// ADT BinaryTree – a proper binary tree (every internal node
// has exactly two children)
// ItemType is the type of element stored in the tree
// NodeType is the type of the tree's nodes

+createBinaryTree ()
// post: new binary tree is created with a single node (the root)

+destroyBinaryTree ()
// post: binary tree is destroyed

+size () : integer {query}
// post: returns the number of nodes in the tree

+isEmpty () : boolean {query}
// post: return true if size() == 0, false otherwise
// (in this version of BinaryTree, this will always return true)

+swapElements (in node1:NodeType, in node2:NodeType)
// post: swaps the elements stored at node1 and node2

+replaceElement (in node:NodeType, in elt:ItemType)
// post: the element stored in the node is replaced by 'elt'

+root () : NodeType {query}
// post: returns the root of the tree

+parent (in node:NodeType) : NodeType {query}
// post: returns the parent of the specified node
// error: if the node is the root

+leftChild (in node:NodeType) : NodeType {query}
// post: returns the left child of the specified node
// error: if the node is a leaf (i.e. has no left child)

+rightChild (in node:NodeType) : NodeType {query}
// post: returns the right child of the specified node
// error: if the node is a leaf (i.e. has no right child)

+sibling (in node:NodeType) : NodeType {query}
// post: returns the sibling of the specified node
// error: if the node has no sibiling (i.e. it is the root)

+isRoot (in node:NodeType) : bool {query}
// post: returns true if the node is the root, false otherwise

+isLeaf (in node:NodeType) : bool {query}
// post: returns true if the node is a leaf, false otherwise

+isInternal (in node:NodeType) : bool {query}
// post: returns true if the node is an internal node (not a leaf), false
// otherwise

+expandLeaf (in node:NodeType)
// post: two new nodes have been created and attached to the tree as
// the left and right children of the specified node
// error: if node is not a leaf

+removeAboveLeaf (in node:NodeType)
// post: removes the specified node and its parent, replacing the parent
// with the node's sibling
// error: if node is not a leaf or is the root

// TreeNode, a node of the tree
// (the tree node will likely have other operations – constructor,
// destructor, way to set the element, etc – but these will be
// private and may be specific to the particular implementation
// of BinaryTree)

+element () : ItemType {query}
// post: returns the element stored at this node

---

notation

- "+" means the method is public
- "in" means that the parameter is an input parameter (i.e. the function makes use of the value passed to it, and no information is returned back to the caller via the param)
- "{query}" means that the method is read-only (no state is changed)