```java
import java.util.Scanner;
import unit4.collectionsLib.*;
import unit4.utilsLib.BinTreeUtils;

public class BTNmethods
{
    public static void main(String[] args)
    {
        int maxNodes = 15;      //        מספר הצמתים בעץ
        int low = -10;          //   value ערך מינימאלי של
        int high = 10;          //   value ערך מקסימאלי של

        BinNode<Integer> bt = TreeUtils.buildRandomTree(maxNodes, low, high);
        TreeUtils.showTree(bt, "bt");

        System.out.println(numNodes(bt) + " : מספר הצמתים בעץ ");
        System.out.println(sumNodes(bt) + " : סכום הצמתים בעץ ");
        System.out.println();

        System.out.println(numPositive(bt) + " : מספר הצמתים החיוביים בעץ ");
        System.out.println(sumPositive(bt) + " : סכום הצמתים החיוביים בעץ ");
        System.out.println();

        System.out.println(countLeftSons(bt) + " : מספר הבנים השמאליים");
        System.out.println(sumLeftSons(bt) + " : סכום הבנים השמאליים");
        System.out.println();

        System.out.println(countRightSons(bt) + " : מספר הבנים הימניים");
        System.out.println(sumRightSons(bt) + " : סכום הבנים הימניים");
        System.out.println();

        System.out.println(countLeaves(bt) + " : מספר העלים");
        System.out.println(sumLeaves(bt) + " : סכום העלים");

        System.out.println(maxInTree(bt) + " : ערך מקסימאלי בעץ");
    }









    /*****************************************
     *      אוסף פעולות שימושיות בעץ בינארי     *
     *****************************************/

    //--- מספר הצמתים בעץ ---
    public static int numNodes (BinNode <Integer> bt)
    {
        if (bt == null)
            return 0;
        return  1 +                         //-- בגלל הצומת הנוכחי
                numNodes(bt.getLeft()) +    //-- מספר הצמתים בבן השמאלי
                numNodes(bt.getRight());    //-- מספר הצמתים בבן הימני
    }

    //--- סכום הצמתים בעץ ---
    public static int sumNodes (BinNode <Integer> bt)
    {
        if (bt == null)
            return 0;
        return  bt.getValue() +             //  ערך הצומת הנוכחי
                sumNodes(bt.getLeft()) +    //  סכום הצמתים בבן השמאלי
                sumNodes (bt.getRight());   //  סכום הצמתים בבן הימני
    }

    //--- *** מספר הצמתים המקיימים תנאי **** ---
    //  --- מספר הצמתים החיוביים בעץ ---
    public static int numPositive (BinNode <Integer> bt)
    {
        if (bt == null)
            return 0;               //---         זהו התנאי המבוקש
        if (bt.getValue() > 0)      //---  אפשר להחליף אותו בתנאי אחר
```

```java
            return  1 + numPositive(bt.getLeft()) +
                        numPositive (bt.getRight());
        return  numPositive(bt.getLeft()) +
                numPositive(bt.getRight());
    }

    //  --- סכום הצמתים החיוביים בעץ ---
    public static int sumPositive (BinNode <Integer> bt)
    {
        if (bt == null)
            return 0;
        if (bt.getValue() > 0)
            return  bt.getValue() +
                    sumPositive(bt.getLeft()) +
                    sumPositive (bt.getRight());
        return  sumPositive(bt.getLeft()) +
                sumPositive(bt.getRight());
    }

    //  --- מספר הבנים השמאליים בעץ ---
    public static int countLeftSons (BinNode <Integer> bt)
    {
        if (bt == null)
            return 0;
        if (bt.getLeft() != null)
            return  1 + countLeftSons(bt.getLeft()) +
                        countLeftSons(bt.getRight());
        return  countLeftSons (bt.getRight());
    }

    //  --- מספר הבנים הימניים בעץ ---
    public static int countRightSons (BinNode <Integer> bt)
    {
        if (bt == null)
            return 0;
        if (bt.getRight() != null)
            return  1 + countRightSons(bt.getLeft()) +
                        countRightSons(bt.getRight());
        return  countRightSons (bt.getLeft());
    }

    //  --- סכום הבנים השמאליים בעץ ---
    public static int sumLeftSons (BinNode <Integer> bt)
    {
        if (bt == null)
            return 0;
        if (bt.getLeft() != null)
            return  bt.getLeft().getValue() +   // ערך הבן השמאלי
                    sumLeftSons(bt.getLeft()) +
                    sumLeftSons (bt.getRight());
        return  sumLeftSons (bt.getRight());
    }

    //  --- סכום הבנים הימניים בעץ ---
    public static int sumRightSons (BinNode <Integer> bt)
    {
        if (bt == null)
            return 0;
        if (bt.getRight() != null)
            return  bt.getRight().getValue() +
                    sumRightSons(bt.getLeft()) +
                    sumRightSons (bt.getRight());
        return  sumRightSons (bt.getLeft());
    }

    //---        ?האם עלה      ---
    public static boolean isLeaf (BinNode <Integer> bt)
    {
        if (bt == null)
            return false;
        if (bt.getLeft() == null && bt.getRight() == null)
            return true;
        return false;

        /*-----  אפשרות נוספת ------
         * null  הבנים יכולים להיות שווים רק כאשר הם
```

```java

          return (bt.getLeft() ==  bt.getRight()) ;

          */
    }

    //--- מספר העלים בעץ ---
    public static int countLeaves (BinNode <Integer> bt)
    {
        if (bt == null)
            return 0;
        if (isLeaf(bt))
            return 1;
        return countLeaves(bt.getLeft()) + countLeaves(bt.getRight());
    }

    //  --- סכום העלים בעץ ---
    public static int sumLeaves (BinNode <Integer> bt)
    {
        if (bt == null)
            return 0;
        if (isLeaf(bt))
            return bt.getValue();
        return sumLeaves(bt.getLeft()) + sumLeaves(bt.getRight());
    }

    //--- עץ בינארי לא ריק ---
    //--- הערך המקסימאלי בעץ ---
    public static int maxInTree (BinNode <Integer> bt)
    {
        if (bt == null)
            return 0;
        int maxLeft = maxInTree(bt.getLeft());
        int maxRight = maxInTree(bt.getRight());

        return Math.max(bt.getValue(), Math.max(maxLeft, maxRight));
    }
}

```