ΑΣΚΗΣΗ 5

Δημιουργία-Διαχείριση Δυαδικού Δέντρου Αναζήτησης

(Βλέπε http://www.iee.ihu.gr/~demos/teaching_GR.html)

Άσκηση 5.1

Να υλοποιήσετε σε Java ένα δυαδικό δέντρο αναζήτησης *(Binary Search Tree)*. Στο δέντρο θα εισάγετε τους αριθμούς **40, 15, 25, 50, 20, 10, 70, 55, 45, 5, 18** με τη σειρά που εμφανίζονται και στη συνέχεια θα εκτελέσετε τις 3 διελεύσεις *(inorder, postorder, preorder)*. Το αποτέλεσμα της εκτέλεσης του προγράμματος θα πρέπει να είναι το παρακάτω:

INORDER TRAVERSAL
5 10 15 18 20 25 40 45 50 55 70
PREORDER TRAVERSAL
40 15 10 5 25 20 18 50 45 70 55
POSTORDER TRAVERSAL
5 10 18 20 25 15 45 55 70 50 40

Να ολοκληρώσετε την υλοποίηση των παρακάτω κλάσεων:

Αρχείο TreeNode.java

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

        // Methods
        public int 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(int 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(int data, TreeNode node)
        private void inOrder(TreeNode node)
        private void preOrder(TreeNode node)
        private void postOrder(TreeNode node)
}
```

Αρχείο BSTreeManagement.java

```
public class BSTreeManagement
{

    public static void main(String args[ ])
    {
        int matrix[ ] = {40, 15, 25, 50, 20, 10, 70, 55, 45, 5 };

        BSTree tree = new BSTree( );

        for (int i=0; i<matrix.length; i++)  tree.insertElement(matrix[i]);

        tree.inOrderTraversal( );
        tree.preOrderTraversal( );
        tree.postOrderTraversal( );
    }
}
```

Άσκηση 5.2

Υλοποιήστε τις παρακάτω τρείς μεθόδους

- **public boolean search(int data):** Βρίσκει εάν στο δυαδικό δέντρο υπάρχει κόμβος με περιεχόμενο τον ακέραιο αριθμό **data**. Επιστρέφει true ή false ανάλογα
- **public int treeHeigth( ):** Βρίσκει και επιστρέφει το ύψος ενός δυαδικού δέντρου
- **public int treeHeigth(int data):** Βρίσκει και εφόσον υπάρχει επιστρέφει το ύψος του κόμβου του δυαδικού δέντρου με περιεχόμενο τον ακέραιο αριθμό **data.** Εάν εν υπάρχει επιστρέφει -1.

Άσκηση 5.1

Η εφαρμογή της μεθόδου inOrderTraversal σε ένα δυαδικό δέντρο αναζήτησης εμφανίζει στην οθόνη τα δεδομένα του δέντρου ταξινομημένα (σε αύξουσα τάξη). Ξεκινώντας από τον κώδικα της «private void inOrder(TreeNode node)» και κάνοντας τις κατάλληλες τροποποιήσεις δημιουργήσετε έναν **αλγόριθμο ταξινόμησης πίνακα**.