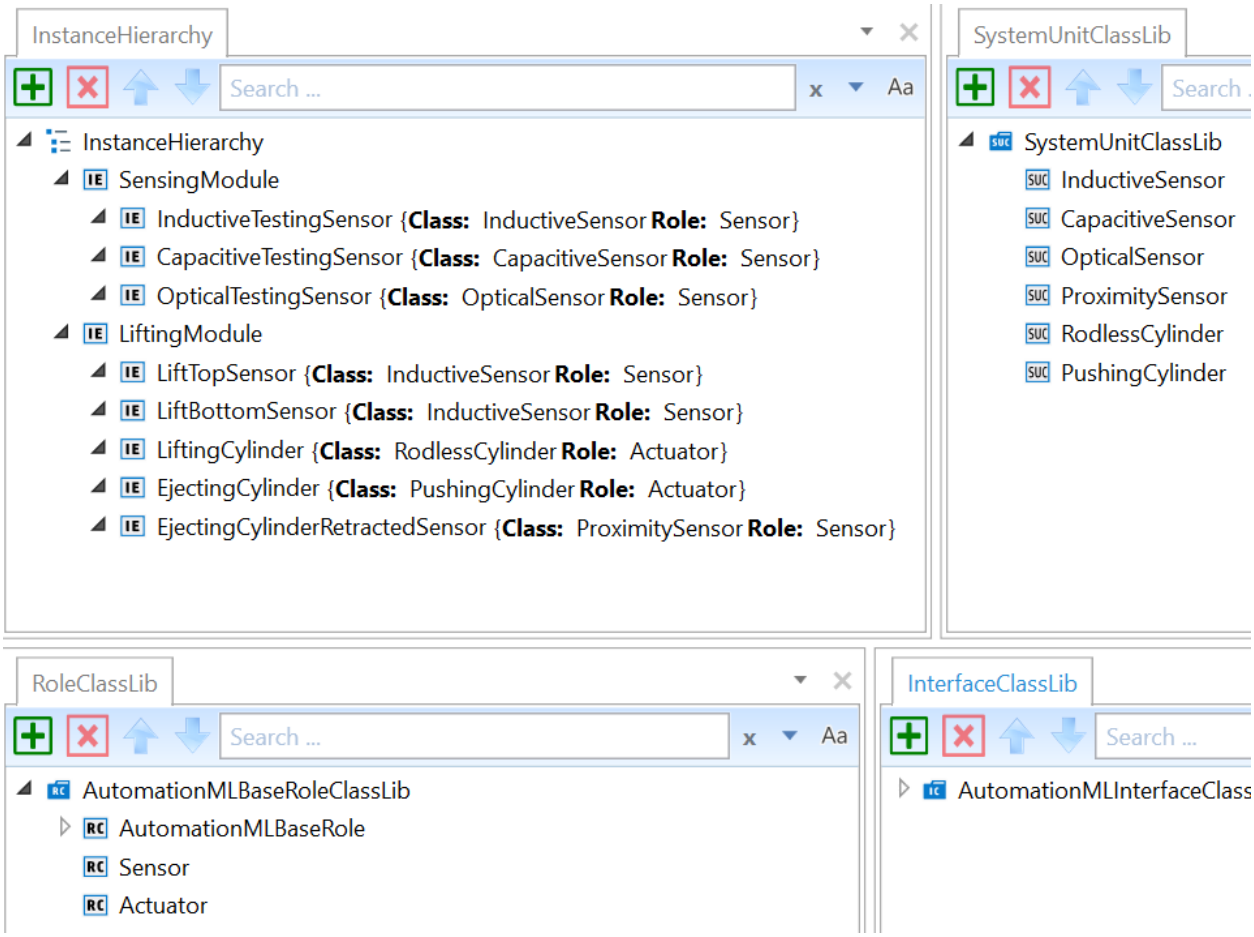
XML ohjelmointi

Oppimistavoitteet

- Perusvalmiuksia XML ohjelmistokehitykseen: opitaan kehittämään Java-ohjelmisto, joka etsii XML tiedostosta kiinnostavia tietoja ja tulostaa ne halutussa muodossa
- Java tekniikat ovat yleisesti päteviä kaikkiin XML dokumentteihin, mutta tehtävässä keskitytään erityisesti AutomationML dokumenttiin.

