Generic BST

You may have noticed that all our BST so far were written to store Strings. What about storing other types of objects, like Widgets?

You may have noticed also that whenever you accessed the value in a TreeNode, you had to cast the Object to String. Wouldn't it be nice not to have to cast?

That is why Java provides **generic** types. The entire point of generics is that these compiler-supplied casts are guaranteed to be type-safe. If you use generics, and the code compiles, then you are guaranteed not to throw a ClassCastException at run-time.

You last saw the generic <E> notation in Unit 5 on Collections and Generics. It is now time to re-write the BST using generics. Here is the interface with generics:

interface BSTinterface<E>  
 {  
 public int size();  
 public boolean contains(E element);  
 public E add(E element);  
 //public E addBalanced(E element);  
 public E remove(E element);  
 public String min();  
 public String max();  
 public String display();  
 public String toString();  
 public ArrayList<E> toList();  
 }

Here are the headers for a generic BST and a generic TreeNode

public class BST\_Generic<E extends Comparable<E>>

implements BSTinterface<E>

class TreeNode<E>

Assignment

The shell is called BST\_Generic. The shell contains BSTinterface and two places in which to copy your non-generic code for BST and TreeNode. Then modify your code to handle generics.

The driver is called BST\_Generic . The first part puts some widgets in the BST, prints the tree, and calls some methods. The second part is a stress test. If you run a stress test on a BST with objects, how can you get the computer to test if the objects are "in order"?