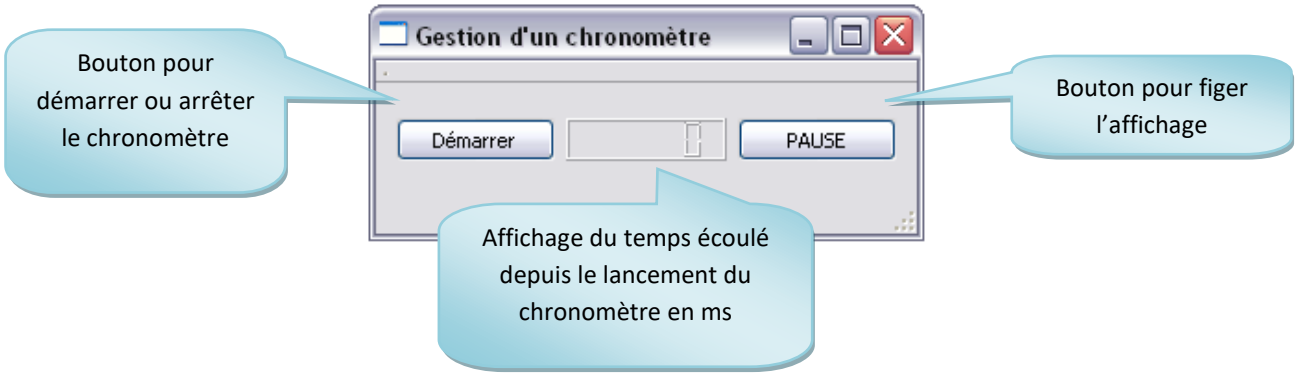
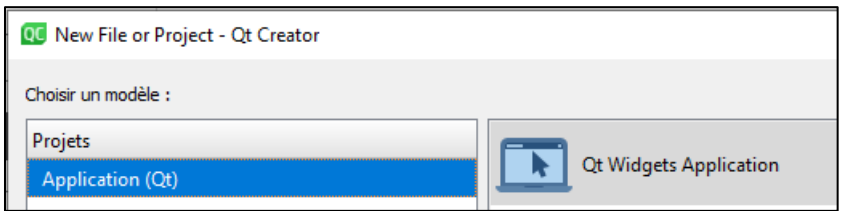
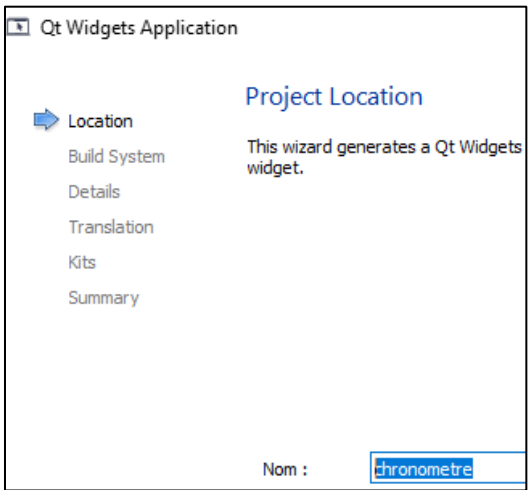
Mise en œuvre d'un chronomètre avec Qt

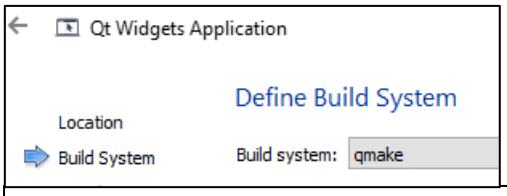
L'application que je vous propose d'écrire permet de gérer un chronomètre de façon graphique. Cette application utilise la classe CChronometre développée et testée en mode console

