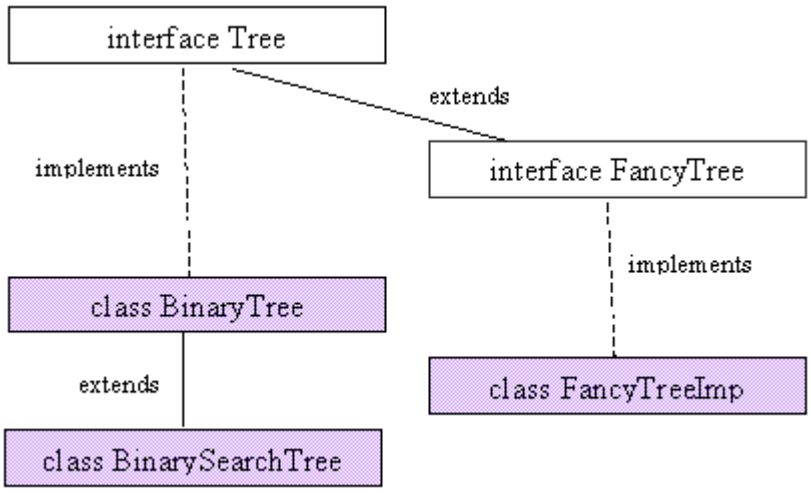
Дисциплина: *СД и ООП* **Тема:** *Двоично дърво*
Специалност: *КН 1 курс, 3 и 4 групи* **Дата**: *05.05.2010 г.*

Задача 1: Дадена е йерархията от интерфейси и класове:



Интерфейсът **Tree** представя операции на *дърво с елементи от тип* ***Object***:

```
public interface Tree {
        public void add(Object element);
        public Object remove(Object element);
        public boolean contains(Object element);
        public boolean isEmpty();
         public int size();
        public int height();
        public int numberOfLiefs();
        public void clear();
        public java.util.Iterator iterator();
        public java.util.Iterator treeInorderIterator();
        public java.util.Iterator treePreorderIterator();
        public java.util.Iterator treePostorderIterator();
        public java.util.Iterator treeBreadthFirstIterator();
        public Object clone();
        public String toString();
}
```

Интерфейсът **FancyTree** представя операции над дървета:

```
public interface FancyTree extends Tree {
        public boolean sameElements(FancyTree arg);
        public boolean sameStructure(FancyTree arg);
        public boolean sameTrees(FancyTree arg);
        public boolean sameHeight(FancyTree arg);
        public boolean sameNumberOfLiefs(FancyTree arg);
        public boolean containsAll(FancyTree arg);
        public boolean addAll(FancyTree arg);
        public boolean removeAll(FancyTree arg);
        public boolean retainAll(FancyTree arg);
}
```

1.Да се реализира клас **BinaryTree** – *двоично дърво с елементи от тип* ***Object***, съгласно спецификацията:

```
public class BinaryTree implements Tree {
        //Данни
        protected TreeNode root;
```

```
        //Конструктори
        public BinaryTree() { this(null); }
        public BinaryTree(TreeNode n) { root = n; }

        //Методи за достъп
        public TreeNode getRoot() { return root; }
        public BinaryTree getLeftSubtree() { return isEmpty() ? null : new BinaryTree(root.left); }
        public BinaryTree getRightSubtree() { return isEmpty() ? null : new BinaryTree(root.right); }

        //Реализация на методите на интерфейса Tree
}
```

Класът **TreeNode** представя *възел на двоично дърво с елементи от тип **Object***:

```
public class TreeNode {
        //Data
        protected Object data;
        protected TreeNode left,right;

        //Constructor
        public TreeNode(Object d) { data = d; left = right = null; }

        //Methods
        public Object data() { return data; }
        public Object setData(Object d) { return data = d; }
        public TreeNode left() { return left; }
        public TreeNode setLeft(TreeNode n) { return left = n; }
        public TreeNode right() { return right; }
        public TreeNode setRight(TreeNode n) { return right = n; }
}
```

Класът **BTreeInorderIterator** представя итератор за *смесено обхождане на двоично дърво с елементи от тип **Object***:

Класът **BTreePreorderIterator** представя итератор за *низходящо обхождане на двоично дърво с елементи от тип **Object***:

Класът **BTreePostorderIterator** представя итератор за *възходящо обхождане на двоично дърво с елементи от тип **Object***:

Класът **BTreeBreadthFirstIterator** представя итератор за *обхождане по нива на двоично дърво с елементи от тип **Object***:

2.Да се реализира клас **BinarySearchTree** – *двоично дърво на търсене с елементи от тип **Object***, съгласно спецификацията:

```
public class BinarySearchTree extends BinaryTree implements Tree {
        //Данни
        protected java.util.Comparator c;

        //Конструктори
        public BinarySearchTree() { this(new NaturalComparator()); }
        public BinarySearchTree(java.util.Comparator c) { this(c,null); }
        public BinarySearchTree(java.util.Comparator c,TreeNode n) { super(n); this.c = c; }

        //Методи за достъп – наследени от класа BinaryTree
        public java.util.Comparator getComparator() { return c; }

        //Предефиниране на методи за достъп, наследени от класа BinaryTree
        public BinaryTree getLeftSubtree() { return isEmpty() ? null : new BinarySearchTree(c,root.left); }
        public BinaryTree getRightSubtree() { return isEmpty() ? null : new BinarySearchTree(c,root.right); }
```

```
        //Наследяване на методите на интерфейса Tree, реализирани в класа BinaryTree

        //Предефиниране на методи, наследени от класа BinaryTree
        public void add(Object element) {...}
        public Object remove(Object element)  {…}
        public boolean contains(Object element) {…}
}
```

Класът **NaturalComparator** представя обект за сравняване, използвайки метода compareTo:

```
public class NaturalComparator implements java.util.Comparator {
        public int compare(Object a,Object b) {
                return ((java.lang.Comparable)a).compareTo(b);
        }

        public boolean equals(Object b) {
                return (b != null) && (b instanceof NaturalComparator);
        }
}
```

3.Да се реализира клас **FancyTreeImp**, съгласно спецификацията:

```
public class FancyTreeImp implements Tree, FancyTree {
        //Данни
        private Tree elements;

        //Конструктор
        public FancyTreeImp(Tree arg) { elements = arg; }

        //Tree methods implementation
        //Реализация на методите на интерфейса Tree
        //Реализация на методите на интерфейса FancyTree
}
```