Exercise 3

(15 points)

The goal in this year's exercise is to develop an XML-based format to represent data for the quiz game "Jeopardy!". In Exercise 2 you will create an XML Schema and XML document and in Exercise 3 you will be querying and transforming this document with XML-related technologies.

In Jeopardy two players compete against each other. The game displays questions of different value in different categories. Each player chooses alternately one of these questions. Then, the question will be displayed and the players can choose one or more answers. At least one answer is correct. If the answer is correct the players account is increased by the value of the question. The player with the highest account balance wins.

Remark: The following data format can be seen as a savegame for Jeopardy.

This exercise consists of two parts. First, we will use DOM and SAX to access and modify XML documents. Second, we will evaluate an xQuery expression over an XML document.

Template

In the following ZIP-file you will find a general template for this exercise. This template also contains an ant-script (build.xml) which can be used to test your solution. A similar ant-script (with more test cases) will be used during the assignment discussions.

- Template SSD exercise 3 - release 2

Please, remember:

- It is **mandatory** that you use this template!
- All files and paths in this description refer to this template.
- Precise instructions for each ant target can be found in the sections below.
- We have also included an XML document and a schema file for your convenience. These files are those that were created in exercise 2. They can be found in `resources/jeopardy.xml` and `resources/jeopardy.xsd`. These files will be used and are configured as input for the ant targets.
- For validation use `resources/jeopardy.xsd`. **Remark:**We have adapted the schema. We now allow games with either 1 or 2 players.

DOM and SAX

Imagine that your application is divided into a client that displays the Jeopardy! game and a server that processes and manages Jeopardy! games. During the course of the game the client displays a question to a player and the player chooses appropriate answers. The player then commits his answers and the client sends the players answers to the server. The client uses the following "move" XML document for this communication:

```
<move session="abc">
```

```
        <player>Lisa</player>
        <question>5</question>
        <answer>What is Semistructured Data?</answer>
        <answer>What is Web Engineering?</answer>
    </move>
```

This XML document states that in session "abc" the player "Lisa" selected for the question with the id "5" two answers given in the `<answer>` elements.

Your goal is to manipulate the `jeopardy.xml` document with DOM in such a way, that the above "move" is stored in the document. The "move" document is parsed using SAX.

You don't have to handle any exceptions, but you have to ensure that your resulting document validates against the given `jeopardy.xsd`.

Description of Classes
The template provides you two classes. The class `SSD` contains the main program logic. The class `JeopardyMoveHandler` is used as a SAX handler for parsing the "move" document and manipulating the Jeopardy document. A detailed description follows:

- *Class:* **SSD**
  - *Variables:*
    - `static DocumentBuilderFactory documentBuilderFactory`:
      stores an instance of a document builder factory.
    - `static DocumentBuilder documentBuilder`:
      stores an instance of a document builder.
  - *Methods:*
    - `static void main(String [] args) throws Exception`:
      Entry point for the program. Parses the program arguments and calls `initialize` and `transform`.
    - `static void initialize() throws Exception`:
      Initializes the `documentBuilderFactory` and the `documentBuilder` variables.
    - `static void transform(String inputPath, String movePath, String outputPath) throws Exception`:
      Your task is to implement this method. First, you have to create a Document from the filename given in the `inputPath` variable. Second, you have to set up the SAX parser, in order to parse the XML document given in the `movePath` variable. For this you have to create an instance of the `JeopardyMoveHandler` class, which needs the previously created Document as an argument to its constructor. Third, you parse the move document. The `JeopardyMoveHandler` will change the Document. Last, retrieve this document with the method `getDocument()` and store this document in the file given by the `outputPath` variable.
    - `static void exit(String message)`:
      You can use this method to output an error and exit the program.
- *Class:* **JeopardyMoveHandler**
  - *Variables:*
    - `static XPath xPath`:
      use this XPath instance to compile xPath queries that can be evaluated over the document or specific context nodes.
    - `Document jeopardyDoc`:
      stores the DOM document of the Jeopardy XML file.

