

Hausaufgaben 4

11.04.2023

Abgabe der Lösung am 16.04.2023

Aufgabe 1

Schreiben Sie eine Klasse `BinTree` für einen binären Suchbaum, der `int`-Werte enthält. Entwerfen Sie eine geeignete Knotenklasse `Node` mit Attributen für die Zahl (`data`) und Referenzen auf den rechten und linken Kindknoten.

Implementieren Sie die folgenden Funktionen:

- `private Node getNode(int x)`
Suchen eines Wertes. Falls `x` nicht vorhanden ist, wird `null` zurückgegeben.
- `private Node getParentNode(int x)`
Suchen des Elternknotens eines Wertes. Falls `x` nicht vorhanden oder in der Wurzel steht, wird `null` zurückgegeben.
- `public void insert(int x)`
Einfügen eines Wertes. Falls versucht wird, einen Wert einzufügen, der schon vorhanden ist, wird eine `ArithmeticException` ausgelöst.
- `public void clear()`
Löschen aller Daten.
- `public void remove(int x)`
Löschen eines Wertes. Falls versucht wird, einen Wert zu löschen, der nicht vorhanden ist, wird eine `ArithmeticException` ausgelöst.

Hinweis: Es ist nicht nötig, die Bäume auszubalancieren.

Testen Sie anschließend Ihren binären Suchbaum mit folgender `main`-Methode:

```
public static void main(String[] args) {
    BinTree tree = new BinTree();
    tree.insert(20);
    tree.insert(10);
    tree.insert(30);
    tree.insert(50);
    int[] testcases = { 30, 35, 50 };
    for (int testcase : testcases) {
        Node node = tree.getNode(testcase);
        if (node == null) {
            System.out.println("Knoten " + testcase + " nicht gefunden.");
        } else {
            System.out.println("Knoten " + testcase + " gefunden: " + node.data);
        }
    }
    tree.remove(30);
    System.out.println("Knoten gelöscht: 30");
    System.out.println("Elternknoten von 50: " + tree.getParentNode(50).data); // 20
}
```