# INF1010 Programmation Orientée-Objet

## Travail pratique #6 : Gestion des exceptions, Interface graphique

| | |
|---|---|
| **Objectifs :** | Permettre à l'étudiant de se familiariser avec les exceptions ainsi que les notions d'interface graphique |
| **Remise du travail :** | 19 avril 2017 à 08:00 |
| **Références :** | Notes de cours sur Moodle & Chapitre 17 du livre Big C++ |
| **Documents à remettre :** | Les fichiers .cpp et .h complétés réunis sous la forme d'une archive au format .zip |
| **Directives :** | Directives de remise des Travaux pratiques sur Moodle |
| | Les en-têtes et les commentaires sont obligatoires. |
| | Les travaux dirigés s'effectuent obligatoirement en équipe de 2personnes faisant partie du même groupe. |
| | Veuillez suivre le guide codage |

## Mise en contexte

Dans ce travail, vous êtes amenez à implémenter une calculatrice minimaliste avec interface graphique qui réalise les opérations suivantes : addition, soustraction, multiplication, division. Elle permet aussi de calculer le carré d'un nombre donné, son inverse et aussi sa racine carrée. Par contre, vous êtes demandé à gérer les cas d'exception qui se présenteront. Par exemple la division par zéro, l'inverse de zéro, la racine carrée d'un nombre négatif, etc… sont des cas d'exception à traiter. Lorsqu'il y'a une exception, un boite de dialogue critique est affiché indiquant la violation en cours.

## Travail à faire

Les fichiers « Calculatrice.h » et « Calculatrice.cpp », « Bouton.h » et « Bouton.cpp » et « Erreurs.h » vous sont remis, vous devrez compléter le fichier **Calculatrice.cpp** et **Erreur.h** , de façon à ce que votre application fonctionne correctement.

## Classe Calculatrice

Votre tâche principale est de rendre cette classe fonctionnelle de la façon présentée sur les imprimes écran ci-dessous.

Les tâches particulières sont les suivantes :

- Commenter cette classe c'est-à-dire :
  - Commenter tous les slots implémentés dans cette classe;
  - Commenter toutes les méthodes implémentées dans cette classe;
  - Commenter tous les widgets déclarés et utilisés dans cette classe.

En ce qui concerne les commentaires, prenez en note qu'un slot n'est rien d'autre qu'une méthode, donc les commentaires doivent respecter le format de commentaire d'une méthode. Par exemple :

```
/*****************************************************************************
        * Description     : Réinitialise toutes les valeurs et toutes les opérations
        * Paramètres      : Aucun
        * Type de retour   : void
 *****************************************************************************/

void Calculatrice::clearAll()
{
   sumSoF
   ar = 0.0;
   factorSo
   Far =
   0.0;
   pendingAdditiveOperator.cle
   ar();
   pendingMultiplicativeOperat
   or.clear(); display-
   >setText("0");
   waitingForOperand = true;
}
```

Pour ce qui est des widgets, vous devrez fouillez la documentation de Qt pour comprendre le fonctionnement du widget en question et commenter les instructions. Par exemple :

```
//création d'un objet de type QLineEdit avec par défaut la chaine de caractère 0
//cet objet représente une zone de texte qui sera utilisé comme afficheur de la
        //calculatrice display = new QLineEdit("0");
// la zone de texte est paramétrée en
        lecture seule display-
        >setReadOnly(true);
// l'affichage se fera de la droite vers la
        gauche display-
        >setAlignment(Qt::AlignRight);
// définition de la largeur de la zone de
        texte à 15 display-
        >setMaxLength(15);
```

Compléter l'implémentation de Calculatrice.cpp, c'est-à dire :

- Création des boutons manquant :

  - clearMemoryButton = > "MC".
  - readMemoryButton => "MR"
  - setMemoryButton => "MS"
  - addToMemoryButton => «M+ «

