TD2:Fractions

Dans cet exercice, on tâchera de mener une approche « compilation séparée ». Au fur et à mesure de l'exercice, on pourra compléter une fonction principale qui utilise les éléments créés.

Question 1- Après avoir réfléchi aux attributs qui caractérisent une fraction, définir en C++ une classe `fraction`. Englober cette classe dans un espace de noms « `math` ». Représenter cette classe en UML. Cette représentation sera complétée au fur et à mesure des exercices.

Question 2- Déclarer et définir les accesseurs en visualisation et en édition de la classe. Faire attention à la validité des valeurs stockées dans les attributs.

Question 3- Déclarer et définir un ou plusieurs constructeurs pour cette classe. Faire attention à la validité des valeurs stockées dans les attributs.

Question 4- Définir une méthode privée qui permet de simplifier une fraction. Utiliser cette fonctions pour améliorer le(s) constructeur(s) et/ou d'autres méthodes quand cela vous paraît utile.

Question 5- Ecrire la méthode `somme` qui permet de faire une addition de 2 fractions. Réfléchir au type de retour de cette fonction.

Question 6- Surcharger les opérateurs binaires `+`, `-`, `/`, `*` de façon à pouvoir effectuer des opérations de somme, différence, multiplication et division entre objets de type `fraction`. Après avoir étudié les conversions automatiques d'entier en fraction, surcharger, si besoin, ces opérateurs de manière à rendre possible ces opérations entre entiers et fractions.

Question 7- Définir les opérateurs de comparaison de fractions (`==`, `!=`). Ces fonctions devront renvoyer un booléen.

Question 8- Surcharger l'opérateur `++` en postfixe et en prefixe.

Question 9- Définir une fonction `print` qui permet d'afficher une fraction sur un objet `ostream` donné (passé en argument par référence).

Question 10- Surcharger l'opérateur `<<` qui pourra permettre d'afficher une fraction en utilisant un flux `ostream` comme `cout`. Faire attention au type de retour de la fonction.