Licence Informatique – 3ème Année

POO & Java

TP n°2 – Abstraction

---

1. Arbres

L'objectif de ce TP est de développer un programme permettant de gérer des **arbres** dont chaque **nœud** contient un objet. Cet objet peut être :
- un **entier** (Integer),
- une **chaîne de caractères** (String).

Nous voulons représenter des **arbres de tout type**, tout en garantissant que chaque arbre implémente les méthodes suivantes :

- **Ajouter** un objet comme nœud de l'arbre (l'endroit où on l'ajoute dépend du type d'arbre).
- **Supprimer** un objet de l'arbre.
- **Obtenir la hauteur** de l'arbre.
- **Vérifier la présence** d'un objet dans l'arbre.

  Note :
  La manière d'implémenter ces méthodes dépend du type d'arbre.
  Par exemple :
  - Dans un **arbre général**, la recherche d'un objet nécessite un **parcours exhaustif**.
  - Dans un **arbre binaire de recherche**, un **parcours dichotomique** suffit.

Étapes du TP

1.1 Arbres généraux

- Écrire du code objet pour représenter les **arbres d'objets en général**.
- Créer une **interface** commune que tous les arbres doivent implémenter.
- Les méthodes à définir dans l'interface :

```java
public void addNode(Object i);
public void removeNode(Object i);
public int getHeight();
public boolean exists(Object i);
```

1.2 Arbres binaires

- Écrire du code objet pour représenter les **arbres binaires**, où chaque nœud peut avoir **au plus 2 fils**.
- Implémenter la méthode pour calculer la **hauteur** d'un arbre binaire.
- Restriction :
  - On ne peut **supprimer** qu'un **nœud feuille** dans un arbre binaire.

1.3 Arbres binaires de recherche

- Écrire du code objet pour représenter les **arbres binaires de recherche**, en respectant les règles suivantes :
  - Le **fils gauche** d'un nœud contient une valeur **inférieure ou égale** à celle du nœud.
  - Le **fils droit** contient une valeur **strictement supérieure** à celle du nœud.
- Implémenter les fonctionnalités suivantes :
  - **Ajout** d'un objet comme nœud de l'arbre.
  - **Recherche** d'un objet dans l'arbre.
  - **Suppression** d'un objet de l'arbre, en suivant l'exemple du fichier de cours `arbres_2.pdf`.

Optionnels

1. **Arbres n-aires**
   - Écrire du code pour représenter les **arbres n-aires**, où chaque nœud peut avoir un **nombre quelconque de fils**.
   - Implémenter la méthode pour calculer la **hauteur** d'un arbre n-aire.
2. **Chargement d'un dictionnaire**
   - Adapter le programme pour charger le fichier `dictionnaire.fich` dans un arbre binaire de recherche.
   - Tester la **vitesse d'exécution** de la recherche d'un mot dans cet arbre.
3. **Explorations libres**
   - Ajouter toute autre fonctionnalité ou exploration qui enrichit vos compétences en **programmation orientée objet**.

Livrables

Le travail peut être réalisé en **binôme**. Chaque binôme devra soumettre les éléments suivants **au plus tard le vendredi 13 décembre 2024 à 23h59** :

1. Un **dossier** contenant :
    - Un **manuel utilisateur**.
    - Un **diagramme des classes**.
    - Un **bilan du projet**.
    - Les **listings** du code source.
    - La **documentation Javadoc** des classes.
2. Un fichier **ZIP** contenant :
    - Les **sources** (`.java`).
    - Les fichiers objets (`.class`) permettant de tester l'implémentation sans avoir besoin de compiler.