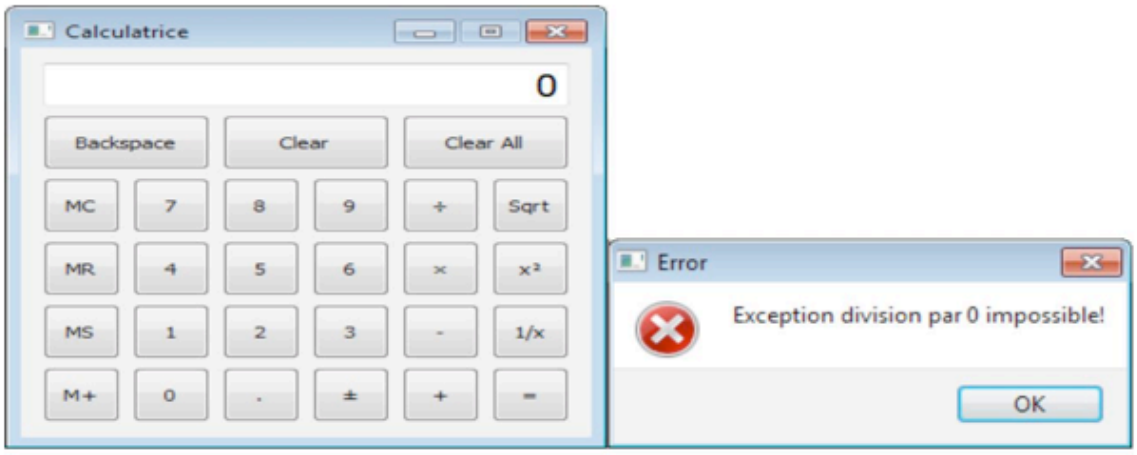
En suivant la logiques des boutons déjà crée et en y appliquant les slots appropriés (ces slots sont déjà implémentés vous n'avez qu'à les retrouver dans le fichier Calculatrice.cpp)

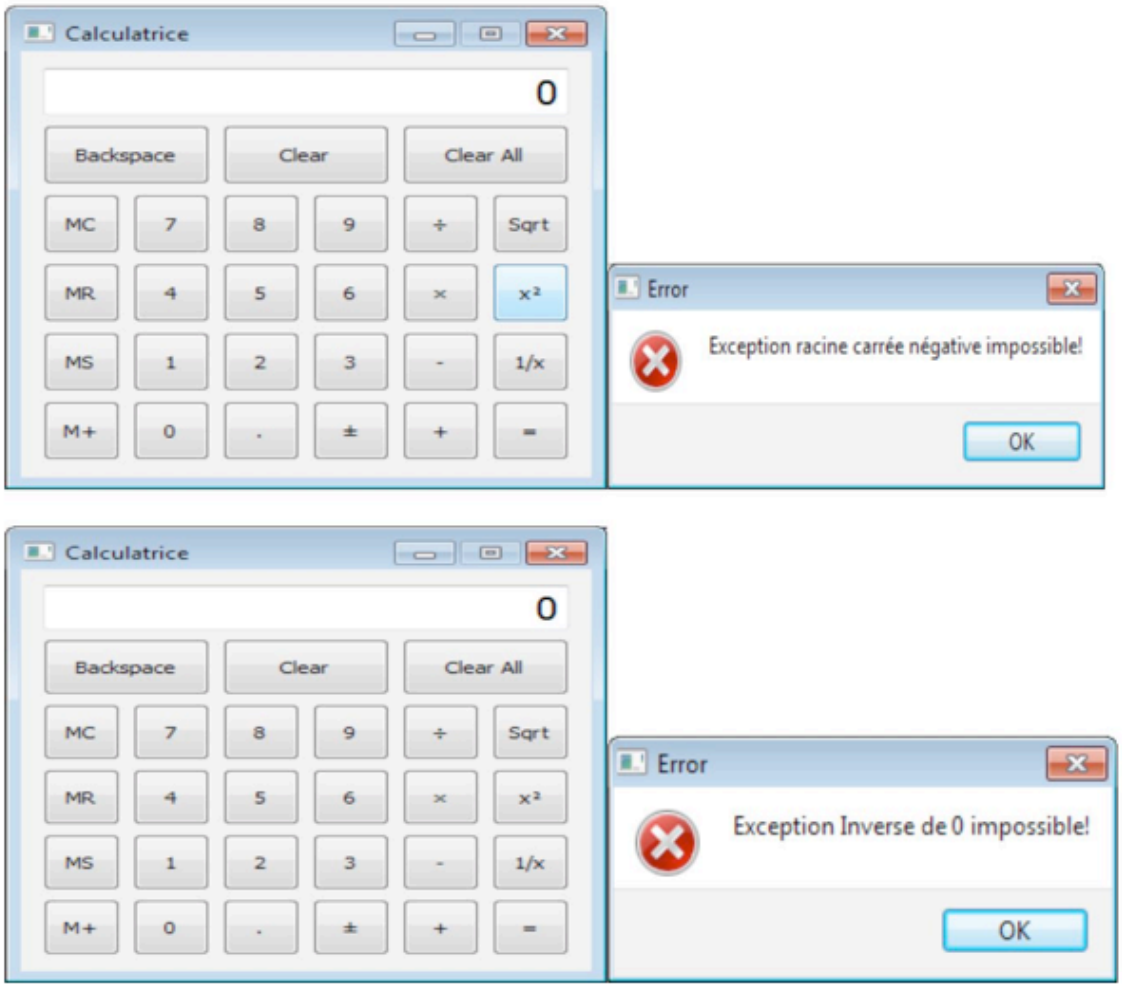
- Positionner tous les boutons sur le « **layout** » de façon à avoir la même disposition que sur l'imprime écran ci-dessous. Le choix du « **layout** » approprié pour cet affichage est laissé a votre choix (Vous devez au préalable Déclarer votre **layout**, vous vous en doutez bien). N'oubliez pas d'ajouter le titre à la fenêtre (« Calculatrice »)

