Commencé le	lundi 13 mars 2023, 14:49
État	Terminé
Terminé le	lundi 13 mars 2023, 15:04
Temps mis	14 min 37 s
Note	**8,50** sur 10,00 (**85**%)

Question 1

Correct

Note de 7,00 sur 7,00

Implémenter dans la classe public class *BinaryTree*, la méthode *isPerfect* qui renvoie vrai si tous les noeuds ont 0 ou 2 fils et que toutes les feuilles sont au même niveau.

Implement in the public class *BinaryTree*, the "*isPerfect*" method that returns true, if a Tree is perfect, i.e., if all the nodes have 0 or 2 children and all the leaves are at the same level.

Par exemple:

Test	Résultat

`//One node`
`BinaryTree<Integer> bt = new BinaryTree<>(1);`
`assertTrue(bt.isPerfect());`
	`true`

`//Three nodes`
`BinaryTree<Integer> leaf = new BinaryTree<>(100);`
`BinaryTree<Integer> bt = new BinaryTree<>(1, new BinaryTree<>(2), leaf);`
`assertTrue(bt.isPerfect());`
	`true`

`//Tree full but not perfect`
`BinaryTree<Integer> leaf = new BinaryTree<>(100);`
`BinaryTree<Integer> bt = new BinaryTree<>(1, new BinaryTree<>(2), leaf);`
`leaf.setLeftBT(new BinaryTree<>(200, new BinaryTree<>(300), new BinaryTree<>(400)));`
`leaf.setRightBT(new BinaryTree<>(500));`
`assertTrue(bt.isFull());`
`assertFalse(bt.isPerfect());`
	`true`
	`false`

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1  package ads.poo2.exams.test1;
2
3  /**
4   * A class for simple binary nodes
5   */
6
7  public class BinaryTree<T> {
8
9      private  T data;
10     private BinaryTree<T> leftBT;
11     private BinaryTree<T> rightBT;
12
13
14     ///////////////// constructors
15     /**
16      * Build a binary node which is
17      * a leaf holding the value 'getData'
18      */
19     public BinaryTree(T data) {
20         this(data,null,null);
21     }
22
23     /**
24      * Build a binary node holding the value 'getData' with
25      * 'leftBT' as the leftBT subtree and 'rightBT' as the rightBT subtree
26      */
27     public BinaryTree(T data, BinaryTree<T> left, BinaryTree<T> right) {
28         this.data = data;
29         this.leftBT = left;
30         this.rightBT = right;
31     }
32
33     ///////////////// accessors
34
35     public T getData() {
36         return data;
37     }
38
39     public void setData(T data) {
40         this.data = data;
41     }
42
43     public BinaryTree<T> left() {
44         return leftBT;
45     }
```

