Assignment #3 – Binary Tree

Read through the assignment to get a basic idea of the requirements and then start working through the assignment in the recommended order.

Objectives:

The main objectives of this assignment are:

- Implement abstract data structure Binary Tree with linked data structure.
- Implement tree traversal algorithms (preorder, post order, in order and level order)
- Use binary tree to represent arithmetic expression
- Use JUnit test to debug and test each method

Allowed imports:

net.datastructures.*;

If you didn't do well with your queue, list or positional list implementation in prog1 and prog2, please meet with the TA/instructor to get approval and instructions to import:

```
cs2321util.DoublyLinkedList
cs2321util.ArrayList;
cs2321util.CircularArrayQueue.java
```

The Assignment:

1. Download and import the zip file: `Assignment3_Tree.zip` For Eclipse:
   1) Open Eclipse
   2) Choose "File → Import".
   3) Select the "General" category.
   4) Pick "Existing Projects into Workspace"
   5) Click on "Next"
   6) Select the button for "Select Archive File" and browse to/select the zip file.
   7) Click on "Finish"
   8) A new project `Tree` has been added to the workspace:
2. Study the new project.
   1) First, study `tree.java, binarytree.java` in `net.datastructures` package.
   2) Then, study `LinkedBinatryTree.java` in `cs2321` package.
   3) Then, study `ExpressionTree.java in CS2321 package.`
   4) Last, but very important, study the **JUNIT test source code under folder `tests/`** Make sure you understand the test setup and how each test case is written. You should make sure that your code passes all these test cases. You are encouraged to

write more test cases whenever you find a situation that the existing test cases does not cover.

3. Implement LinkedBinaryTree.java.

   If you need to use any data structures that you implemented in previous assignment, please copy all the related files to the current project.

   If your implementation is buggy and you have hard time to fix the bugs by yourself, please contact instructor or TAs. They will work with you and may give you permission to use library cs2321util.

4. Implement ExpressionTree.java. Read the file block comment carefully to understand the requirement.

Important requirement and information

1. Write meaningful comments while you are implementing the code.

2. Test, Test, Test!!! Test your code for each method separately! Remember many test cases have been provided to you. See the section "Study the new project". Identify the related test cases under folder tests, and run them every time you implement a method. You are encouraged to create more Junit test cases to test your code.

3. NEVER, ever, ever modify ANY classes in the `net.datastructures` package. Don't create new files, don't remove file from net.datastructures. We overwrite this package with the originals when we test your code.

4. DO NOT change the class and method signature. They need to stay as-is. Don't change the number of the arguments, the type of each argument, the order of the arguments. Don't change the return type. Don't change access modifiers.

5. DO NOT import java.util or other packages

6. You may create your own class files, main method and other methods for implementation or testing purpose. If you create constructor, make sure it is PUBLIC, otherwise your program may break when we test it.

7. DELETE System.out.print() in all the methods that will be tested. You should clean up your code before you submit, especially you need to remove all your debugging System.out.print statements in the methods that we ask you to implement. Those print statments will cause your program takes much longer to finish. They will affect the performance when we test the efficiency of your code, and they may even cause the testing code to timeout. But you may keep your main methods and testing methods in place. It is totally fine to have print statements in your main methods or testing methods, because we don't run them when we grader your code.

Submission:

First, look at the following checklist:

1. Did you `import` any thing from `java.util`? . If so, remove them immediately.
2. Is the indentation easily readable? You can have Eclipse correct indentation by highlighting all code and select "Source → Correct Indentation".
3. Are comments well organized and concise?

If you have reviewed your code and all the criteria on the checklist are acceptable, follow the following procedure to export the files.
For Eclipse:

4. Open Eclipse
5. Choose "File → Export".
6. Select General -> Archive File
7. Click on "Next"
8. Click the down arrow for project "List", check the box next to src.
9. Browse for the destination/enter the archive file name as **prog3.zip**
10. Click on "Finish"
11. file prog2.zip will be created under the destination of your choice.

Double check the zip file has the correct folder structure and ALL the java source code files are under the correct folders. Missing any source file or directory could cause compilation failure. When the project does not compile, you will get 0 point.

Submit your file:
Login to Canvas and load your progN.zip to correct the assignment. You may submit many times. We will grade the latest submission only. When you choose the zip file to submit, make sure to submit the correct file. The timestamp on the file is a good indicator.

Grading Criteria:

- `LinkedBinaryTree` and `ExpressionTree` that correctly implements all the required interfaces methods: 55 Points.
- Clear concise code with consistent style and good comments: 5 points