7. LAB H – ITERATOR

Purpose

This exercise will demonstrate the use of the iterator design pattern and will rely on the solution for Lab F (Composite). We will modify `class XMLElement` to return an iterator. Clients will be able to use the iterator to iterate on `XMLElement` children.

Lab Scenario (WHAT to do)	First, re-implement the previous `XMLElement.getChildren()` method – now refactored to be called `iterator()`, so that it returns an iterator. Create a `iterator(String name)` method which returns an iterator, listing only those children with the given name.
Implementation Steps (HOW to do it)	**Create a `class NamedIterator`** - extend Iterator - this iterator constructor will take a parent `XMLElement` and a String name value - iteration should only return nodes matching the name **Modify the `XMLElement` class** Change method iterator to return an iterator (simply return the iterator for the list) Add a `iterator(String name)` method which creates a `NamedIterator` with the reference to the parent element and a named element, and returns the iterator.

Review of Iterator

This lab will reuse the solution from the previous lab and will add Iterator functionality to the `XMLElement` class. `XMLElement` should support 2 types of iterations:

1. Normal – iterates through all children
2. By name, iterates children matching the given name value only

7.1.1. Modify class XMLElement

1. Edit the class `XMLElement`
2. Add 2 methods :

```
public Iterator<XMLElement> iterator() {
      return children.iterator();
}

public Iterator<XMLElement> iterator(String name) {
      return new NamedIterator(this, name);
}
```

3. Create a `class NamedIterator`. This iterator will take a parent `XMLElement` element and name value in its constructor and only return nodes matching the given name

   a. Extend iterator

   b. Implement the following data members

```
private Iterator<XMLElement> iterator;
private String elementName;
private XMLElement current;
```

   c. Implement the constructor

```
public NamedIterator(XMLElement parent,
                     String elementName)
```

   When the iterator is created, it should find the first element to return and store it in `current`.

   d. Implement the iterator methods :

```
public boolean hasNext()
```

   will return true if `current` != null

```
public Object next()
```

   The method should return the `current`

   The method should set the next `current` before returning

```
private void findNext()
```

This method uses the default iterator to iterate the next value in the list, until it finds a value matching the name member variable and stores it in `current`. Method should be used from the constructor and the next() method.

```
public void remove()
```

Throw an `UnsupportedOperationException` to simplify the iterator.

Note: The solution for this Exercise is available in the 'solutions' directory