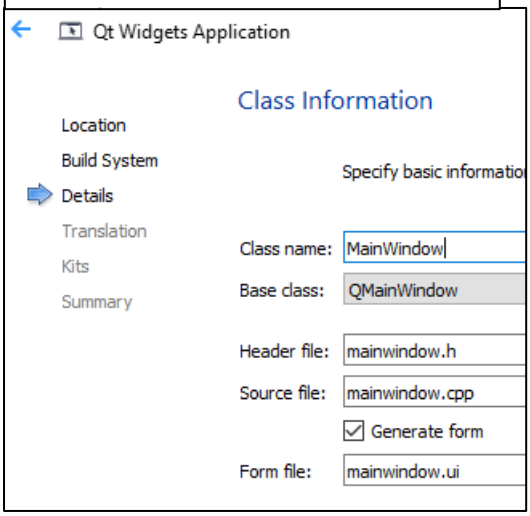


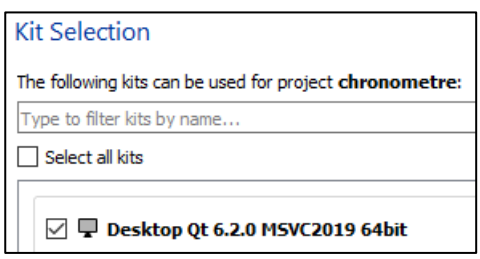
- Lancer QtCreator puis choisir créer un nouveau projet
- Sélectionnez ensuite la même chose que les copies d'écran ci-dessous











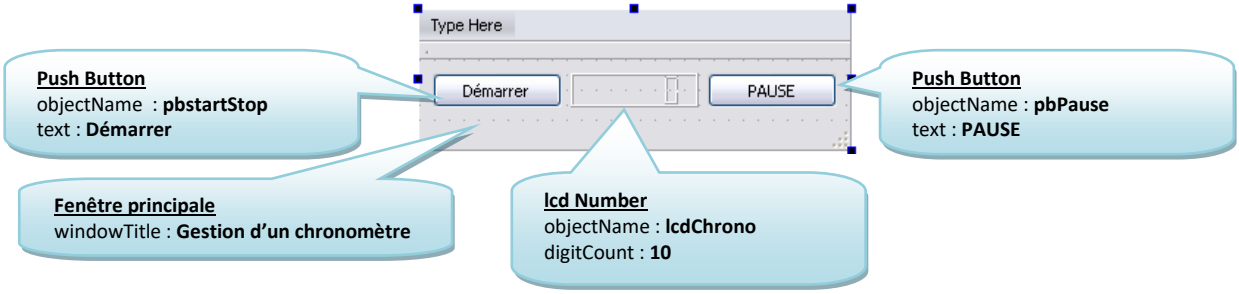
Validez les dernières options proposées par défaut.

A ce stade, QtCreator a créé :

- Une classe nommée « MainWindow » qui correspond à l'application principale
- Une interface graphique associée au fichier « mainwindow.ui »

1 Dessin de l'IHM

En faisant un double clic sur le fichier « mainWindow.ui », Qt Designer s'ouvre et présente un ensemble d'outils pour pouvoir créer l'IHM. Nous allons y ajouter nos éléments graphiques.



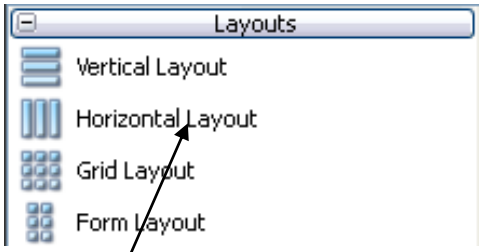
2 Organisation de l'IHM

Il ne suffit pas de placer les éléments graphiques sur l'écran pour que l'IHM soit terminée. Il faut aussi veiller à plusieurs choses :

- Vérifier le focus des éléments et l'ordre dans lequel la tabulation les sélectionne.
- Vérifier que les éléments ont des dimensions homogènes et que le redimensionnement de la fenêtre s'accompagne d'un redimensionnement des éléments graphiques.

