# ΑΣΚΗΣΗ 5Α

## Δημιουργία-Διαχείριση Δυαδικού Δέντρου Αναζήτησης (συνέχεια – κόμβοι με περιεχόμενα Objects)

Στο δυαδικό δέντρο αναζήτησης της προηγούμενης άσκησης (Εργαστήριο 7), θεωρήστε ότι τα δεδομένα του είναι τύπου object και κάντε τις απαραίτητες τροποποιήσεις.

**Άσκηση 5Α.1**
Εμπλουτίστε την κλάση **BSTree** με τις μεθόδους:

```
public int numberOfNodes( )
// Υπολογίζει και επιστρέφει το πλήθος των κόμβων του δένδρου
public int treeHeight( )
// Υπολογίζει και επιστρέφει το ύψος του δένδρου
public int treeHeight(Object data)
// Υπολογίζει και επιστρέφει το ύψος του δένδρου για τον συγκεκριμένο κόμβο
// (με δημιουργία και χρήση equals ή compareTo)
```

**Άσκηση 5Α.2**
Θεωρήστε ότι στο δυαδικό δέντρο αναζήτησης έχουν εισαχθεί φοιτητές σύμφωνα με τη διάταξη του αριθμού μητρώου τους (βλέπετε Παράρτημα στη συνέχεια). Κάντε τις κατάλληλες τροποποιήσεις στον αλγόριθμο ταξινόμησης του προηγούμενου εργαστηρίου ώστε ο πίνακας που προκύπτει να έχει ταξινομημένους τους φοιτητές με βάση τον αριθμό μητρώου.

Αρχείο TreeNode.java

```
public class TreeNode
{
        // Instance fields (data members)
        private TreeNode left;
        private object item;
        private TreeNode right;

        // Methods
        public object getNodeData( )
        public TreeNode getLeftNode( )
        public TreeNode getRightNode( )
        public boolean isLeaf( )
        public void setLeftNode(TreeNode node)
        public void setRightNode(TreeNode node)
}
```

```
public class BSTree
{
        // Instance field (data member)
        private TreeNode root;

        // Methods
        public BSTree( )
        {       root = null;     }

                public boolean isEmpty( )
        {       return (root == null);  }

        public void insertElement(object data)
        {       if (isEmpty( ))
                        root = new TreeNode(data);
                else
                        insertNode(data,root); }

        public void inOrderTraversal( )
        {       System.out.println("INORDER TRAVERSAL");
                inOrder(root);
                System.out.println( );   }

        public void preOrderTraversal( )
        {       System.out.println("PREORDER TRAVERSAL");
                preOrder(root);
                System.out.println( );   }

        public void postOrderTraversal( )
        {       System.out.println("POSTORDER TRAVERSAL");
                postOrder(root);
                System.out.println( );   }


        // RECURSIVE PRIVATE METHODS
        // Υλοποιήστε τις παρακάτω:

        private void insertNode(object data, TreeNode node)
        private void inOrder(TreeNode node)
        private void preOrder(TreeNode node)
        private void postOrder(TreeNode node)
}
```

**ΠΑΡΑΡΤΗΜΑ: Σύγκριση δύο αντικειμένων μεταξύ τους**

Για να γίνει δυνατόν να συγκρίνουμε το περιεχόμενο δύο κόμβων του δυαδικού δέντρου η κλάση που ορίζει τον τύπο του περιεχομένου πρέπει να υλοποιεί το interface Comparable.

***public interface Comparable***: Το interface αυτό απαιτεί να ορίζεται η διάταξη των αντικειμένων κάθε κλάσης που το υλοποιεί. Παραδείγματος χάριν, η κλάση ***String***, που υλοποιεί το interface Comparable, ορίζει σαν διάταξη μεταξύ των συμβολοσειρών (που είναι τα αντικείμενα της κλάσης) την αλφαβητική σειρά. Αυτό σημαίνει, ότι όταν δοθούν δύο τυχαίες συμβολοσειρές είναι ξεκάθαρο ποια από τις δύο προηγείται της άλλης σε αλφαβητική διάταξη.

Κάθε κλάση που υλοποιεί το interface Comparable είναι υποχρεωμένη να ορίζει τη μέθοδο **compareTo** που έχει τη γενική μορφή:

***public int compareTo(Object ob):*** Συγκρίνει το αντικείμενο της κλάσης (this object) στο οποίο θα εφαρμοστεί με το αντικείμενο ***ob***, ως προς τη διάταξή τους. Επιστρέφει:

- έναν αρνητικό ακέραιο αν το αντικείμενο είναι μικρότερο σε διάταξη από το αντικείμενο ***ob***
- μηδέν (0) αν το αντικείμενα είναι ίσα μεταξύ τους
- ένα θετικό ακέραιο αν το αντικείμενο είναι μεγαλύτερο σε διάταξη από το αντικείμενο ***ob***

**Παράδειγμα**: Ας υποθέσουμε ότι η κλάση Student ορίζει έναν σπουδαστή από το επίθετο, το όνομα και τον αριθμό μητρώου του και ας υποθέσουμε ότι ορίζουμε την έννοια της διάταξης ανάμεσα σε δύο σπουδαστές με τη βοήθεια του αριθμού μητρώου. Η κλάση του Student θα μπορούσε να υλοποιεί το interface Comparable με την αντίστοιχη comareTo, όπως φαίνεται στο παρακάτω τμήμα κώδικα:

```
public class Student implements Comparable {
  private String lastName;
  private String firstName;
  private int am;

. . .

  public int compareTo(Object S) {
    if (this.am < ((Student))S.am) {
      return -1;
    } else if (this.am > ((Student)S).am) {
      return 1;
    } else return 0
  }
}
```

**Εναλλακτική Υλοποίηση** της compareTo με χρήση generics:

```
public class Student implements Comparable<Student> {
  private String lastName;
  private String firstName;
  private int am;

. . .

  public int compareTo(Student S) {
    if (this.am < S.am) {
```

```
      return -1;
    } else if (this.am > S.am) {
      return 1;
    } else return 0
  }
}
```

**<u>Περισσότερα</u> για το interface Comparable**:
https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Comparable.html