- `String eleText`:
  stores the text content of XML elements.
- **You can specifiy additional variables in this class.**

◦ *Methods:*

- `JeopardyMoveHandler(Document doc)`:
  The constructor which has a Document variable as argument.
- `void characters(char[] text, int start, int length)`:
  SAX calls this method, when it sees character data. This character data is stored in the `eleText` variable.
- `Document getDocument()`:
  Returns the XML document stored in this class.
- **Specify additional methods to parse the "move" document (e.g.: `startElement`, etc.) and modify the `jeopardyDoc` document.**

Call of program and result
The code implemented in `src/ssd/SSD.java` can be run using three program arguments. The first is a Jeopardy document as input (e.g. `jeopardy.xml` from exercise 2 or the file located in the `resources` folder). The second is a "move" document (e.g. `resources/jeopardy-move.xml`). The last is a Jeopardy document as output (e.g. `output/jeopardy.xml`). We have configured two ant-targets:

- `ant run-dry`:
  applies the "move" document `resources/jeopardy-move.xml` to the Jeopardy document `resources/jeopardy.xml` and stores the output to `output/jeopardy-out.xml`.
- `ant run-persistent`:
  applies the "move" document `resources/jeopardy-move.xml` to the Jeopardy document `resources/jeopardy.xml` and stores the output to `resources/jeopardy.xml`. Therefore, the input file will be overwritten and the "move" is made persistent.

We provide a file `resources/jeopardy-sample-out.xml` that shows the output of the program by applying `resources/jeopardy-move.xml` to `resources/jeopardy.xml`.

Hints
You can evaluate XPath expressions over an XML document (or also relative to a node) using the following statements:

```
XPathExpression xpathExpr = xPath.compile("//player");
NodeList playerList = (NodeList)xpathExpr.evaluate(jeopardyDoc,
                                                  XPathConstants.NODESET);
```

Summary

- **Resulting Files**: `SSD.java` and `JeopardyMoveHandler.java`
- **Total points**: 10

XQuery

In this exercise we will create an XQuery that evaluated over a Jeopardy! document, will give you a list of games together with the players and their correct answers.
The output should look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<stats>
   <session id="abc">
       <correct player="Bart">1</correct>
   </session>
   <session id="abcd">
       <correct player="Bart">2</correct>
       <correct player="Lisa">4</correct>
   </session>
</stats>
```

The meaning of the document is as follows: In session "abc" the player "Bart" has "1" answer correct. In session "abcd" the player "Bart" has "2" answers correct and the player "Lisa" has "4" answers correct. To obtain such a document you can follow this guideline (optional):

Part (a)
First, you create a query `src/xquery-a.xq` that outputs for the session "abcd" and the player "Bart" the correct `givenanswer` elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<correct>
   <givenanswer player="Bart">Who is Maggie?</givenanswer>
   <givenanswer player="Bart">Who is Sheldon Lee Cooper?</givenanswer>
</correct>
```

For this you join the `<givenanswer>` elements of this session and player, with the correct `<answer>` elements of the corresponding `<question>` elements having the same `text()` content as the `<givenanswer>` element. Then, you output the `<givenanswer>` elements.

You can write this query in `src/xquery-a.xq`. The query can be evaluated over `resources/jeopardy.xml` using the following command:

- `ant run-xquery-a`

The output is written to `output/xquery-out-a.xml`.

Part (b)
Second, you create a query `src/xquery-b.xq` that uses the above query for each `<game>` element and counts the outputted `<givenanswer>` elements. Then, you create the XML document given above.

You can write this query in `src/xquery-b.xq`. The query can be evaluated over `resources/jeopardy.xml` using the following command:

- `ant run-xquery-b`

The output is written to `output/xquery-out-b.xml`.

Summary
You don't have to follow the guideline above. If your `xquery-b.xq` outputs the correct answer, you will get all points immediately. If not, you can still receive 2 points for having a correct output for Part (a) (`xquery-a.xq`).

- **Resulting Files**: `xquery-b.xq` (optional `xquery-a.xq`)
- **Points**: 5 (optional if `xquery-b.xq` is not correct, then 2 points for a correct `xquery-a.xq`)

Upload

In order to create a zip file `ssd-exercise3-ss15.zip` you have to use the following command:

- `ant zip`

This zip file has to be uploaded in our CourseManager until 04.06.2014 at 23:59. The newest version will be checked during the assignment discussion.

Assignment discussion

You can receive at most 15 points for Exercise 3. During the assignment discussion we will not only check your solution for correctness, but will also ask some questions regarding the used technologies.

To obtain the full number of points your solution has to be solved correctly and you have to be able to explain it. Copied solutions are awarded 0 points!

In your own interest come **ontime** to your assignment discussion or else we don't guarantee that your solution is completely checked in the remaining time of the reserved slot.