## Les exceptions

Le fichier Erreur.h qui vous est fourni doit être complété avec les définitions et les implémentations de toutes les classes représentant les différentes exceptions à lever. Vous devez réalisez vos exceptions de façon à ce que chaque exception montre un message d'erreur, dans une boite à message, et qui s'affiche suivant imprimes écrans ci-dessous. Repérez toutes méthodes ou les slots dans le code susceptibles de créer une exception ajoutez-y le trio {try, throw et catch} pour gérer l'exception.

Étant donné que vous n'avez pas encore vu les interfaces graphiques en cours, pour cette semaine, vous pourrez gérer l'affichage des messages d'erreur avec des « string » comme en programmation console, et la semaine prochaine lorsque vous aurez vu les interfaces graphiques, vous transformerez vos messages en boîtes de message comme sur la figure ci-dessous.






## Main.cpp

Le main est fourni.

## Pour partir du bon pied

- **Documentation**

La documentation officielle de Qt est votre meilleur ami, vous pouvez la retrouvez à l'adresse :

http://qt-project.org/doc/

Tapez simplement le nom de votre Widget, slot ou signal… dans la zone de recherche « Search doc » et le site vous renvoie directement vers le lien menant à la description de votre recherche.

- **Environnement de programmation**

L'environement de programmation pour ce TP se fait sur Qt Creator et non sur MSVC.

Pour lancer votre environnement de programmation Qt Creator, s'il n'apparait pas dans le menu Démarrer->tous les programmes.

Allez sur le bureau, vous trouverez le fichier nommé Logiciels, dans ce fichier chercher Qt ou QtSDK, entrez y et cliquez sur le raccourcis de Qt qui s'y trouve.

**Création d'un projet Qt :**

Avec QtCreator :

Dans un premier temps : Cliquez sur l'onglet « File », ensuite « New File or Project », Choisissez l'option « Other Project » dans l'onglet « Project ».
Choisissez dans la fenêtre de droite « Empty Qt Project » et cliquez sur « choose ».
Entrez le nom de votre projet en faisant « next » et en laissant les paramètres par défaut fournis.
Une fois que votre projet est crée, cliquez une seconde fois sur « File », « New File or Project », cette fois ci choisissez « C++ » dans l'onglet « File and Classe » et créer le type de fichier que vous voulez : entete(.h) ou source(.cpp).
À partir de ce moment là tout ce passe comme sous Visual Studio.

**Ajout des fichiers déjà existant :**

Vous pouvez directement importer les fichiers avec Add Existing Files.

Une fois votre projet crée avec la procédure énoncé ci-haut, vous faites un clic droit sur le nom de votre projet,

Vous choisissez l'option add existing file et vous pouvez importer directement vos fichier à partir de l'emplacement ou vous les avez sauvegardé. Là aussi les choses se passent comme avec Visual Studio

Pour compiler votre code :

Cliquez sur Debug ensuite sur Run ou encore utilisez simplement le raccourci clavier Ctrl + R.

## Correction

La correction du tp1 sera sur 20 points. Voici les détails de la correction :

- (4 points) Compilation & exécution exactes des différentes méthodes
- (08 points) Documentation du code source et position correcte des boutons
- (08 points) Les exceptions sont levées correctement