8 – XML Processing in Java & .NET | Exercise

1. Give the list of SAX events produced from the following XML document.

```
<?xml version="1.0"?>
<doc>
   <para>Hello world!</para>
</doc>
```

start document
start element (doc)
start element (para)
characters (Hello world!)
end element (para)
end element (doc)
end document

Exercise

1. Write a SAX application that produces the following console output

```
Pierce Brosnan played in 4 James Bond movie(s):
  - Goldeneye
  - Tomorrow never dies
  - The world is not enough
  - Die another day
Daniel Craig played in 3 James Bond movie(s):
  - Casino Royale
  - Quantum of Solace
  - Skyfall
George Lazenby played in 1 James Bond movie(s):
  - On her majesty's secret service
Roger Moore played in 7 James Bond movie(s):
...
```

```
public class SAX_Example {
	public static void main(String[] args) {
		try {
			XMLReader reader = XMLReaderFactory.createXMLReader();
			reader.setContentHandler(new BondHandler2());

			File file = new File("../bond_movies.xml");
			reader.parse(new InputSource(new FileReader(file)));

		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

```
public class BondHandler2 extends DefaultHandler {

	private String title, bond, current;
	private HashMap<String, ArrayList<String>> data = new HashMap();

	@Override
	public void startElement(String uri, String local, String name, Attributes attr) {
		current = name;
	}

	@Override
	public void endElement(String uri, String localName, String name) throws SAXException {
		if(name.equals("movie")) {
			ArrayList<String> list = data.get(bond);
			if(list == null) {
				list = new ArrayList<String>();
				data.put(bond, list);
			}
			list.add(title);
		}
	}
```

```
        @Override
        public void characters(char[] ch, int start, int length) {
                if(current.equals("bond")) {
                        String str = new String(ch, start, length);
                        if (!str.contains("\n")) {
                                bond = str;
                        }
                } else if(current.equals("title")) {
                        String str = new String(ch, start, length);
                        if (!str.contains("\n")) {
                                title = str;
                        }
                }
        }

        @Override
        public void endDocument() throws SAXException {

                for(Map.Entry<String, ArrayList<String>> entry : data.entrySet()) {
                        System.out.println(entry.getKey()+" played in "+entry.getValue().size()+"
James Bond movie(s):");
                        for(String m : entry.getValue()) {
                                System.out.println("\t- "+m);
                        }
                }
        }
}
```

Exercise

1. Write a JDOM application that produces the following console output

```
Pierce Brosnan played in 4 James Bond movie(s):
  - Goldeneye
  - Tomorrow never dies
  - The world is not enough
  - Die another day
Daniel Craig played in 3 James Bond movie(s):
  - Casino Royale
  - Quantum of Solace
  - Skyfall
George Lazenby played in 1 James Bond movie(s):
  - On her majesty's secret service
Roger Moore played in 7 James Bond movie(s):
...
```

```
SAXBuilder builder = new SAXBuilder();
Document doc = builder.build(new FileInputStream("bond_movies.xml"));
Element root = doc.getRootElement();

List<String> usedBonds = new ArrayList<String>();
for (Element bondElement : root.getChildren()) {

        String bond = bondElement.getChildText("bond");
        if (!usedBonds.contains(bond)) {

                XPathExpression<Element> xpath = XPathFactory.instance()
                                .compile("//bond[text() = \"" + bond + "\"]/../title",
                                                Filters.element());

                List<String> movies = new ArrayList<String>();
                for (Element movieElement : xpath.evaluate(doc)) {
                        movies.add(movieElement.getText());
                }

                System.out.println(bond + "played in " + movies.size()
                                + " James Bond movie(s):");
                for (String title : movies) {
                        System.out.println("- " + title);
                }
                System.out.println();

                usedBonds.add(bond);
        }
}
```

Exercise

1. Write a JDOM application that produces the following XML file (Observe that it is all sorted!)