Si vous testez la fenêtre (Menu Tools - Form Editor – Preview), vous voyez un aperçu de la fenêtre. Vous pouvez constater que lorsque l'on redimensionne la fenêtre, les éléments graphiques ne changent pas de taille. Pour résoudre ce problème, nous allons attacher à la zone principale un gestionnaire de présentation. Il en existe de différents types.

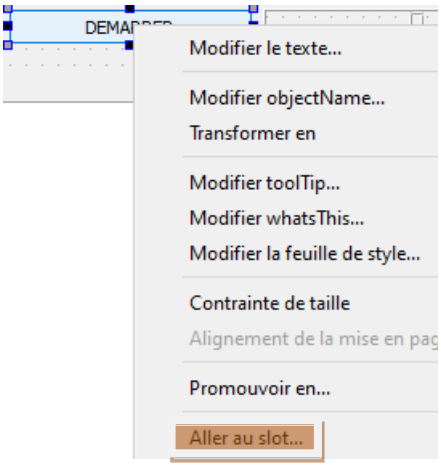
- verticaux
- horizontaux
- sur une grille
- de type formulaire
  (label à gauche et zone d'édition à droite)



Nous allons choisir un « layout » de type Horizontal Layout. Pour cela, sélectionner la zone principale de la fenêtre (le fond), puis cliquer sur l'icône « Lay Out horizontally » ou en Français "Mettre en page horizontalement". Si vous testez de nouveau l'IHM, vous devriez voir que les éléments graphiques s'adaptent maintenant à la taille de la fenêtre.

3 Codage

Nous allons dans un premier temps uniquement gérer le bouton **Démarrer/Arrêter.**



- Dans l'environnement graphique, sélectionner le bouton "Démarrer" puis choisir le menu contextuel "Aller au slot".
- Choisir ensuite le signal "clicked()" puis OK
- Qt vous ouvre maintenant l'éditeur de code et vous place le curseur sur la méthode "`on_pbStartStop_clicked()`" qui sera exécutée chaque fois que l'on cliquera sur le bouton. Remarquez aussi la déclaration de cette méthode dans le fichier de déclaration "`mainwindow.h`" dans une zone "private slots:"
- Ajouter ligne `#include <QtWidgets>` en haut du fichier "`mainWindow.h`". Ceci nous permettra d'utiliser les Widgets dans notre projet
- Compléter la méthode `on_pbStartStop_clicked()`

```
void MainWindow::on_pbStartStop_clicked()
{
    QMessageBox::about(this,"Test","Appui sur le bouton startStop");
}
```

- Tester votre application. Vérifiez que le clic sur le bouton startStop affiche le message « Appui sur le bouton startStop ».
- Rajoutez un slot pour la gestion du bouton PAUSE.
- Après avoir testé vos slots, supprimez les lignes QMessageBox ::about(.....) ;
- Nous allons maintenant écrire un peu de code dans la méthode ***on_pbStartStop_clicked()*** pour changer le nom du bouton à chaque clic (1er clic ➔ arrêter, 2ème clic ➔ démarrer, 3ème clic ➔ arrêter, etc…)

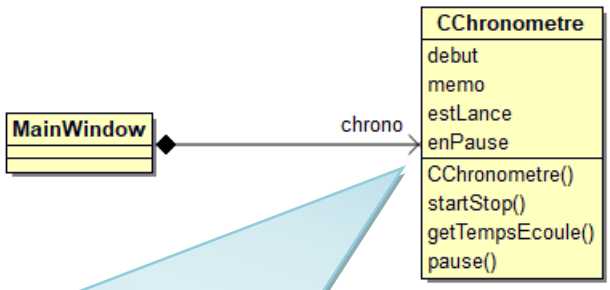
  Pour changer le nom du bouton, il suffit d'écrire : ui***->btStartStop->setText("NouveauNom");***

  Pour connaître le nom d'un bouton, il faut appeler ***ui->pbStartStop->text()*** qui retourne la chaîne de caractères actuelle correspondant au nom (type QString sous Qt, équivalent de la classe <string> de la STL).

  Ecrivez le code de cette méthode pour changer le nom du bouton à chaque clic.

