Assn#3 Remedial & Qt Tutorial #1

2017.11.20

Any Question for Assn3?

Introduction

Introduction

- Graphical User Interface (GUI)
  - 그래픽 요소를 사용한 유저 인터페이스
  - Ex. 창, 텍스트 상자, 버튼 등

- GUI toolkits
  - Qt
  - MFC
  - wxWidget
  - Motif
  - GTK+

What is Qt?

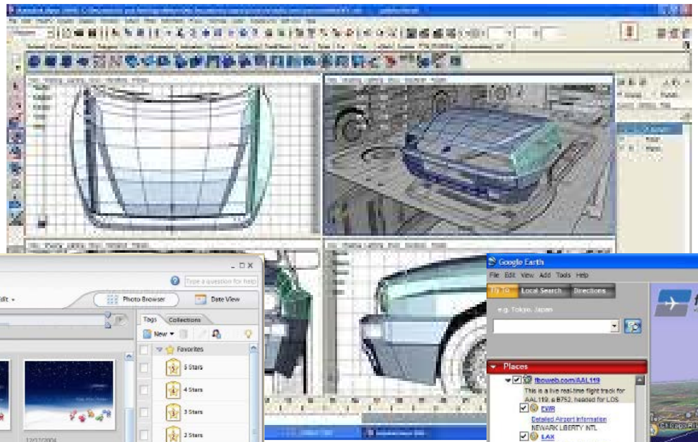
- Qt is a GUI toolkit
  - GUI를 필요로 하는 소프트웨어 개발에 널리 사용
  - Trolltech 회사에서 개발 (이후 다른 회사에서 인수)
  - 처음에는 User interface를 위해서 만들어졌지만,
    현재는 database, networking 등 일반적인 목적으로도 많이 사용 가능
  - Dual License
    - Commercial license
    - Open source license

Qt의 장점?

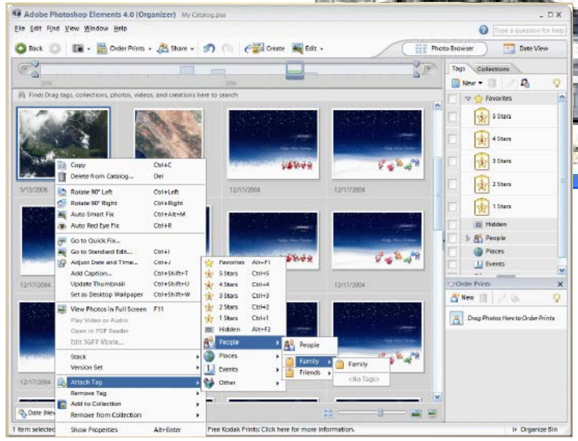
- Qt is a cross-platform toolkit
  - Windows, Mac OS X, Linux/Unix, Mobile, etc.
- Cross-platform?
  - Applications are not bound to given HW or OS
- Various languages
  - C++, C#, Python, etc.

Where can you see Qt?

