Programming in Smalltalk: Dictionaries and Binary Trees (part 2)

In this part of this exercise you should integrate your new kind of dictionary into the dictionary framework discussed in the lectures (lecture 1, slides 47-58). First load in the code of this framework into your Smalltalk environment. Study the framework as it is for a moment before you go on to integrating your own kind of dictionary.

1. The framework will need some redesigning to fit in your new kind of dictionary. The current root class of the framework for example expects subclasses to implement an `indexAtKey:` message, which doesn't fit a tree-based dictionary. How will you restructure the framework to fit in your dictionary class?

Do the necessary restructuring of the existing framework, and integrate your tree-based dictionary into it. Extend your dictionaries with new messages if necessary to support the requirements of the framework.

2. Can you identify a few places in the new framework and your implementation of part 1 of this exercise where polymorphism is exploited? *(Hint: contrast this implementation with the one for the line-editor from the course Algo>Data2)*
Reflections

Restructuring code without changing its behavior is also called *refactoring*, something you often need to do before actually extending some existing system with some new behavior. There's a book available with a catalog of commonly applicable *refactorings* and the VisualWorks class browser has some support for automating these. It is best to ignore this for the moment for this exercise, but if you have some time left afterwards, you can explore this refactoring support a bit to see how you could have used it to accomplish the same. The refactorings most useful to you in this exercise can be found in the class browser in the following menus: *Class > Other > Convert to Sibling* and *Method > Refactoring > Push Up*. 