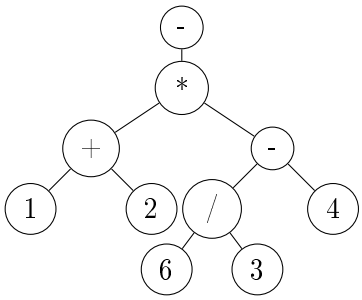
Exercice 1 : Polynôme. Dans cet exercice nous allons considérer des expressions arithmétiques modélisées par des arbres, comme par exemple :



Son écriture en étant exhaustif avec les parenthèses serait : $(-(((1+2)*((6/3)-4))))$.

1. Reprenez rapidement ce que vous avez déjà fait aux TP précédents dans un nouveau répertoire, et modifiez la variable `etiquette` pour qu'elle soit de type `String`.
2. Créez une méthode `private static boolean estNombre(String x)` qui renvoie `true` si et seulement si la chaîne x correspond à un nombre : c'est à dire que le premier symbole est un signe `+` ou `-` ou un chiffre entre `0` et `9`, et tous les autres symboles sont un chiffre entre `0` et `9` avec éventuellement la présence d'un seul point entre deux chiffres.

   Plus tard, après avoir vérifié qu'une chaîne x est bien un nombre vous pourrez récupérer sa valeur à l'aide de l'instruction : `Double.parseDouble(x)`.
3. Créez une méthode `public boolean verif()` qui renvoie `true` si et seulement si l'arbre correspond à une expression arithmétique correcte. C'est à dire que lorsque l'étiquette est un opérateur `+`, `/`, ou `*` alors l'arbre a deux enfants qui correspondent à une expression arithmétique correcte, et que pour l'opérateur `-` il y a au moins un enfant droit correct. Les autres noeuds sont des feuilles étiquetées par des chaînes représentant des nombres.
4. Testez la méthode `afficheInfixe()` d'un TP précédent sur l'arbre donné en exemple. Sur ce modèle, écrivez une méthode `public void afficheExpression()` qui ajoute des parenthèses.
5. Créez une méthode `public double eval()` qui évalue l'expression d'un arbre (supposé vérifié).
6. On accepte à présent que l'étiquette soit la lettre `x`. Nous travaillons donc sur des expressions avec une variable, une fonction de x. Modifiez donc la méthode `public boolean verif()` pour tenir compte de cet état.
7. Ecrivez une méthode `public double évaluer(double z)` qui calcule la valeur de la fonction pour la valeur `z`.
8. Créez une méthode `public Arbre simplifier()` qui retourne un arbre mathématiquement équivalent, mais légèrement simplifié, par exemple, vous pouvez remplacer `2+3` par `5` et d'autres choses simples de ce genre.
9. Créez une méthode `public Arbre deriver()` qui retourne un arbre correspondant à la fonction dérivée de `this`.

Pour rappel :

Fonction f	Dérivée f'
constante	0
`x`	1
$-c$	$-c'$
$c+d$	$c'+d'$
$c \times d$	$c' \times d + c \times d'$
$\frac{f}{g}$	$\frac{f' \times g - f \times g'}{g^2}$