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 its original data type. Wouldn't it be nice not to have to cast?

What we want is a way to compile the source code with the original data types, before knowing what the data types actually are. That’s what Java’s **generic** **types** do. The compiler performs the casts and checks the code to guarantee that the code is *type-safe* and will never throw a ClassCastException at run-time. All that magic happens through the <E> and E notation.

The Java Collections are all generic. You can see the <E> and E on the cheat sheet. As you may remember, the <E> and E are placeholders at compile-time which will be replaced by an actual object during run-time. We will now re-write our BST so that any E that isa Comparable can be stored in our BST. We also have to re-write our TreeNode class to handle generics.

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

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

implements BST\_Generic\_interface<E>

class TreeNode<E>

Your BST\_Generic class must implement this interface, which is written generically:

interface BST\_Generic\_interface<E>  
 {  
 public int size();  
 public TreeNode<E> getRoot() ;  
 public boolean contains(E obj);  
 public void add(E obj); //does not balance  
 public void addBalanced(E obj); //AVL  
 public void remove(E obj); //does not balance  
 public E min();  
 public E max();  
 public String display();  
 public String toString(); //in-order traversal  
 }

Assignment

The shell is called BST\_Generic. The shell contains BST\_Generic\_interface<E> and two places in which to copy your non-generic code for BST and TreeNode. Then modify BST and TreeNode to handle generics. Remember:

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. Specifically, a cast by using (E) should not appear in your code.

The driver is called BST\_Generic\_Driver which builds trees of Doubles, then Integers, then Widgets.