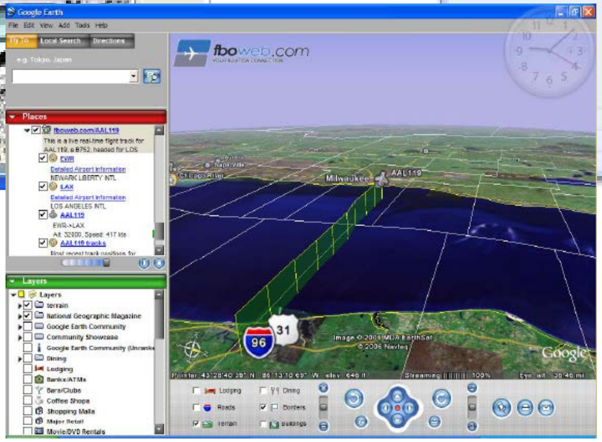
Autodesk Maya



Skype



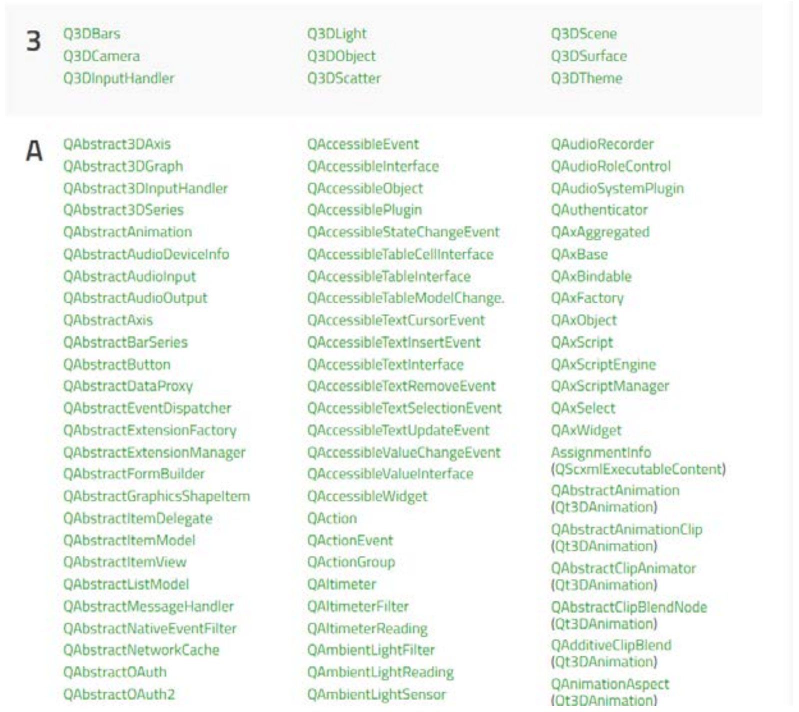
Abode Photoshop Elements



Google Earth

Qt Classes

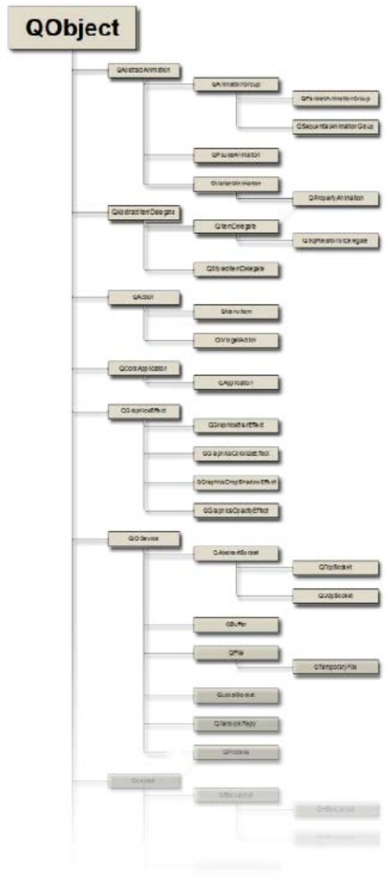
- Qt is written in C++ with OOP concepts
  - All modules in Qt are composed of ‘Classes’
  - Every class has its own methods
  - Each object is capable of sending messages to other objects and receiving messages by **signal/slot**
  - http://doc.qt.io/qt-5/classes.html



3	Q3DBars	Q3DLight	Q3DScene
	Q3DCamera	Q3DObject	Q3DSurface
	Q3DInputHandler	Q3DScatter	Q3DTheme
A	QAbstract3DAxis	QAccessibleEvent	QAudioRecorder
	QAbstract3DGraph	QAccessibleInterface	QAudioRoleControl
	QAbstract3DInputHandler	QAccessibleObject	QAudioSystemPlugin
	QAbstract3DSeries	QAccessiblePlugin	QAuthenticator
	QAbstractAnimation	QAccessibleStateChangeEvent	QAxAggregated
	QAbstractAudioDeviceInfo	QAccessibleTableCellInterface	QAxBase
	QAbstractAudioInput	QAccessibleTableInterface	QAxBindable
	QAbstractAudioOutput	QAccessibleTableModelChange.	QAxFactory
	QAbstractAxis	QAccessibleTextCursorEvent	QAxObject
	QAbstractBarSeries	QAccessibleTextInsertEvent	QAxScript
	QAbstractButton	QAccessibleTextInterface	QAxScriptEngine
	QAbstractDataProxy	QAccessibleTextRemoveEvent	QAxScriptManager
	QAbstractEventDispatcher	QAccessibleTextSelectionEvent	QAxSelect
	QAbstractExtensionFactory	QAccessibleTextUpdateEvent	QAxWidget
	QAbstractExtensionManager	QAccessibleValueChangeEvent	AssignmentInfo (QScxmlExecutableContent)
	QAbstractFormBuilder	QAccessibleValueInterface	QAbstractAnimation (Qt3DAnimation)
	QAbstractGraphicsShapeItem	QAccessibleWidget	QAbstractAnimationClip (Qt3DAnimation)
	QAbstractItemDelegate	QAction	QAbstractClipAnimator (Qt3DAnimation)
	QAbstractItemModel	QActionEvent	QAbstractClipBlendNode (Qt3DAnimation)
	QAbstractItemView	QActionGroup	QAdditiveClipBlend (Qt3DAnimation)
	QAbstractListModel	QAltimeter	QAnimationAspect (Qt3DAnimation)
	QAbstractMessageHandler	QAltimeterFilter	
	QAbstractNativeEventFilter	QAltimeterReading	
	QAbstractNetworkCache	QAmbientLightFilter	
	QAbstractOAuth	QAmbientLightReading	
	QAbstractOAuth2	QAmbientLightSensor	

