| | |
|---|---|
| **Commencé le** | lundi 6 mars 2023, 14:46 |
| **État** | Terminé |
| **Terminé le** | lundi 6 mars 2023, 15:02 |
| **Temps mis** | 15 min 35 s |
| **Note** | **10,00** sur 10,00 (**100**%) |

**Question 1**

Correct

Note de 4,00 sur 4,00

Implémenter dans la classe public class *BinaryTree*, la méthode *contains* qui prend en paramètre l'objet recherché, o.

Elle renvoie vrai si un objet contenu dans l'arbre est égal à 0, faux sinon.

---

Implement in the public class *BinaryTree*, the "*contains*" method that takes as a parameter the searched object, o.

It returns true if an object contained in the tree is equal to o and false otherwise.

**Réponse :** (régime de pénalités : 0 %)

Réinitialiser la réponse

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
package ads.poo2.exams.test1;

/**
 * A class for simple binary nodes
 */

public class BinaryTree<T> {

        private  T data;
        private BinaryTree<T> leftBT;
        private BinaryTree<T> rightBT;


        ///////////////// constructors
        /**
         * Build a binary node which is
         * a leaf holding the value 'getData'
         */
```

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✔ | `BinaryTree<Integer> bt = new BinaryTree<>(1);`<br>`assertTrue(bt.contains(1));`<br>`assertFalse(bt.contains(2));` | true<br>false | true<br>false | ✔ |
| ✔ | `BinaryTree<Integer> leaf = new BinaryTree<>(100);`<br>`BinaryTree<Integer> bt = new BinaryTree<>(1, leaf, null);`<br>`assertTrue(bt.contains(1));`<br>`assertTrue(bt.contains(100));`<br>`assertFalse(bt.contains(2));` | true<br>true<br>false | true<br>true<br>false | ✔ |
| ✔ | `BinaryTree<Integer> leaf = new BinaryTree<>(100);`<br>`BinaryTree<Integer> bt = new BinaryTree<>(1, new BinaryTree<>(2), leaf);`<br>`assertTrue(bt.contains(1));`<br>`assertTrue(bt.contains(2));`<br>`assertTrue(bt.contains(100));`<br>`assertFalse(bt.contains(3));` | true<br>true<br>true<br>false | true<br>true<br>true<br>false | ✔ |
| ✔ | `BinaryTree<Integer> leaf = new BinaryTree<>(100);`<br>`BinaryTree<Integer> bt = new BinaryTree<>(1, new BinaryTree<>(2), leaf);`<br>`leaf.setLeftBT(new BinaryTree<>(200, new BinaryTree<>(300), null));`<br>`assertTrue(bt.contains(100));`<br>`assertTrue(bt.contains(200));`<br>`assertTrue(bt.contains(300));`<br>`assertFalse(bt.contains(-10));` | true<br>true<br>true<br>false | true<br>true<br>true<br>false | ✔ |

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✔ | `BinaryTree<Integer> bt = oneBigFull(1,10);`<br>`assertTrue(bt.contains(1));`<br>`assertTrue(bt.contains(100));`<br>`assertTrue(bt.contains(2));`<br>`assertTrue(bt.contains(1000));`<br>`assertFalse(bt.contains(-100));` | `true`<br>`true`<br>`true`<br>`true`<br>`false` | `true`<br>`true`<br>`true`<br>`true`<br>`false` | ✔ |

Tous les tests ont été réussis ! ✔

▸ **Montrer / masquer la solution de l'auteur de la question (Java)**

Correct

Note pour cet envoi : 4,00/4,00.

**Question 2**

Correct

Note de 6,00 sur 6,00

Implémenter dans la classe public class *BinaryTree*, la méthode *isFull* qui renvoie vrai quand tous les noeuds ont 0 ou 2 fils

---

Implement in the public class *BinaryTree*, the "*isFull*" method that returns true, if a Tree is full, i.e., if all the nodes have 0 or 2 children

**Réponse :** (régime de pénalités : 0 %)

Réinitialiser la réponse

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
                if (root.isLeaf())
                        return str ;
                String newHeight = height + makeString(' ',branch.length()+1 )+"|";
                String newBranch = makeString('_',addedChar);
                str+= traversePreOrderToString(root.left(),newHeight,newBranch);
                str += newHeight + "\n";
                //pour le 2e fils, on ne veut plus descendre la branche du parent
                newHeight = height + makeString(' ',branch.length()+1 );
                str += traversePreOrderToString(root.right(),newHeight,"|"+newBranch);
                return str;
        }
        protected String makeString(char c, int k) {
                return  String.valueOf(c).repeat(k);
        }


}
```

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✔ | `BinaryTree<Integer> bt = new BinaryTree<>(1);`<br>`assertTrue(bt.isFull());` | true | true | ✔ |
| ✔ | `BinaryTree<Integer> leaf = new BinaryTree<>(100);`<br>`BinaryTree<Integer> bt = new BinaryTree<>(1, leaf, null);`<br>`assertFalse(bt.isFull());` | false | false | ✔ |
| ✔ | `BinaryTree<Integer> leaf = new BinaryTree<>(100);`<br>`BinaryTree<Integer> bt = new BinaryTree<>(1, new BinaryTree<>(2), leaf);`<br>`assertTrue(bt.isFull());` | true | true | ✔ |
| ✔ | `BinaryTree<Integer> leaf = new BinaryTree<>(100);`<br>`BinaryTree<Integer> bt = new BinaryTree<>(1, new BinaryTree<>(2), leaf);`<br>`leaf.setLeftBT(new BinaryTree<>(200, new BinaryTree<>(300), null));`<br><br>`assertFalse(bt.isFull());` | false | false | ✔ |
| ✔ | `BinaryTree<Integer> bt = oneBigFull(1,10);`<br>`assertTrue(bt.isFull());` | true | true | ✔ |
| ✔ | `BinaryTree<Integer> leaf = new BinaryTree<>(100);`<br>`BinaryTree<Integer> bt = new BinaryTree<>(1, new BinaryTree<>(2), leaf);`<br>`leaf.setLeftBT(new BinaryTree<>(200, new BinaryTree<>(300), null));`<br>`assertFalse(bt.isFull())` | false | false | ✔ |
| ✔ | `BinaryTree<Integer> bt = new BinaryTree<>(1);`<br>`for (int i = 2; i < 100; i++) {`<br>`bt = new BinaryTree<>(i, bt, null);`<br>`};`<br>`assertFalse(bt.isFull());` | false | false | ✔ |

Tous les tests ont été réussis ! ✔

▸ **Montrer / masquer la solution de l'auteur de la question (Java)**

Correct

Note pour cet envoi : 6,00/6,00.