|  |  |
| --- | --- |
| Ponctuel 13 | **Les arbres binaires (implantation)** |

1. Importez le projet existant *ExerciceArbre*.
   1. Examinez le programme principal contenu dans la classe *Go.java*. Celui-ci crée un arbre binaire de clients et l’affiche. Modifiez la classe *Client.java* de manière à faire fonctionner le programme principal sans erreur.
   2. A votre avis, est-ce que la version suivante de la méthode *compareTo* fonctionnerait ?

public int compareTo(Object objet) {

Client client = (Client) objet;

return(this.noClient - client.noClient);

}

* 1. Examinez les commentaires décrivant les différentes méthodes incluses dans la classe *ArbreBinaire*. Ajoutez à votre programme principal du code qui demande à l'utilisateur d'entrer le numéro d'un client et qui supprime ce client dans arbre. Affichez l'un des messages suivants*: Le client a été supprimé*  ou encore *Désolé, ce client n'est pas dans l'arbre*.
  2. Dans la classe *Client*, ajoutez un *setter* pour la propriété *nom*. Puis, dans votre programme principal, demandez à l’utilisateur d’entrer un numéro de client ainsi qu’un nom de client : modifiez ensuite (dans l’arbre binaire) le nom du client ayant ce numéro.

1. On vous demande d'ajouter les méthodes suivantes à la classe *ArbreBinaire* (continuez de travailler dans le projet du numéro précédent). Votre code doit être efficace et éviter les parcours multiples de l'arbre!
   1. Ajoutez et testez la méthode *public* *E* **trouverMax**( ) qui parcourt l'arbre et retourne la donnée de valeur maximale dans celui-ci.
   2. Ajoutez la méthode suivante:

public String toString ( ){

String chaine = **toStringRecursif** (racine);

return ("[" + chaine + "]");

}

Cette méthode forme et retourne une seule chaîne de caractères contenant toutes les données de l'arbre en ordre croissant. Elle fait appel à une méthode privée récursive *private String* **toStringRecursif***(Nœud<E>* courant*)*, que vous devez écrire. Inspirez-vous des méthodes *afficher* et *afficherRecur* déjà présentes dans votre classe (et au besoin, retournez-voir la méthode *nbElements* écrite au ponctuel 17).

* 1. Ajoutez et testez la méthode *public* *int*  **profondeur**( ) qui calcule et retourne la profondeur de l'arbre (nombre maximal de niveaux). À vous de déterminer si votre méthode sera récursive ou non.