Tutorial 6: Trees and `JTree`

Computer Science 214: Data Structures and Algorithms

20 March 2009 Due: 17 April 2009

Instructions

This tutorial is the first of a two-session assignment. As such, the hand-in instructions will be given next week. However, the usual terms and conditions apply: You may not leave early unless your work is finished, and so forth.

Overview

In this tutorial, you will:

- do the basic footwork required to use tree data structures for bookmarks and a browse history in your browser;
- write a tree node class that implements the interface `javax.swing.tree.MutableTreeNode`; and
- use this class in conjunction with `javax.swing.JTree` to write a GUI component capable of navigating through bookmarks (as a test case).

Tutorial

1. Open your existing Eclipse project, and add a new package `cs214.trees`, in which you should put all but the test classes for this tutorial.
2. Write a class `BrowserMutableTreeNode` that implements the interface `javax.swing.tree.MutableTreeNode`. Note that we do not use generics, so the element you store at a particular node must be of type `Object`. This makes it easier when we store different data types in a particular tree, as would be the case when the tree for bookmarks contains both bookmarks and folders.
3. You may not use (or extend) the class `DefaultMutableTreeNode`, found in `javax.swing.tree`. You might, however, gain valuable insight from browsing through its API documentation.
4. Each node must have references to its parent, user object (i.e., element), and children. Each of these references may be `null`. For example, a `null` parent reference indicates that this node is the root of the tree.
5. Keep the children in an `ArrayList` of `MutableTreeNode`s. Assume that every node may have children, but since we expect the (bookmarks) tree to consist mainly of leaves, use *lazy initialisation* for the `ArrayList`. That is, do not instantiate the `ArrayList` until you actually want to add children.

6. Treat `null` values by throwing exceptions *where appropriate*. You will find that existing run-time exceptions, including but not limited to `java.lang.ArrayIndexOutOfBoundsException` and `NoSuchElementException` in `java.util`, can be quite useful. Be very careful when the list of children may still be empty: When the list is `null`, you might easily incur the wrath of a `NullPointerException`.

7. At no point in your tree code may any exception trigger a system crash, either by omission or by design with the `System.exit()` method. You may only use the latter in the main class for your GUI. The idea is that exceptions should be passed along to where they can be handled appropriately, for example, by displaying a message accusing the user of idiocy. And remember: Users quite often *are* idiots, so your application must be able to withstand the onslaught of the imbecilic to some extent.

8. When you remove a node from a tree, remember to null out the relevant references so that the removed node (and user object) is available for garbage collection.

9. An `Enumeration` over the children of a node may very well be called on a leaf node. Your enumeration, which you may implement with anonymous classes that delegate their functionality to that found in `ArrayList`, should behave sensibly in this case. That is, it must never crash, but may throw a `NoSuchElementException`.

10. Test your implementation by making a GUI component that uses `javax.swing.JTree` to display bookmarks. Create the two classes `Bookmark` and `Folder` in the test package `cs214.tests`: A `Bookmark` has a title, description, and URL; a `Folder` has a name and a description. Design these classes well, and you can use them in your final browser. So, use getters and setters—think, for example, if you would allow a bookmark with a `null` URL—and override at least the `equals(Object)` and `toString()` methods inherited from `Object`. There are standard classes available that implement URL functionality.

11. To make the bookmarks clickable, we will in the future need to move beyond the default rendering used by `JTree`. If you complete the tutorial with time to spare, read up on the cell renderers and tree events: Work out how they may be used in the final version of the bookmarks component.

Background Information and Reading

`MutableTreeNode` is documented in the Java API docs. Remember that this interface extends `TreeNode`, so the documentation for the latter should be studied as well. Pay close attention to the specification, in particular where boundary cases should be handled in a special way, for example, where an invalid child index is given.

Read the *How to use trees* section in the Java Tutorial at `http://star.sun.ac.za/java/docs/tutorial/uiswing/components/tree.html`. The relevant information on using `JTree` is either on this page or follows from links on the page. Especially if you need to override the default tree behaviour and rendering, read the parts and follow the examples on tree customisation.