Tehtävänanto

Kurssihenkilökunta on mallintanut AutomationML Editorilla tuotantolinjamme testausaseman (testing station), joka on kuvattu FestoModularProductionSystem.pdf luvussa 3. Malli pitää sisällään lukujen 3.2 ja 3.3. taulukoissa luetellut instrumentit. Kuva AutomationML Editor mallista on alla:



Tämä on tallennettu testing_station.aml nimiseen XML tiedostoon. Tässä tehtävässä tehdään Java ohjelma, joka käsittelee tätä tiedostoa. Kyseinen tiedosto pitää laittaa Java projektisi kansioon, jotta se

löytyy ilman että kikkaillaan polkunimillä (mikä voi erityisesti Aalto domainissa aiheuttaa kaikenlaisia ongelmia. Jos tulee virheilmoitus siitä että tiedostoa ei löydy niin mitä todennäköisimmin asia korjaantuu laittamalla tiedosto oikeaan kansioon.

Kurssin puolesta on annettu seuraava koodi, joka on saatavilla myös erillisenä java-tiedostona AutomationMLReader_skeleton.java:

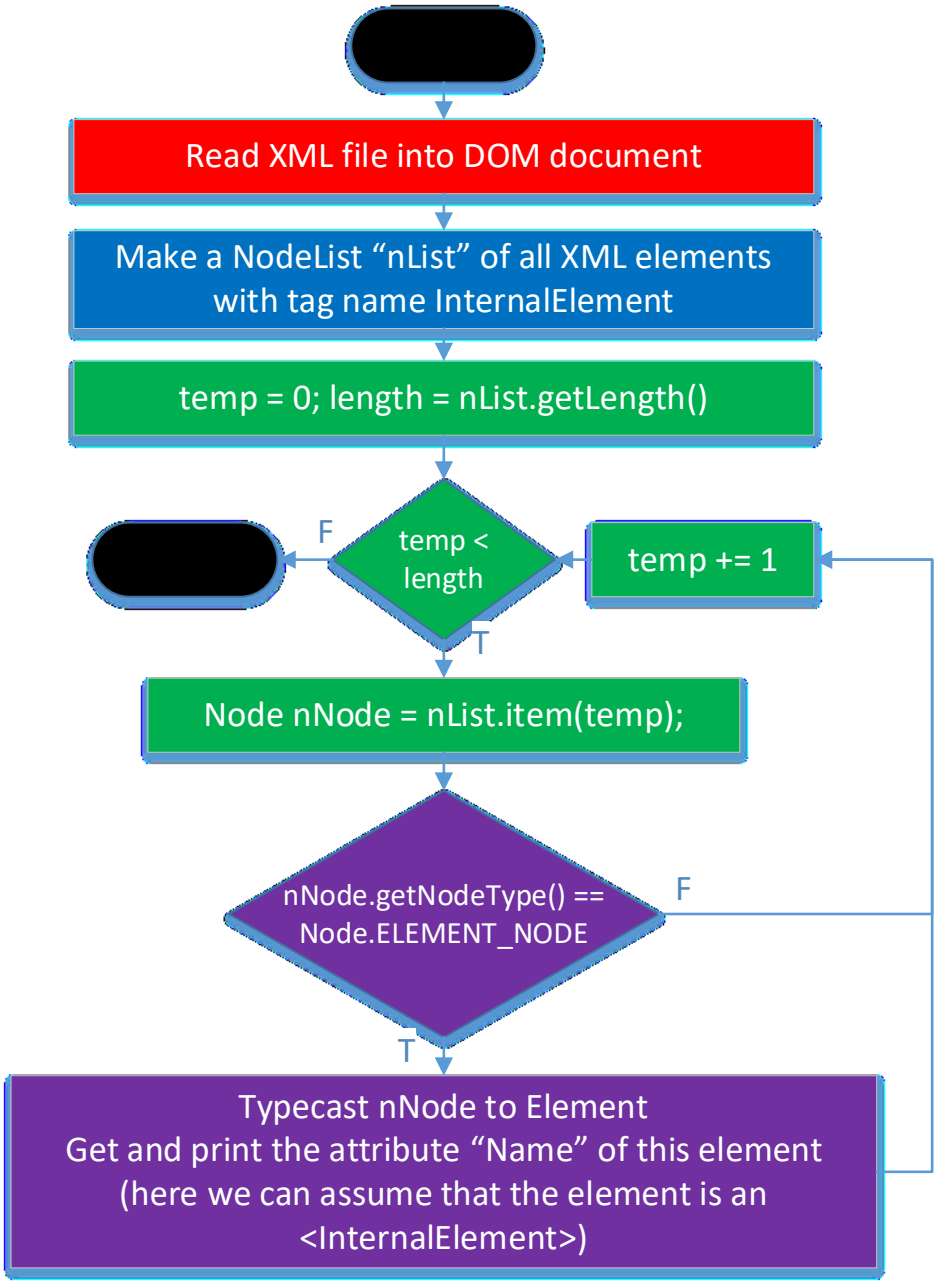
```
import java.io.File;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;

public class AutomationMLReader_skeleton {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		try {
		  File inputFile = new File("testing_station.aml");
		  //File inputFile = new File("proteusexample.xml");
		  DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
		  DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
		  Document doc = dBuilder.parse(inputFile);
		  doc.getDocumentElement().normalize();
		  System.out.println("Root element :" +
doc.getDocumentElement().getNodeName());
		  NodeList nList = doc.getElementsByTagName("InternalElement");
		  System.out.println(nList.getLength());

		  for (int temp = 0; temp < nList.getLength(); temp++) {
		       Node nNode = nList.item(temp);
		       if (nNode.getNodeType() == Node.ELEMENT_NODE) {
		              Element eElement = (Element) nNode;
		              String componentName = eElement.getAttribute("Name");
		              System.out.println(componentName);
		       }
		  }
		} catch (Exception e) {
		       e.printStackTrace();
		}
	}

}
```