```
46
47      public BinaryTree<T> right() {
48          return rightBT;
49      }
50
51      public void setLeftBT(BinaryTree<T> node) {
52          leftBT = node;
```

	Test	Résultat attendu	Résultat obtenu	

✔	`//One node`
	`BinaryTree<Integer> bt = new BinaryTree<>(1);`
	`assertTrue(bt.isPerfect());`
		`true`	`true`	✔

✔	`//Two nodes`
	`BinaryTree<Integer> leaf = new BinaryTree<>(100);`
	`BinaryTree<Integer> bt = new BinaryTree<>(1, leaf, null);`
	`assertFalse(bt.isPerfect());`
		`false`	`false`	✔

✔	`//Three nodes`
	`BinaryTree<Integer> leaf = new BinaryTree<>(100);`
	`BinaryTree<Integer> bt = new BinaryTree<>(1, new BinaryTree<>(2), leaf);`
	`assertTrue(bt.isPerfect());`
		`true`	`true`	✔

✔	`//Tree full but not perfect`
	`BinaryTree<Integer> leaf = new BinaryTree<>(100);`
	`BinaryTree<Integer> bt = new BinaryTree<>(1, new BinaryTree<>(2), leaf);`
	`leaf.setLeftBT(new BinaryTree<>(200, new BinaryTree<>(300), new BinaryTree<>(400)));`
	`leaf.setRightBT(new BinaryTree<>(500));`
	`assertTrue(bt.isFull());`
	`assertFalse(bt.isPerfect());`
		`true`
		`false`
			`true`
			`false`
				✔

✔	`BinaryTree<Integer> bt = oneBigFull(1,10);` `assertTrue(bt.isPerfect());`	`true`	`true`	✔

✔	`//Tree full but not Complet more complex`
	`BinaryTree<Integer> leaf = new BinaryTree<>(100);`
	`BinaryTree<Integer> bt = new BinaryTree<>(1, new BinaryTree<>(2), leaf);`
	`leaf.setLeftBT(new BinaryTree<>(200, new BinaryTree<>(300), new BinaryTree<>(400, new BinaryTree<>(401), new BinaryTree<>(402))) );`
	`leaf.setLeftBT(new BinaryTree<>(200, new BinaryTree<>(300), new BinaryTree<>(400, new BinaryTree<>(401),  new BinaryTree<>(402))) );`
	`leaf.setRightBT(new BinaryTree<>(500, new BinaryTree<>(600), new BinaryTree<>(700)));`
	`assertTrue(bt.isFull());`
	`assertFalse(bt.isPerfect());`
		`true`
		`false`
			`true`
			`false`
				✔

✔	`BinaryTree<Integer> bt = new BinaryTree<>(1);`
	`for (int i = 2; i < 100; i++) {`
	`  bt = new BinaryTree<>(i, bt, null);`
	`  };`
	`assertFalse(bt.isPerfect());`
		`false`	`false`	✔

✔	`BinaryTree<String> bt = new BinaryTree<>("A", new BinaryTree<>("B", new BinaryTree<>("B1"), new BinaryTree<>("B2")),`
	`  new BinaryTree<>("C", new BinaryTree<>("C1"), new BinaryTree<>("C2")));`
	`assertTrue(bt.isPerfect());`
		`true`	`true`	✔

✔	`//isPerfect with Three Nodes containing strings`
	`BinaryTree<String> bt = new BinaryTree<>("A", new BinaryTree<>("B"), new BinaryTree<>("C"));`
	`assertTrue(bt.isPerfect());`
		`true`	`true`	✔

Tous les tests ont été réussis ! ✔

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

Correct

Note pour cet envoi : 7,00/7,00.

Question 2

Incorrect

Note de 0,00 sur 1,50

Soit un BST ***T*** et 2 éléments **distincts** x et y qui ne sont pas dans ***T***.

On note :

- $T_{x,y}$ le BST obtenu par insertion des 2 éléments dans l'ordre x puis y
- $T_{y,x}$ le BST obtenu par insertion des 2 éléments dans l'ordre y puis x.

Let be a BST **T** and 2 **distinct** elements x and y which are not in **T**.

We note :

- $T_{x,y}$ the BST obtained by inserting the 2 elements in the order x then y
- $T_{y,x}$ the BST obtained by inserting the 2 elements in the order y then x.

Veuillez choisir au moins une réponse.

- $T_{x,y}$ est toujours identique à $T_{y,x}$

  $T_{x,y}$ is always identical to $T_{y,x}$

- $T_{x,y}$ est parfois identique à $T_{y,x}$ et $T_{x,y}$ est parfois différent de $T_{y,x}$

  $T_{x,y}$ is sometimes identical to $T_{y,x}$ et $T_{x,y}$ is sometimes different from $T_{y,x}$

- $T_{x,y}$ est toujours différent de $T_{y,x}$ ✖ Soit un BST qui contient le noeud 3 si on ajoute 2 puis 4 ou 4 puis 2 le résultat est toujours le même

  $T_{x,y}$ is always different from $T_{y,x}$

Votre réponse est incorrecte.

La réponse correcte est :
$T_{x,y}$ est parfois identique à $T_{y,x}$ et $T_{x,y}$ est parfois différent de $T_{y,x}$

$T_{x,y}$ is sometimes identical to $T_{y,x}$ et $T_{x,y}$ is sometimes different from $T_{y,x}$

Question 3

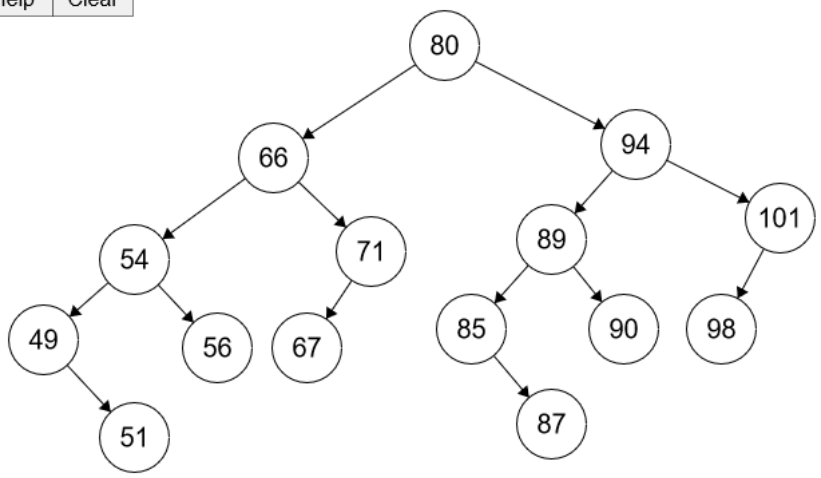
Correct

Note de 1,50 sur 1,50

Calculer la **somme** du minimum et du maximum du BST ci-dessous :

Calculate the sum of the minimum and maximum of BST below:

Help Clear



Réponse : 150

La réponse correcte est : 150