Generic BST

Goals of the Assignment

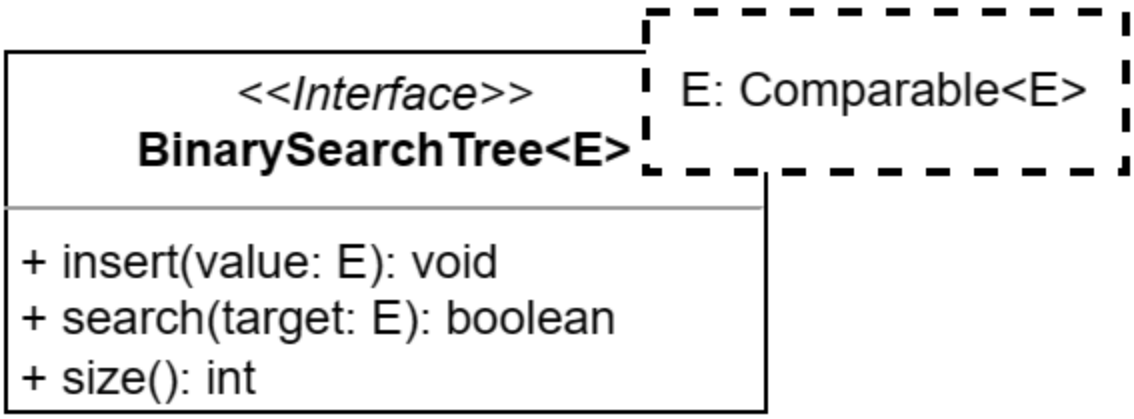
The goal of this assignment is to create a generic `BinarySearchTree` class. You will modify versions of the `BinarySearchTree`, `NodeBST`, `Pokedex`, and `Pokemon` classes that you created during the first two days of this unit. Since BST's are inherently ordered, they will need to understand and use `Comparable` objects to be built.

As always, you are expected to demonstrate good software engineering practices including the use of version control and testing. ***Read this document in its entirety*** before asking the course staff for help.

Activities

1. Start by refactoring the `BinaryNode` class to work with any type instead of only `int`.
    a. Add a generic type parameter to the class declaration.
    b. Replace all instances of the `int` value with the generic type parameter.
    c. Update the `main` method that you wrote in class to use the new generic `BinaryNode`. Use a type of value that is not an integer, e.g. a string.

2. Modify the `BinarySearchTree` so that it is generic using the following UML diagram as a reference:



    Where:
    a. The generic type for `BinarySearchTree` is `<E extends Comparable<E>>`. This enforces that the type must be a `Comparable`, which we will require since BSTs are an inherently ordered data structure.

3. Update the `NodeBST` class so that it implements the generic `BinarySearchTree` interface correctly.

   a. You will need to modify the private `binaryInsert` and `binarySearch` helper methods to use the `compareTo` method on the objects being stored in the nodes. To be clear, you will no longer use the less than (`<`) operator to try to compare values directly; instead, you will call the `compareTo` method to compare the `target` to the node's value, and insert into/search the correct subtree depending on whether the value is negative (left) or not (right).

4. Update `Pokedex` to incorporate the generic BinarySearchTree (BST) interface and/or `NodeBST` class to work with `Pokemon` ***objects*** (not integers) using the `Pokemon` class that you wrote in this unit. In the `main` method of the `Pokedex` class, create an instance of `Pokedex` with at least five `Pokemon` and test all methods.

Submission Instructions

You must ensure that your solution to this assignment is pushed to GitHub *before* the start of the next lecture period.