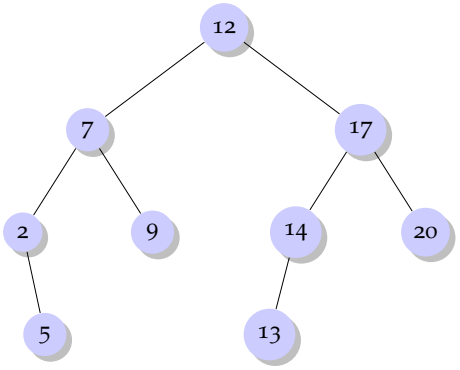
Tests sur ABR

⌨💬**Exercice 2 :** On veut implémenter un *arbre binaire de recherche*. C'est un petit défi algorithmique, cependant, le focus du TD est bien de réaliser une classe pour faire des tests avec `JUnit`. L'arbre ne doit pas forcément être équilibré.

Un arbre binaire est constitué d'une valeur et d'au plus deux sous arbres (le gauche et le droit), qui sont eux aussi considérés comme des arbres binaires. Pour l'exercice, on va supposer que toutes les valeurs sont des **int** (on verra plus tard dans le cours comment rendre cette structure plus utile).

L'arbre est dit de recherche car il satisfait la propriété suivante : si une valeur y se trouve dans le sous arbre gauche d'un nœud qui a pour valeur x, alors $y \leq x$. De plus, on va ajouter la contrainte qu'une valeur ne se trouve que dans un seul nœud de l'arbre (i.e. on interdit les doublons).

Exemple



Si on insère la valeur 10, où doit-on placer le noeud ? Pour la valeur 16 ? Pour la valeur 19 ?

1. Ci-dessous, on va vous demander d'implémenter une classe pour utiliser un arbre binaire de recherche avec plusieurs méthodes. Lisez leur description ci-dessous et avant de les implémenter, préparez des tests dans une classe TestABR. Essayez de penser aux différents cas qui pourraient arriver. Ecrivez des tests pour toutes les fonctionalités.
2. Ecrivez une classes représentant un arbre binaire de recherche.
3. Ecrivez une méthode `inWalk()` de la classe représentant l'arbre qui va afficher les valeurs de l'arbre avec la propriété suivante : la valeur du noeud x dans l'affichage sera imprimée *entre* les valeurs des noeuds de son sous arbre à gauche et les valeurs de son sous arbre à droite. Par exemple, l'arbre de l'exemple devrait imprimer `2 5 7 9 12 13 14 17 20`.
4. Ecrivez une méthode `max()` dans la classe d'arbre binaire de recherche qui retourne la plus grande valeur présente dans l'arbre.
5. Ecrivez une méthode nommée `contains` qui prend en paramètre un **int** et qui retourne un `boolean` indiquant si la valeur se trouve ou non dans l'arbre.
   Si h est la hauteur de l'arbre, votre méthode ne devrait visiter qu'au plus h noeuds.
6. Dans la classe représentant un arbre, écrivez une méthode `add` qui prend en paramètre une valeur et qui ajoute un noeud contenant la valeur. On ne va pas permettre les doublons, et si une valeur se trouve déjà dans l'arbre, on ne va pas la mettre de nouveau. Evidemment, il va falloir placer judicieusement ce noeud.