TP1 *Création de notre première interface en Qt5*

Le but de ce TP est de réaliser une petite application simple permettant d'effectuer une opération arithmétique entre deux nombres et d'afficher le résultat. Les opérations à gérer sont : '+', '-', '*', '/'. Vous devez obtenir une interface similaire à celle ci-dessous :

Exercice 1 : Découverte de l'environnement Qt et compilation avec Qt5

Dans un premier temps, vous devrez identifier les différents composants qui ont servis à réaliser cette interface puis les placer pour obtenir le même comportement que celui qui apparaît sur le dessin ci-dessus. Enfin il faudra créer et connecter les divers `slots` et `signaux Qt` nécessaires au bon fonctionnement de l'application.

Pour réaliser cet exercice, vous devez utiliser les composants `Qt` à votre disposition. Vous devez exécuter toutes les phases de compilation manuellement. Voici quelques éléments qui vous permettront de compiler une application `Qt5` :

- Pour compiler avec `g++` un code source écrit en Qt5, vous devez inclure (options `-I`) les entêtes (headers) de qt5. Ces entêtes se trouvent dans le répertoire `/usr/include/qt5`. Aussi, vous devez utiliser l'option de compilation (Position Independant Code) `-fPIC` ainsi que l'option `--std=c++11` si vous utilisez des fonctions lambda.
- Il est parfois nécessaire de lier (options -l des bibliothèques partagées avec votre code source (phase de linkage). Pour déterminer quelles sont les bibliothèques à lier, je vous invite à prendre connaissance du programme pkg-config (options `--list-all` associé à `grep` et `--libs`).
- Il est nécessaire de générer un fichier `.cpp` supplémentaire pour toute classe (déclarée dans un fichier `.h`) qui définit sa propre section `signals` et/ou `slots`. Par exemple : `moc -o moc_calculator.cpp calculator.h` génère le fichier `moc_calculator.cpp` à partir du fichier `calculator.h` qui contient une classe définissant sa propre section `slots` et sa propre section `signals`. Il ne faut pas oublier de compiler ce fichier et de le lier avec tous les autres fichiers issus des phases de compilation.

Exercice 2 : Séparation de l'interface du moteur

C'est une bonne pratique de dissocier le code concernant l'interface graphique de celui du "moteur" (i.e. : la partie algorithmique). En effet, en cas de changement de librairie d'interface graphique, vous n'avez pas à changer le code source du moteur. De plus, sur une application conséquente, cette dissociation peut faciliter le travail collaboratif : une personne se charge de l'implémentation du moteur pendant qu'une autre se charge de l'implémentation de l'interface.

Vous allez maintenant refaire l'exercice 1 en pensant à dissocier le code concernant l'interface graphique de celui du "moteur". Pour cela, vous aurez besoin d'une classe qui fera la liaison (la traduction) entre l'interface graphique et le moteur. Vous aurez donc au final 3 classes :

- Vue : affiche les composant graphique et "communique" avec Traducteur.
- Traducteur : fait la liaison entre Vue et Moteur.
- Moteur : effectue les calculs.

La classe Moteur **ne doit pas faire appel à la librairie Qt**, elle doit être implémentée uniquement avec du C++ standard.