Introduction to Java (Introduction to Java (CS2514)) Assignment 4

Generic Classes (Due: 3 April. Marks: 5)

General Comments

Carefully read the submission guidelines before you submit the assignment.

Like all other exercises, this is an exercise about implementing *maintainable* classes. You should always assume the specifications may change (slightly) and make sure your implementation can accommodate these changes with the minimum amount of effort.

Before you start implementing your classes, please make sure you understand the API. If you don't you'll make your life much more difficult.

Learning Objectives

For this assignment you will learn how to implement a proper generic class for binary trees. To keep things simple you need to provide support for inserting comparable data items into the trees and for printing trees. There should be three methods for printing a tree: one method should print a tree using an in-order traversal, one should print a tree using a pre-order traversal, and one should print a tree using a post-order traversal.

Main Details

For this assignment you will implement a generic binary tree class for representing collections of compatible comparable items. A tree can be an empty leaf node or an internal node consisting of a comparable item and two children, both of which are trees. A non-empty tree implements an order by making sure the nodes of its left subtree are always less than or equal to the value of the root and by making sure the nodes of its right subtree are always greater than the value of the root.

You should implement an instance method for inserting items into a tree and three instance methods for printing trees.

The following describes the method `insert( )` that inserts an item into the tree. When you insert an item into a tree the resulting tree doesn't have to be balanced. However, if a tree is non-empty, the inserted item should always be inserted to the left or to the right of the current root of the tree. Please note that this may lead to trees that are effectively lists. The following creates a tree with 1 at to root, a child 0 to the left of the top, and a child 2 to the right of the top.

```
final Tree<Integer> tree = new Tree<Integer>( );
tree.insert( 1 );
tree.insert( 2 );
tree.insert( 0 );
```

The following creates a tree with 2 at the top, an empty right child, and a left child that has the value 0 at the top.

```
final Tree<Integer> tree = new Tree<Integer>( );
tree.insert( 2 );
tree.insert( 0 );
tree.insert( 1 );
```

The time complexity of the insert method should be proportional to the current height of the tree. There is no need to provide a `remove( )` method.

The following describes the three methods for pretty printing a tree to the terminal. The methods should be called `showPreOrder( )`, `showInOrder( )`, and `showPostOrder( )`. Assuming we have an integer tree with node 1 at the top, left child 0, and right child 2, the methods will print the tree to the terminal as follows.

- `showPreOrder( )`

```
1
  0
  2
```

- `showInOrder( )`

```
  0
1
  2
```

- `showPostOrder( )`

```
  0
  2
1
```

Notice that the members of the subtrees should be shown indented relative to the parent nodes level of indentation. Empty leaf nodes are not shown.

Submission Details

- Please remember that from Assignment 3 on all classes, attributes, and public methods should be commented using a proper `JavaDoc` comment.
- The `JavaDoc` class comment should explain the purpose of the class; *not the purpose of the assignment.*
- Please provide your name and student ID as part of your class `JavaDoc`. You can use the `@author` tag for this:

```Java
@author Java Joe (ID 12345678)
```

- Use the `CS2514` `moodle` site to upload your program as a single *.tgz* archive called *Lab-4.tgz* before 23.55pm, 3 April, 2017. To create the *.tgz* archive, do the following:
  - Create a directory `Lab-4` in your working directory.
  - Copy `Animal.java`, `Main.java` and your other user-defined Java files into the directory. Do not copy any other files into the directory.
  - Run the command '`tar cvfz Lab-4.tgz Lab-4`' from your working directory. The option '`v`' makes `tar` very chatty: it should tell you exactly what is going into the `.tgz` archive. Make sure you check the `tar` output before submitting your archive.
  - Note that file names in `Unix` are case sensitive and should not contain spaces.
- Note that the format of your submission should be `.tgz`: do *not* submit `zip` files, do *not* submit `tar` files, do *not* submit `bzip` files, and do *not* submit `rar` files. If you do, it may not be possible to unzip your assignment.
- No marks shall be awarded for programs that do not compile.