keys.

Your task is to define an XML schema for these entity types. There are two different forms of solutions, as described below.

For your first solution, use the *choice element* of XML schemas to represent the choice among the different forms of treatment records. Define a generic treatment record that specifies the information common to all treatment types, and use the choice element to represent the different forms of information that are recorded for the different types of treatment:

```
<element name="treatment-dto">
  <complexType>
      <sequence>
        <element name="id" type="long" />
        <element name="diagnosis" type="string" />
        <element name="patient" type="long" />
        <element name="provider" type="long" />
        <choice>
            <element name="drug-treatment"
```

```
                    type="tns:drug-treatment-type" />
                ...
            </choice>
        </sequence>
    </complexType>
</element>
```

For your second solution, use *element substitution* to represent the choice among the different forms of treatment records. Define a generic treatment record whose content specifies the information common to all treatment types, and use different elements in a common substitution group to represent the different forms of information that are recorded for the different types of treatment:

```
<element name="treatment-dto" type="tns:treatment-dto-type"
         abstract="true"/>

<complexType name="treatment-dto-type" abstract="true">
    <sequence>
        <element name="id" type="long"/>
        <element name="diagnosis" type="string"/>
        <element name="patient" type="long"/>
        <element name="provider" type="long"/>
    </sequence>
</complexType>

<complexType name="drug-treatment-type">
    <complexContent>
        <extension base="tns:treatment-dto-type">
            <sequence>
                <element name="drug" type="string"/>
                <!-- TODO fill in other elements -->
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

You are provided with two Eclipse Maven projects[1]:

1. ClinicSchema provides a partial solution using the choice element (see Schema.xsd in this project, in `src/main/resources`).
2. ClinicSchema2 provides a partial solution using element substitution (see Schema2.xsd in this project).

Both projects use JAXB to automatically generate a Java representation from the schema. Select `Run As… | Maven Install` to compile each project. When you edit the schema, these classes should be automatically regenerated by the JAXB Maven plugin[2].

In addition these projects contain factory classes, that you need to complete, for creating patient, provider and treatment objects. They also contain a tester class, `SchemaTest`, that

[1] To import into Eclipse, select `Import | Maven | Import as Maven Project`.
[2] You may find it useful to clean the projects (`Project | Clean` in Eclipse).

uses these factories to create sample data as JAXB objects, and then uses JAXB to write these JAXB objects to standard output (as XML).

For each project, you should do the following:

1. Complete the schema with the definitions of the different forms of treatment types.
2. Complete the factory classes with operations for creating treatment records. This should use the operations defined by the `ObjectFactory` class generated by JAXB.
3. In `SchemaTest`, provide code to instantiate, and fill in the fields of, objects for patients, providers and the various forms of treatment.
4. Use JUnit to run the unit testing provided by `SchemaTest` (Right-click on the project and select `Run As ... | JUnit Test`).
5. Record videos of unit testing for each solution. The video should include your name, the testing code, and the console output.

In addition, complete the rubric provided with these assignment, for your own self-evaluation. Note that a penalty may be incurred for misrepresentation in the rubric.

Your solution should be uploaded via the Canvas classroom, as a zip file. This zip file should have the same name as your Canvas userid. It should unzip to a folder with this same name, which should contain the files and subfolders with your submission. In addition to the completed projects, your submission should include videos of your unit tests, as well as a completed rubric.