Лабораторные работы №№9-10. Работа с деревом поиска

Формулировка задания: написать программу, которая позволяет работать с деревом поиска, осуществляя следующие операции:

- 1) поиск вершины с заданным ключом в дереве;
- 2) вставка вершины с заданным ключом в дерево;
- 3) удаление вершины с заданным ключом из дерева поиска;
- 4) вывод структуры дерева.

В случае, если искомая вершина содержится в дереве, необходимо выдать информацию о ее левом и правом потомках. В противном случае выдать информацию об отсутствии искомой вершины в дереве.

Входные данные:

- 1) ключ искомой вершины;
- 2) ключ добавляемой в дерево вершины;
- 3) ключ удаляемой из дерева вершины.

Выходные данные:

- 1) результат поиска;
- 2) информация о структуре дерева.

Важно: способ выдачи информации о структуре дерева определяется разработчиком самостоятельно. Как вариант эта информация может быть представлена следующим образом: для каждой вершины дерева выдается информация о ее левом и правом потомках.

Результат работы должен быть представлен в виде исходного кода программы. Наличие комментариев в коде программы обязательно!

Ниже приведен листинг классов на языке Java для работы с деревом поиска (реализованы функции поиска узла с заданным ключом и добавления узла в дерево).

```
// tree.java
class Node { // Узел дерева
    public int iData; // Ключевое поле
    public Node leftChild;
    public Node rightChild;
    public void displayNode() {// Вывод данных узла
        System.out.print("(" + iData + ", ");
        if (leftChild != null)
            System.out.print(leftChild.iData);
        else System.out.print("null");
        if (rightChild != null)
        System.out.print(", " + rightChild.iData + ")");
else System.out.print(", null)");
    }
class Tree { // Camo depeβo
    private Node root; // Корневой узел
    public Tree() {// Конструктор
        root = null; // Пока нет ни одного узла
    public Node find(int key){ // Поиск узла с заданным ключом
        // (предполагается, что дерево не пустое)
        Node current = root; // Начать с корневого узла
        while (current.iData != key) {// Пока не найдено совпадение
```

```
if (key < current.iData) // Двигаться налево?
                current = current.leftChild;
            else // Или направо?
                current = current.rightChild;
            if (current == null) // Если потомка нет
                return null; // nouck завершился неудачей
        return current; // Элемент найден
    public void insert(int id) { // Вставка узла
        Node newNode = new Node(); // Создание нового узла
        newNode.iData = id; // Вставка данных
        if (root == null) // Корневой узел не существует
            root = newNode;
        else {// Корневой узел занят
            Node current = root; // Начать с корневого узла
            Node parent;
            while (true){ // (внутренний выход из цикла)
                parent = current;
                if (id < current.iData){ // Двигаться налево?</pre>
                    current = current.leftChild;
                    if (current == null){ // Если достигнут конец цепочки,
                        parent.leftChild = newNode; // вставить слева
                        return;
                    }
                } else { // или направо?
                    current = current.rightChild;
                    if (current == null){ // Если достигнут конец цепочки,
                        parent.rightChild = newNode; // вставить справа
                        return;
                    }
                }
           }
       }
   }
}
// TreeApp.java
import java.util.Scanner;
public class TreeApp {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Tree theTree = new Tree();
        theTree.insert(50);
        theTree.insert(25);
        theTree.insert(75);
        theTree.insert(12);
        theTree.insert(37);
        theTree.insert(43);
        theTree.insert(30);
        theTree.insert(33);
        theTree.insert(87);
        theTree.insert(93);
        theTree.insert(97);
        System.out.println("Введите ключ для поиска:");
        int key = input.nextInt();
        Node found = theTree.find(key);
        if (found != null)
            found.displayNode();
        else System.out.println("Заданный ключ в дереве не найден");
    }
}
```