CSE 2221 - Project 7

Task
Gain familiarity with recursion by using it to evaluate arithmetic expressions. Gain familiarity with **XMLTree** objects and methods. Gain familiarity with **NaturalNumber** objects and methods.

Original Project Instructions
Project 7 Instructions from CSE2221 Project Site

Program Requirements

- Implement a **recursive** algorithm that will compute an expression correctly from an XMLTree object
- You will implement the above algorithm twice where one works for ints and one works for NaturalNumber objects

Creating XMLTree Expressions & How To Run
You can create expressions represented as XMLTrees using the jar file here.

To run the jar file, in some cases you can double click. If your OS does not allow this (for security reasons), you can open up a terminal (MacOS & Linux) or cmd prompt (Windows), navigate through your directories/folders using ”cd <directory name>” (works for all OS), and run the jar file using ”java -jar xml-expression-generator.jar”. If you would like to list what directories/folders are present in your current directory (to know what directory you are in, type ”pwd” for MacOS & Linux and ”echo %cd%” for Windows), type ”ls” (MacOS & Linux) or ”dir” (Windows).

Example of Created XMLTree

This is an example for 12 * (7 - 3) / 6 + 8

```
<expression>
  <plus>
    <divide>
      <times>
        <number value="12" />
        <minus>
          <number value="7" />
          <number value="3" />
        </minus>
      </times>
      <number value="6" />
    </divide>
    <number value="8" />
  </plus>
</expression>
```

Tips, Rules, & Things to Note

- Order of computation follows standard PEMDAS, except we won’t have an exponentiation
- Top level element is an <expression>, underneath <expression> you can have <plus>, <minus>, <multiply>, <divide>, and <number>
- <number> can only be a NON-NEGATIVE value
- There are no text nodes

- <plus>, <minus>, <multiply>, and <divide> nodes WILL have TWO children, NO LESS
- <number> nodes will have ZERO children, NO MORE
- NO LOOPS. none. zero.
- ONE RETURN per method
- No using the toInt or toString functions
- There should be no TODOs left in your code when you submit
- Should you pull values out as ints or convert directly from string to NaturalNumber?
- You can assume the XMLTree is correctly formatted
- Make sure you do not divide by 0 or do (num1 - num2) where num2 > num1, Use the components.utilities.Reporter's fatalErrorToConsole method to report the error and make the program terminate: FOR THE NATURALNUMBER (PART 2) ONLY
- Again, know when to use transferFrom over copyFrom
- Watch out for aliasing

Recursive Method Structure

```
private static int evaluate(XMLTree exp) {
     declare our variable to return
     if (base case condition) {
          set our return variable to the result of our base case
     } else {
          evaluate subproblem(s)
          combine result(s) of evaluate our subproblem(s), set return variable equal to this
     }
     return the variable variable
}
```

Steps

1. Copy and paste the *ProjectTemplate* project to create a new project folder for this project
2. Name the project *XMLTreeExpressionEvaluator*
3. Open the *src* folder, then open *(default package)*
4. Rename any ONE file to *XMLTreeIntExpressionEvaluator.java*
5. Delete the other files
6. Open *XMLTreeIntExpressionEvaluator.java*
7. Go to this page and copy and paste the source code there into *XMLTreeIntExpressionEvaluator.java*
8. Implement the recursive evaluate function
9. Start by asking yourself, "what is our base case?". In other words, what is the smallest subproblem we can trivially solve?
10. Next, ask yourself, how can we make this problem easier? What subproblems exist in my current problem?
11. Once *XMLTreeIntExpressionEvaluator.java* is done and working successfully, copy *XMLTreeIntExpressionEvaluator.java* to create *XMLTreeNNExpressionEvaluator.java*

12. Replace the evaluate method and method contract from *XMLTreeIntExpressionEvaluator.java* with:

    ```
    /**
     * Evaluate the given expression.
     *
     * @param exp
     *            the {@code XMLTree} representing the expression
     * @return the value of the expression
     * @requires <pre>
     * [exp is a subtree of a well-formed XML arithmetic expression]  and
     *  [the label of the root of exp is not "expression"]
     * </pre>
     * @ensures evaluate = [the value of the expression]
     */
    private static NaturalNumber evaluate(XMLTree exp) {
    }
    ```

13. Reimplement the evaluate algorithm from *XMLTreeIntExpressionEvaluator.java*, except this time do all computation with NaturalNumbers and return NaturalNumbers

14. Once completed and convinced your code looks good: Zip it up witht the naming scheme I recommend and submit to Carmen