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 Integers or Widgets?

You also may have noticed that whenever you accessed the value stored 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 to let the compiler do the casting, which guarantees that the code is *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. As you may remember, the E is a placeholder at compile-time which will be replaced by an actual object during run-time. We will now re-write 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 E min();  
 public E max();  
 public String display();  
 public String toString();  
 public List<E> toList(); //returns an in-order list of E  
 }

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<E> and two places in which to copy your non-generic code for BST and TreeNode. Then modify your code to handle generics:

1. you are no longer using TreeNode, you are using TreeNode<E>.
2. you are no longer using Objects, you are using E.
3. you never have to cast anything. Specifically, do not use (E).

The driver is called BST\_Generic\_Driver which puts some widgets in a BST, prints the tree, and calls some methods. The BST\_Generic\_Stress adds N random widgets and then randomly removes them one at a time. After each removal, the code checks to see if the tree is still in order. You must write the checkInOrder method.