PROGRAM DESCRIPTION

Create a program that validates the nesting of elements in an XML document. The nesting rule simply states that while elements may be nested, they cannot overlap. This is one of several rules that must be satisfied for an XML document to be considered well-formed. For more information on XML documents see **this page** and **this page**.

TASKS & REQUIREMENTS

1. **NOTE**: Naming is critical in the tasks and requirements described below. If the names don't match those described below exactly, your project will not compile and can't be graded.

2. Right-click on the **JavaCoreTemplate** project and select Copy. Right-click in the white area of the Package Explorer and select Paste.

3. The project name must follow this pattern: `{FLname}_XmlValidator_{TERM}`, where {FLname} is replaced by the first letter of your first name plus your last name, and {TERM} is the current semester and year. E.g. if your name is Mario Marciano and its Fall 2019, your project name must be `MMarciano_XmlValidator_F19`. If it doesn't match this pattern, it will not be graded.

4. In your `src` folder, create a package named `xmlvalidator`.

5. `BasicXmlTagStack` class:
   1. Download the **XmlTag.java** class and the **XmlTagStack.java** interface and put them in the `xmlvalidator` package. Read the comments above each method to understand what they are supposed to do.
   2. ~~Create a new class named `BasicXmlTagStack` in the xmlvalidator package. In the class creation wizard, click the Add (interface) button, and search for `XmlTagStack`. Check the "Inherited abstract methods" box. Click Finish. Eclipse will create the class and automatically add method stubs that meet the `XmlTagStack` interface.~~
   3. You do not modify the `XmlTagStack` interface. You add your code in the `BasicXmlTagStack` class.
   4. **You must write your own stack code** - i.e. you can't use `java.util.Stack` or any other existing implementation. You are welcome to use the guidebook's code as a guide if you like.

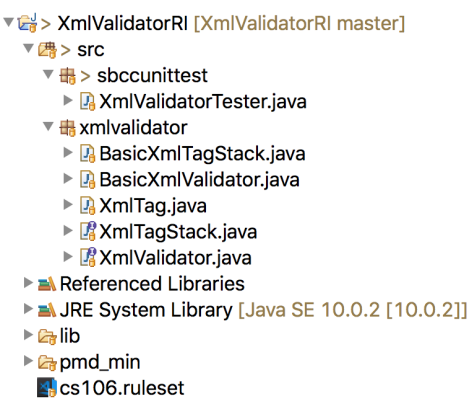
6. `BasicXmlValidator` class:
   1. ~~Download the **XmlValidator.java** interface and put it in the `xmlvalidator` package.~~ Read the comments above each method to understand what they are supposed to do.
   2. ~~Create a new class named `BasicXmlValidator` in the `xmlvalidator` package. In the class creation wizard, click the Add (interface) button, and search for `XmlValidator`. Check the "Inherited abstract methods" box. Click Finish. Eclipse will create the class and automatically add method stubs that meet the `XmlValidator` interface.~~
   3. ~~You do not modify the `XmlValidator` interface. You add your code in the `BasicXmlValidator` class.~~

7. Validation notes:

1. Ignore any tag that doesn't start with '/' or a letter character. This includes xml version/encoding elements like `<?xml version="1.0" encoding="UTF-8"?>`, and comments like `<!-- diblah -->`.
2. Ignore self-closing tags like `<sometag />`.
3. Ignore comments like `<!-- This is a comment -->`.
4. Line numbers start at 1. `'\n'` marks the end of a line.
5. You can assume that the XML document won't have any CDATA sections.
6. Note that XML start tags can span multiple lines. E.g. :

```
<sometag
   attr1="blah"
   attr2="diblah">
```

8. Add the unit test class. This class will be used to test the code that you write.
   1. ~~Create a new package in the **`src`** folder called **`sbccunittest`**~~. To be clear: **`sbccunittest`** should be a child of the **`src`** folder, not of the **`xmlvalidator`** package.
   2. Download **XmlValidatorTester.java** into **`sbccunittest`**.
9. Here is an example of what your project directory should look like once you've downloaded and created all of the files:



10. Here is a **sample XML file with invalid nesting** (right-click "Save Link As..."). Calling `validate()` with the contents of the file should return the following `List<String>`: `[ "Tag mismatch", "name", "14", "buildCommand", "17"]`

11. Unit Testing
    1. Debug all java compilation Errors (see the Problems view). The unit tests can't be run until these errors are fixed. NOTE: as shown above, some of the XML test files will have errors. Don't try to fix these files, the errors are meant to be detected by your `BasicXmlValidator` class.
    2. In the Package Explorer, right-click on the `sbccunittest` package | Run As... | JUnit Test.
    3. Initially, all of the tests will probably fail. However, as you add functionality to the `BasicXmlTagStack` class, tests will begin to pass.
    4. Work on `BasicXmlTagStack` first. Continue adding functionality to `BasicXmlTagStack` until `testPush()`, `testPop()`, and `testExercise()` all pass.

12. There is no user interface requirement (i.e. no `Main` class) for this assignment.

SCORING

```
   2 pts  -  testPush
   3 pts  -  testPop
   5 pts  -  testExercise

  10 pts  -  testTagMismatch
   5 pts  -  testOrphanClosingTag
  10 pts  -  testUnclosedTagAtEnd
  10 pts  -  testValidFile
   5 pts  -  testBigValidFile
   5 pts  -  testPmd

   3 pts extra credit - testAttributeNotQuoted
```


BasicXm