3.1 Utilisation dans cette fenêtre de la classe CChronometre (testée en mode console)

Nous allons modéliser le diagramme de classe suivant :



Nous allons réaliser une **composition applicative**

Le principe consiste à

- déclarer chrono en tant que **pointeur** dans la classe MainWindow
- faire une **allocation mémoire** de cet attribut chrono **dans le constructeur** de MainWindow de MainWindow

- Copier les fichiers Chronometre.h et Chronometre.cpp dans le répertoire de votre projet Qt.
- Ajouter ces fichiers à votre projet (bouton droit sur le projet, puis Ajouter fichier existant).
- Maintenant, nous allons déclarer un pointeur sur un objet de la classe CChronometre dans le fichier MainWindow.h (pour que le chronometre puisse être connu, donc utilisé dans toutes les méthodes de la fenêtre graphique). Nous allons aussi créer ce chronomètre au lancement de l'application (dans le constructeur de la fenêtre graphique)

Mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QtWidgets>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class CChronometre;

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_pbStartStop_clicked();

    void on_pbPause_clicked();

private:
    Ui::MainWindow *ui;
    CChronometre *chrono;
};
#endif // MAINWINDOW_H
```

Il faut que la classe CChronometre soit connue de la fenêtre graphique

Déclaration d'un pointeur sur la classe CChonometre.

Mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "CChronometre.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    chrono=new CChronometre;
}

…
```

Ce **new** montre bien que c'est une composition (losange rempli en UML) et non une agrégation (losange vide).

En effet, **new** montre que l'on crée l'objet chrono en même temps que la fenêtre principale. Le chronomètre et la fenêtre principale ont donc des durées de vie liées (caractéristique principale de la composition).

- Vérifiez que votre programme compile après avoir ajouté ces modifications. N'allez pas plus loin s'il ne compile pas, corrigez d'abord vos erreurs.
- Complétez la méthode **on_btStartStop_clicked()** de la fenêtre graphique pour lancer ou stopper le chronomètre lors de chaque appui sur le bouton. Faire de même avec le bouton PAUSE

Il nous reste maintenant à gérer l'affichage. Pour cela, nous allons devoir créer un **Timer** qui va nous permettre de rafraîchir l'affichage en permanence. Le **Timer** permet simplement de rendre multitâches une application graphique.

Les modifications à apporter sont les suivantes :

- **Dans le fichier MainWindow.h**
  - Création d'un pointeur sur un objet de la classe **QTimer**
  - Déclaration d'une méthode qui va être appelée régulièrement.
- **Dans le fichier MainWindow.cpp**
  - Dans le constructeur : Création du timer et association de ce timer à une méthode qui va être appelée régulièrement.
  - Codage de la méthode appelée régulièrement pour qu'elle mette à jour l'affichage.

Mainwindow.h

```
…

class CChronometre;

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
…

private slots:
    void on_pbStartStop_clicked();

    void on_pbPause_clicked();
    void update();
private:
    Ui::MainWindow *ui;
    CChronometre *chrono;
    QTimer *timer;
};
#endif // MAINWINDOW_H
```

Méthode appelée régulièrement par le Timer

Mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "CChronometre.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    chrono=new CChronometre;

    timer=new QTimer(this); // allocation mémoire
    connect(timer, SIGNAL(timeout()), this, SLOT(update()));

     timer->start(10);
}
…

void MainWindow::update()
{
    ui->lcdChrono->display((int)chrono->getTempsEcoule());
}
```

La méthode update() sera lancée régulièrement par le timer

Lancement du timer avec une durée de 10 millisecondes ➔ Chaque fois que 10 millisecondes sont écoulées, la méthode update() est lancée

Mise à jour de l'affichage LCD

FOIRE AU 20 : Améliorer le programme en :

- Séparant les secondes des millièmes de secondes dans l'affichage
- Gérant plusieurs chronomètres (nombre de chronomètres entre 1 et 3 choisi par l'opérateur)