6. LAB F – COMPOSITE

Purpose

This lab will demonstrate the composite design pattern.

We will simulate an XML DOM structure.

Create an XMLElement class representing an XML element.

Such an element may contain textual data, as well as ordered children elements.

Remember that composite has 2 common implementations:

1. The objects tree contains Node elements, each may or may not have children.
2. The object tree contains Node elements which have children and leaf elements without children.

For this exercise we will choose the 1st approach.

Lab Scenario (WHAT to do)	Develop an XMLElement class representing nodes in the tree Each node has a name, textual data and a list of children
Implementation Steps (HOW to do it)	**Code the XMLElement class** Implement member variables: String name, data, List children Implement methods: `XMLElement addElement(XMLElement),` List getChildren (), getName(), setName(), getData(), setData() **Test it** Create a main method which creates a few XMLElement objects adding them to one another

Review of XMLElement

We wish to create a custom hierarchy of objects for XML processing, using the composite design pattern (holding objects in a tree like structure gives the benefit of faster searching, sorting, etc.).

6.1.1. Create a new class XMLElement

1. This class represents a node in an XML tree. Each node can have 0-n children.
2. Create the class XMLElement
3. Implement member variables

```
private String name; // xml element name
private String data; // xml element data
private List<XMLElement> children =
                    new ArrayList<XMLElement>(); // child nodes
```

4. Implement methods:

```
public XMLElement(String name) {
    // store the name and initialize the list
}
public XMLElement addElement(XMLElement element) {
    // add the element to the list and return this
    // returning this for each operation allows chaining
    // operations: XMLElement.add("bb).add("cc)…}
public String getName() { return name; }
public String getData() { return data; }
public List getChildren() { return children; }
public XMLElement setData(String data) {//set data, return this
}
```

Note: The solution for this Exercise is available in the ‘solutions’ directory.