Javassa on DOM (Document Object Model) kirjasto, jossa on metodit XML tiedoston lukemiseksi tietorakenteeseen, jota voidaan käsitellä kätevästi sitä varten tehdyillä metodeilla. Tämä on melkein kokonaan niin sanottua ’boilerplate’ koodia, eli samat asiat pitää tehdä aina kun prosessoidaan XML dokumenttia. Yllä olevaa koodia muokkaamalla voi päästä pitkälle Java-XML hommissa eikä syvällisempi

DOMiin perehtyminen kuulu tämän kurssin vaatimuksiin. Asiasta kiinnostuneet voi katsoa DOM tutoriaalin:

https://www.w3schools.com/jsref/dom_obj_document.asp

Alla on vuokaavio joka kuvaa mitä tämä koodi tekee (värikoodaus havainnollistaa vastaavuudet koodiin). Vuokaaviossa timantti on ehto, jolla kuvataan if-lause tai silmukka. Erona on että silmukan tapauksessa timanttiin on takaisinkytkentä.



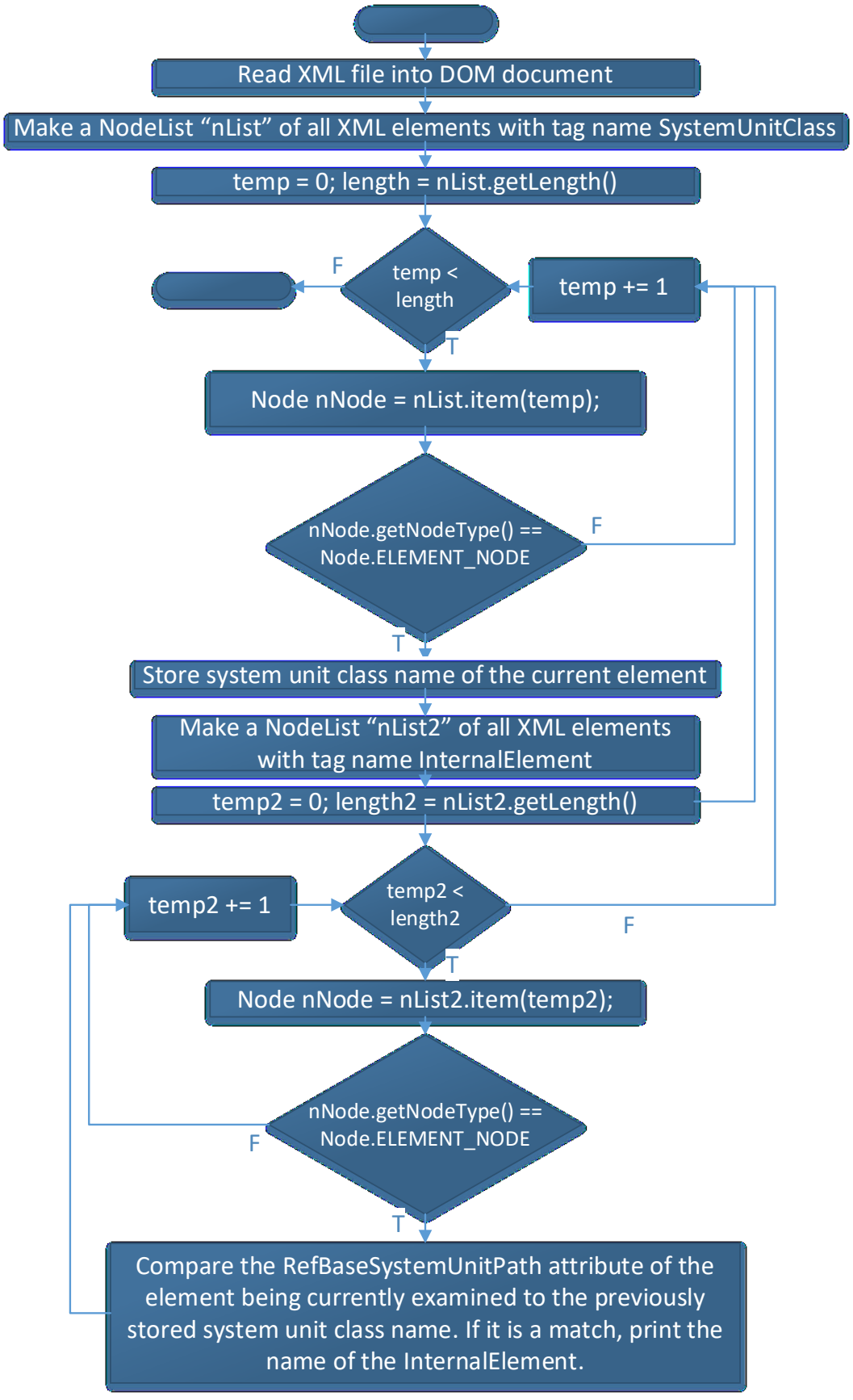
Boilerplate koodin lisäksi tämä koodi etsii kaikki <InternalElement> ja printtaa niiden nimet. Konsoliin pitäisi tulla tällainen tulostus:

```
Root element :CAEXFile
10
SensingModule
InductiveTestingSensor
CapacitiveTestingSensor
OpticalTestingSensor
LiftingModule
LiftTopSensor
LiftBottomSensor
LiftingCylinder
EjectingCylinder
EjectingCylinderRetractedSensor
```

