Java Programming for Beginners

Lab Exercise 11

Answer Key

1) When we turn the raw text of an XML document into searchable information, we say that we have the document's "Tree". What is the significance of this term?

In the programming/computer-science world, a Tree is a data structure in which each element has exactly one parent (except for the "top" or "root" element, which has no parent). This means that we can access any element in the tree by starting at the top and stepping through the correct children (we never have to look back "up"). XML documents are structured so that they are always trees.

2) When parsing an XML document in Java we must know XML's terminology. What are the differences between the following three XML "pieces":

Element: In an XML document, elements are defined by their start and end tags. If you see a pair of tags (or a start/end tag like: <element \>) those tags, and everything inside of them is an element. This means that one element may contain other elements within it.

Node: A Node is almost anything in an XML document. Elements and Attributes are both nodes, themselves. (Though, they are Nodes of different types.)

Attribute: XML attributes are "additional information" placed within the start tag of an element. For example, if the "Element" element had a "name" attribute, it might look like this:

```
<Element name="value">BlaBlaBla</Element>
```

Attributes are paired with string values that we access through the attribute's name.

3) When writing Java code to parse or write XML to/from a file, we will often be prompted to handle a number of exceptions (like IOException). Why does Java require us to always handle these exceptions, but not every exception that could possibly be thrown?

Java divides Exceptions into categories. Some exceptions are "Checked" and others are "Unchecked". Java expects that we will explicitly handle any Checked exception that a method might throw. Because of this, Checked exceptions should be used for the type of events that we expect might occur in our program and that we can recover gracefully from. IOExceptions are a good example of this.