```
<?xml version="1.0" encoding="UTF-8"?>
<bond_actors>
  <actor name="Daniel Craig">
    <title>Casino Royale</title>
    <title>Quantum of Solace</title>
    <title>Skyfall</title>
  </actor>
  <actor name="George Lazenby">
    <title>On her majesty's secret service</title>
  </actor>
  <actor name="Pierce Brosnan">
    <title>Die another day</title>
    <title>Goldeneye</title>
    <title>The world is not enough</title>
    <title>Tomorrow never dies</title>
  </actor>
  <actor name="Roger Moore">
    <title>A view to a kill</title>
    <title>For your eyes only</title>
    <title>Live and let die</title>
    <title>Moonraker</title>
    <title>Octopussy</title>
    <title>The man with the golden gun</title>
    <title>The spy who loved me</title>
  </actor>
<actor name="Sean Connery”>...</actor>
<actor name="Timothy Dalton">...</actor>
</bond_actors>
```

```
SAXBuilder builder = new SAXBuilder();
Document input = builder.build(new FileInputStream("../bond_movies.xml"));
XPathExpression<Element> xpath = XPathFactory.instance().compile("//movie", Filters.element());

HashMap<String, Element> map = new HashMap<String, Element>();
Comparator cmp = (new JDOM_Example4()).new ElementComparator();

for (Element e : xpath.evaluate(input)) {

        String actor = e.getChildText("bond");
        if(!map.containsKey(actor)) {
                Element a = new Element("actor");
                a.setAttribute(new Attribute("name", actor));
                map.put(actor, a);
        }

        Element title = new Element("title");
        title.setText(e.getChildText("title"));
        map.get(actor).addContent(title);
}

Element root = new Element("bond_actors");

for(Element e : map.values()) {
        e.sortChildren(cmp);
        root.addContent(e);
}

root.sortChildren((new JDOM_Example4()).new AttributeComparator());

Document output = new Document(root);

XMLOutputter out = new XMLOutputter();
FileOutputStream file = new FileOutputStream("../bond_actors.xml");
out.output(output, file);
```

Control Questions

1. XML documents can be processed sequentially or by creating an in-memory representation. Name advantages, disadvantages and a framework implementation of each variant.

   SAX:

   - *Sequentiell*
   - *Memory effizient*
   - *Kann nur vorwärts durchlaufen werden*

   JDOM:

   - *In-Memory DOM (~4x RAM der Dateigrösse)*
   - *Wahlfreier-Zugriff*
   - *XPath-Builder*
   - *Anbindung an Java-Collection-Klassen*

2. What is the different between pull and push processing ?

   SAX: push (Event based)

   JDOM: pull (while…)

3. Explain the steps for processing an XML document with SAX.

   1. Create a parser (XMLReader)

   2. Give DefaultHandler extension to parser

   3. Start reader on XML file

   - *DefaultHandler erweitern*
   - *benötigte Methoden überschreiben (Listener erzeugen)*

4. What is the difference between DOM and JDOM ?

   DOM verwendet eigene Collections (da es nicht explizit auf Java aufbaut)

   JDOM verwendet die Java-Collections

5. Explain the steps for processing an XML document with JDOM.

   1. SAXBuilder returns JDOM data structure

   2. Access the root node

   3. Browse the tree

```
SAXBuilder builder = new SAXBuilder();
Document doc = builder.build(
   new FileInputStream("../bond_movies.xml"));
Element root = doc.getRootElement();
for(Element e : root.getChildren()) {
   System.out.println(e.getChildText("title"));
}
```

6. Explain the steps for invoking an XSL transformation from Java.

   1. Create transformer with

   i. a streamsource for .xsl file

   ii. a streamsource for input XML file

   iii. a streamresult for output XML file

   2. Start transformation

```
TransformerFactory factory = TransformerFactory.newInstance();
Transformer transformer = factory.newTransformer(
   new StreamSource("../../XML 5 - XSLT/bond_movie_xhtml_table.xsl"));
transformer.transform(
   new StreamSource("../bond_movies.xml"),
   new StreamResult("../result.xhtml"));
```