Обобщенные классы

Пример программы

Написать обобщённый класс `BinaryTree`, реализующий бинарное дерево. Значения в узлах могут быть произвольного типа. Класс должен содержать методы:

- `Add` – добавить узел с указанным значением в дерево в качестве нового листа,
- `Exist` – проверить, содержит ли дерево узел с указанным значением.

Написать программу, использующую этот класс.

```
using System;

namespace Tree
{
    public class BinaryTree<T> where T : IComparable
    {   // IComparable говорит о том, что экземпляры T можно сравнивать
        class Node
        {
            public Node Left, Right;
            public T Value;
            public Node(T value)
            {
                Left = Right = null;
                Value = value;
            }
        }
        Node root = null;

        public void Add(T val)
        {
            if (root == null)
            {
                root = new Node(val);
                return;
            }
            var current = root; // Ссылка на текущий узел

            while (true)
            {
                // CompareTo из IComparable, без него значения не сравнить
                if (val.CompareTo(current.Value) <= 0)  // val <= current.Value
                {
                    if (current.Left == null)
                    {
                        current.Left = new Node(val);
                        return;
                    }
                    else
                        current = current.Left;
                }
                else
                {
                    if (current.Right == null)
                    {
                        current.Right = new Node(val);
                        return;
                    }
                    else
                        current = current.Right;
                }
            }
```

```
        }

        public bool Exist(T val)
        {
            var current = root;
            while (current != null)
            {
                if (val.CompareTo(current.Value) == 0)      // val == current.Value
                   return true;
                else if (val.CompareTo(current.Value) < 0)  // val < current.Value
                    current = current.Left;
                else                                        // val > current.Value
                    current = current.Right;
            }
            return false;
        }

    }

    class MainClass
    {
        public static void Main(string[] args)
        {
            var t = new BinaryTree<int>();
            t.Add(4);
            t.Add(2);
            t.Add(1);
            t.Add(3);
            t.Add(6);
            t.Add(5);
            t.Add(7);
            Console.WriteLine("Введите значение для поиска");
            int x = Convert.ToInt32(Console.ReadLine()); ;
            if (t.Exist(x))
                Console.WriteLine("Дерево содержит узел со значением {0}", x);
            else
                Console.WriteLine("Дерево не содержит узел со значением {0}", x);
        }
    }
}
```

Метод `Exist` может иметь следующий вид:

```
        public bool Exist(T val)
        {
            return Exist(root, val);
        }

        bool Exist(Node current, T val)
        {
            if (current == null)
                return false;
            if (val.CompareTo(current.Value) == 0)
                return true;
            else if (val.CompareTo(current.Value) < 0)
                return Exist(current.Left, val);
            else
                return Exist(current.Right, val);
        }
```