Arbre pour représenter un DOM

Compter les éléments

Version Kylian

```
public static int nbElement(BTree<String> btree, String element){
    int count = 0;

    if(btree.data().compareTo(element) == 0){
        count++;
    }

    if(btree.left() !=  null){
        count += nbElement(btree.left(), element);
    }

    if(btree.right() !=  null){
        count += nbElement(btree.right(), element);
    }

    return count;
}
```

Version Prof

La différence est que le prof utilise des try/catch pour gérer les *null*.

```
public static int countElement(Arbre<String> ab, String element){
    int count = 0;
    if (ab.getElement() == element){
        count += 1;
    }
    try{
        count += countElement(ab.getGauche(), element);
    }
    catch(NullPointerException e) {}

    try{
        count += countElement(ab.getDroite(), element);
    }
    catch(NullPointerException e) {}

    return count;
}
```

Compter les déscendant directes

Version Kylian

```
public static int nbDescendant(BTree<String> btree, String e1, String e2){
    int count = 0;
    if (btree.left() != null){
        if(btree.data().compareTo(e1) == 0 &&
btree.left().data().compareTo(e2) == 0){
            count ++;
        }
        count += nbDescendant(btree.left(), e1, e2);
    }
    if (btree.right() != null){
        if(btree.data().compareTo(e1) == 0 &&
btree.right().data().compareTo(e2) == 0){
            count ++;
        }
        count += nbDescendant(btree.right(), e1, e2);
    }

    return count;
}
```

Version Prof

Ici aussi, il gere les *null* avec des try/catch. Il sépare les blocs de "comptage" et les blocs de relance.

```
public static int countTwoElements(Arbre<String> ab, String e1, String e2)
{
    int count = 0;
    if (ab.getElement() == e1){
        try{
            if (ab.getGauche().getElement() == e2){
                count += 1;
            }
        }
        catch (NullPointerException e) {}

        try{
            if (ab.getDroite().getElement() == e2){
                count += 1;
            }
        }
        catch (NullPointerException e) {}
    }

    try{
        count += countTwoElements(ab.getGauche(), e1, e2);
    }
```

```
        catch(NullPointerException e) {}

        try{
            count += countTwoElements(ab.getDroite(), e1, e2);
        }
        catch(NullPointerException e) {}

        return count;
    }
```