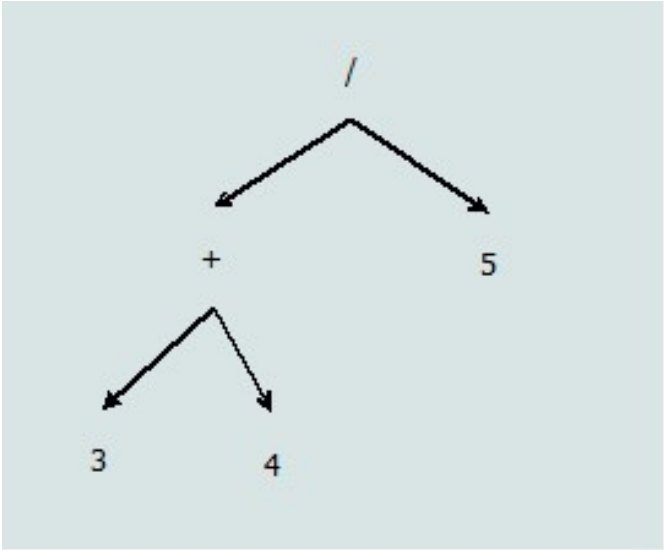
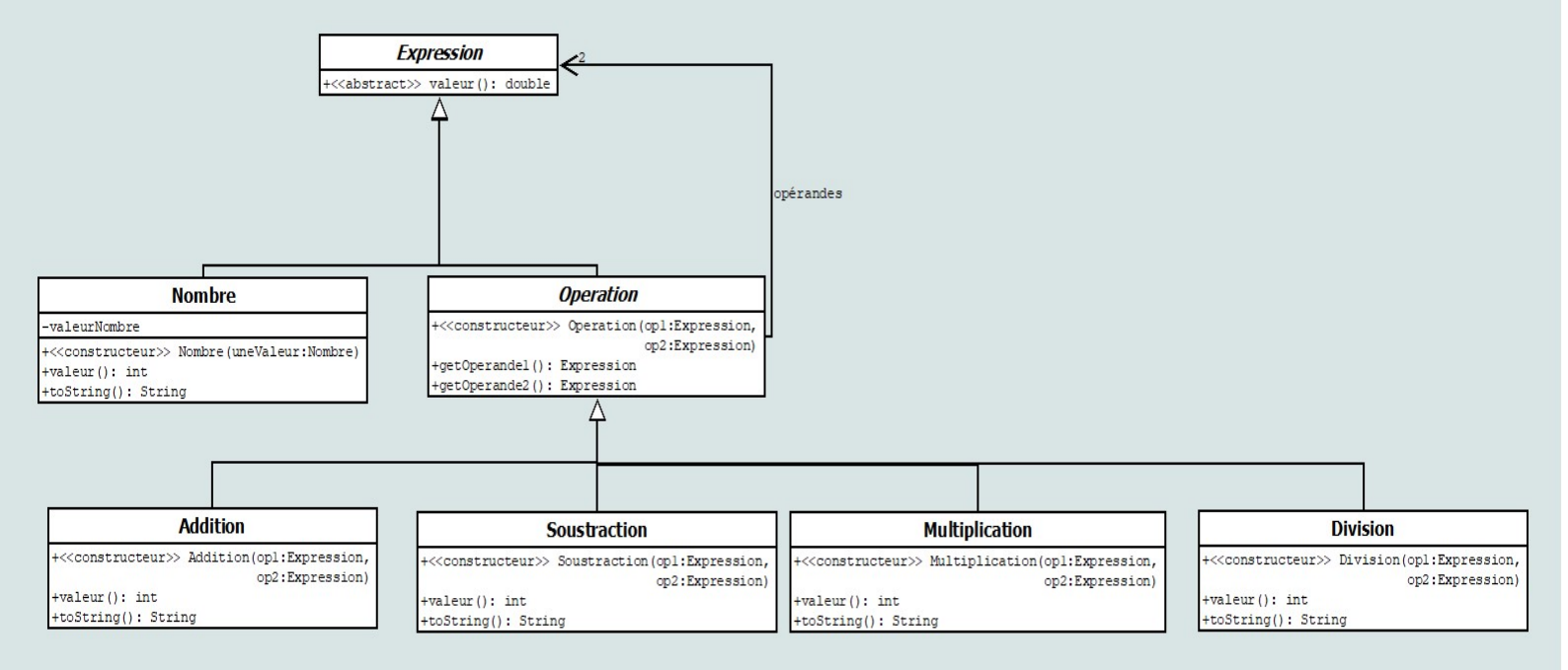
Partie II sensibilisation à la récursivité

Dans cette seconde partie nous considérons que la calculatrice peut effectuer des opérations composées. Une opération est composée lorsque qu'au moins l'une des deux opérandes est-elle même une opération. L'arbre ci-dessous représente l'opération *(3+4) / 5* où l'opérande de gauche est l'opération *3+4*.



III. UML (lecture d'un diagramme des classes)

Pour modéliser la possibilité qu'une opérande puisse être un **Nombre** ou une **Operation** nous introduisons la classe **Expression** dans le diagramme des classes suivant :



On observe que les **Nombres** et les **Operations** sont des **Expressions**. On constate aussi que les 2 opérandes d'une **Operation** sont de type **Expression** modélisant ainsi le fait qu'une opérande peut être un **Nombre** ou une **Operation**. Dans ce diagramme les classes Expression et **Operation** sont abstraites. Dans le programme java une **Expression** sera soit un **Nombre** soit une opération concrète c'est à dire une **Addition**, une **Soustraction**, une **Multiplication** ou une **Division**.

IV. JAVA

Coder ce diagramme des classes en Java et tester (dans une classe **Calculatrice**) différentes opérations composées. Nous donnons ci-dessous un exemple de test de l'opération (17-2) / (2 +3)

Expression deux = new Nombre(2) ;

Expression trois = new Nombre(3) ;

Expression dixSept = new Nombre(17) ;

Expression s = new Soustraction(dixSept, deux) ;

Expression a = new Addition(deux, trois) ;

Expression d = new Division(s, a) ;

System.out.println(d + " = " + d.valeur()) ; // affiche ((17 - 2) / (2 + 3)) = 3

NOTE TRÈS IMPORTANTE :

Une méthode peut s'appeler elle-même (on parle de méthode récursive). Ainsi la méthode

valeur() des opérations concrètes peut s'appeler elle-même (un appel pour chaque opérande, on parle d'appel récursif).

Exemple

```
public int valeur()
{
        return this.getOperande1().valeur() + this.getOperande2().valeur()
}
```