ACS-2947-001

Lab 6

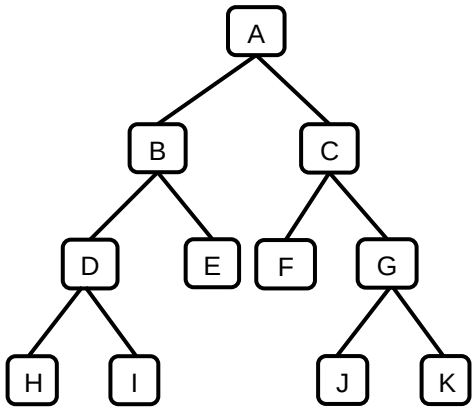
Instructions

- Submit your `.java` files (together in a `Lab6.zip` file) via Nexus.
- Include your name and student number as a comment in every file.

Task

1. Create a class called `LinkedBinaryTree` that implements the `BinaryTree` (which extends `Tree`) and `Position` interfaces. Use the `AbstractBinarytree` and `AbstractTree` classes in your notes/textbook to provide a base for your linked binary tree. Work with your notes from class. Note that:
   - the `Tree` interface is a modified version that does **NOT** include `iterator()` and `positions()`. We will add the tree traversal/iterator in Assignment 3.
   - The `attach()` method is **NOT** required.
   - `toString()` for `LinkedBinaryTree` is **NOT** required for this lab (we will implement Iterators in Assignment 3)

2. In your `Lab6_Driver` class, create a static tree instance and build a simple tree of names that is similar in structure (but not content) to the following tree. That is, the tree contains 11 names. **You must use valid names of your choice instead of the letters.**



3. Include the following **static** methods in your `Lab6_Driver`:
   - `allDescendants` that takes a node/position as parameter and **recursively** displays all the descendants of that position.
     - Note how the `children()` method returns an `Iterable`: you should use this to traverse through the list of children of a node and recursively display their children.

For example, using the node named *C* above, the method displays:

```
F, G, J, K
```

- Illustrate this method by displaying all the descendants of the root node.
- Illustrate this method by displaying all the descendants of the right child of the root node. *You must use the right() method in this.*

- `pathToroot` that displays the path from any given node/position to the root of the tree.
  - Think of this as a position being traversed through the path to get from the position to the root and consider how we've traversed through a linked list from the first to last.

  For example, for the given tree, the following shows the path from the node *H* upwards to the root:

  ```
  Path from H to root(A): H, D, B, A
  ```

  Illustrate this method by displaying the path from the **leftmost node at the deepest level** e.g., node with H above.

4. In your `Lab6_Driver` , include the code to do the following:
   - Display the height of the tree.
   - Display the depth of the tree left child of the root node. *You must use the left() method in this.*
   - Remove the node at the leftmost position of the deepest level. E.g., node H above
   - Display the final size of the tree.

5. Display the output as shown below.

Sample Output

```
The descendants of A: ...
The descendants of C: ...
The path from H to root(A): ...
Tree Height: ...
Depth of left child of the root (B): ...
Removed node with: H
Final tree size: 10
```

Submission

Submit your **`Lab6.zip`** file that includes all the following files (`Position.java`, `Tree.java`, `AbstractTree.java`, `BinaryTree.java`, `AbstractBinaryTree.java`, `LinkedBinaryTree.java`, `Lab6_Driver.java`) and any other files you use via **Nexus**.