TP2 - Correction

```java
package abr;

import java.util.ArrayList;
import java.util.List;
import java.util.NoSuchElementException;

public class ScoreTree {
    private static class TNode {
        int number;
        String name;
        double score;
        TNode left, right;

        public TNode(int number, String name, double score, TNode left, TNode right)
    {
            this.number = number;
            this.name = name;
            this.score = score;
            this.left = left;
            this.right = right;
        }
    }

    private TNode root;

    public ScoreTree() {
        root = null;
    }

    public void listing(double score) {
        listing(score, root);
    }
    private static void listing(double score, TNode r) {
        if (r != null) {
            listing(score, r.left);
            if (r.score == score)
                System.out.println(r.number + "  " + r.name);
            listing(score, r.right);
        }
    }

    public double scoreOf(int num) {
        TNode r = root;
        while (r != null && r.number != num) {
            if (r.number < num)
                r = r.right;
            else
```

```java
                r = r.left;
        }
        if (r != null)
            return r.score;
        else
            throw new NoSuchElementException();
    }

    public void add(int num, String name, double score) {
        TNode current = root, parent = null;
        while (current != null) {
            parent = current;
            if (num > current.number)
                current = current.right;
            else if (num == current.number)
                throw new IllegalStateException();
            else
                current = current.left;
        }
        TNode toAdd = new TNode(num, name, score, null, null);
        if (root == null)
            root = toAdd;
        else
            if (num > parent.number)
                parent.right = toAdd;
            else
                parent.left = toAdd;
    }

    public double bestScore() {
        if (root == null)
            throw new NoSuchElementException();
        else
            return bestScore(root);
    }

    private static double bestScore(TNode r) {  // r != null
        if (r.left == null && r.right == null)
            return r.score;
        else if (r.left == null)
            return Math.max(r.score, bestScore(r.right));
        else if (r.right == null)
            return Math.max(r.score,  bestScore(r.left));
        else
            return Math.max(r.score,  Math.max(bestScore(r.left), bestScore(r.right)));
    }

    public void bestStudent() {
        listing(bestScore());
    }

    private static void successList(TNode r, List<Integer> list, double score) {
        if (r != null) {
            successList(r.left, list, score);
            if (r.score >= score)
                list.add(r.number);
            successList(r.right, list, score);
        }
    }

    public List<Integer> successList(double score) {
        List<Integer> list = new ArrayList<Integer>();
        successList(root, list, score);
        return list;
    }

    public void remove(int num) {
        TNode x,        // Noeud contenant l'Ã©lÃ©ment Ã  supprimer
              y,        // Noeud Ã  supprimer
              z,        // Fils de y
              parent;   // Parent de y
        // Chercher x et son parent
        x = root; parent = null;
        while (x != null && x.number != num) {
            parent = x;
            if (x.number < num)
                x = x.right;
            else
                x = x.left;
        }
        if (x != null) {     // x est trouvÃ©
            if (x.left == null || x.right == null)  // x a un fils maximum
                y = x;
            else {                                   // x a deux fils
                // Chercher y le noeud contenant le minimum du SAD de x
                y = x.right;
                parent = x;
                while (y.left != null) {
                    parent = y;
                    y = y.left;
                }
                // Mettre la valeur de y dans x
                x.number = y.number;
                x.name = y.name;
                x.score = y.score;
            }
            // z est le seul fils de y
            if (y.left == null)
                z = y.right;
            else
                z = y.left;
            if (parent == null) // y = root
                root = z;       // L'arbre restant est le fils de y
            else {              // Supprimer y en ratachant son fils Ã  son parent
                if (y == parent.left)
                    parent.left = z;
                else
                    parent.right = z;
            }
        } else
            throw new NoSuchElementException();
    }
}
```