Tehtävänä on muuttaa boilerplate koodia, siten että se katsoo mitä kaikkia <SystemUnitClass> luokkia .aml tiedostossa on ja sitten listaa kunkin alle kaikki <InternalElement> elementit joilla on kyseinen luokka RefBaseSystemUnitPath attribuuttina. Voi olettaa että <RefBaseSystemUnitPath> attribuutin arvo on string, joka saadaan yhdistämällä ”SystemUnitClassLib/” ja < SystemUnitClass> luokan Name-attribuutti. Pitäisi tulla tämänkaltainen output:

```
Internal elements with system unit class SystemUnitClassLib/InductiveSensor
InductiveTestingSensor
LiftTopSensor
LiftBottomSensor

Internal elements with system unit class SystemUnitClassLib/CapacitiveSensor
CapacitiveTestingSensor

Internal elements with system unit class SystemUnitClassLib/OpticalSensor
OpticalTestingSensor

Internal elements with system unit class SystemUnitClassLib/ProximitySensor
EjectingCylinderRetractedSensor

Internal elements with system unit class SystemUnitClassLib/RodlessCylinder
LiftingCylinder

Internal elements with system unit class SystemUnitClassLib/PushingCylinder
EjectingCylinder
```

Tehtävän voi toteuttaa haluamallaan tavalla jos ei tarvitse teknistä tukea kurssihenkilökunnalta. Jos kaipaa teknistä tukea niin tulee toteuttaa alla olevan vuokaavion mukaan:


Read XML file into DOM document
Make a NodeList “nList” of all XML elements with tag name SystemUnitClass
temp = 0; length = nList.getLength()
temp < length
F
T
temp += 1
Node nNode = nList.item(temp);
nNode.getNodeType() == Node.ELEMENT_NODE
F
T
Store system unit class name of the current element
Make a NodeList “nList2” of all XML elements with tag name InternalElement
temp2 = 0; length2 = nList2.getLength()
temp2 += 1
temp2 < length2
F
T
Node nNode = nList2.item(temp2);
nNode.getNodeType() == Node.ELEMENT_NODE
F
T
Compare the RefBaseSystemUnitPath attribute of the element being currently examined to the previously stored system unit class name. If it is a match, print the name of the InternalElement.

Reflektointi

Entä jos pitäisikin etsiä kaikki <InternalElement> elementit joilla on <RoleRequirement> sensor tai actuator. Missä määrin tekemäsi ratkaisu olisi suoraviivaisesti muokattavissa ja missä kohdin pitäisi keksiä jotain uutta?