Lab Sheet – Serialization with XML

There are various ways we can use XML in our AddressBook but first we will look at some simple examples. First we will create a file called file.xml that contains the following:

```
<tutorial>
        <number>3</number>
        <ta>Michael</ta>
</tutorial>
```

We will use this file as input to our XML parser.
First, let us look at a Simple API for XML (SAX) parser. Create a new class and add the following method to the class:

```
public static void readSAX(File f) throws Exception{
        SAXParserFactory spf = SAXParserFactory.newInstance();
        SAXParser s = spf.newSAXParser();

        DefaultHandler dh = new DefaultHandler(){
                public void startElement(String u, String ln, String qName, Attributes a){
                        System.out.println("START: " + qName);
                }

                public void endElement(String uri, String localName, String qName){
                        System.out.println("END: " + qName);
                }


                public void characters(char[] ch,int start, int length){
                        System.out.println("CHARS: "+new String(ch, start, length));
                }
        };
        s.parse(f, dh);
}
```

In your main method, call the readSAX(..) method and give it the “file.xml” file as a parameter.

Question 1: What is the console output?
Question 2: When are the methods in DefaultHandler called?

Next, we will look at another type of XML parser: Document Object Model (DOM). Add the following method to your class:

```
public static void readDOM(File f) throws Exception{
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder d = factory.newDocumentBuilder();
        Document doc = d.parse(f);

        System.out.println ("Root: " + doc.getDocumentElement().getNodeName());

        NodeList lst = doc.getDocumentElement().getChildNodes();
        for(int ii=0; ii<lst.getLength();ii++){
                Node n = lst.item(ii);
                System.out.println("Child: " + n.getNodeName() + " --> " +
                        n.getTextContent());
        }
}
```

Again, run this method and give "file.xml" as the parameter.

Question 3: What is the console output?
Question 4: What is contained in the doc object?
Question 5: How do the SAX and DOM parsers differ?

Now let's use XML to serialize our Address Book. The first step is to actually come up with an XML format for the Address Book. The simplest idea is to have an element called AddressBook, have BuddyInfo and other instance variables of AddressBook nested as sub elements of AddressBook, and finally the instance variables of BuddyInfo nested as sub elements of BuddyInfo… Since we do not want to rewrite the import/export methods again as we are happy with the Java serialization solution from the last lab, let's define the following methods:

1**. public String toXML():** similar to the toString() methods already defined for AddressBook and BuddyInfo except that the output will be in the XML format that you have defined.
2**. ExportToXmlFile() and importFromXmlFileSAX()** (**or importFromXmlFileDOM()** if you prefer) provided in AddressBook.

Deliverables
1. The source code of the following classes: AddressBook and BuddyInfo.
2. The answers to the questions (in a text file).

Show to the TA.