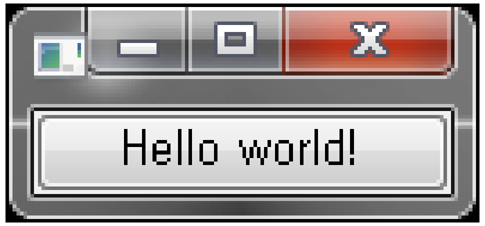
QObject Class

- QObject is the base class of almost all Qt classes and all widgets
  - QString
  - QVector
  - QImage
  - QWidgets
    - QLabel
    - QPushButton, QRadioButton
    - QCheckBox
    - QLineEdit
    - Qslider, QSpinBox
    - …



Sample Code

```
#include <QApplication>
#include <QPushButton>

int main(int argc, char *argv[]){
   QApplication app(argc, argv);

   QPushButton hello("Hello world!");

   hello.show();
   return app.exec();
}
```

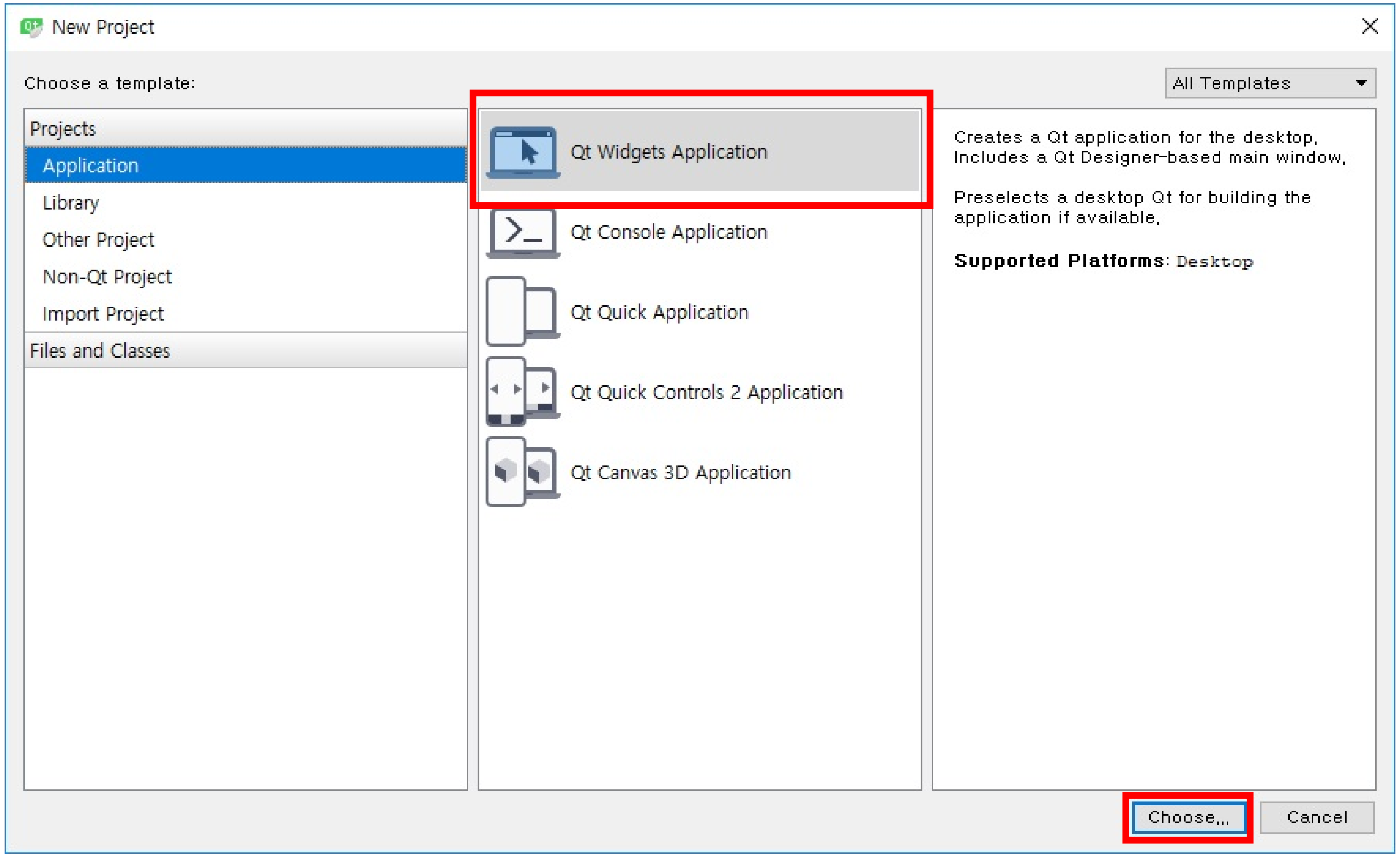


Qt Creator?

- Qt GUI Application development framework
- Visual debugger, integrated GUI layout & forms
- There are other solutions for Qt development (e.g. MS Visual Studio + Qt SDK)

Tutorial

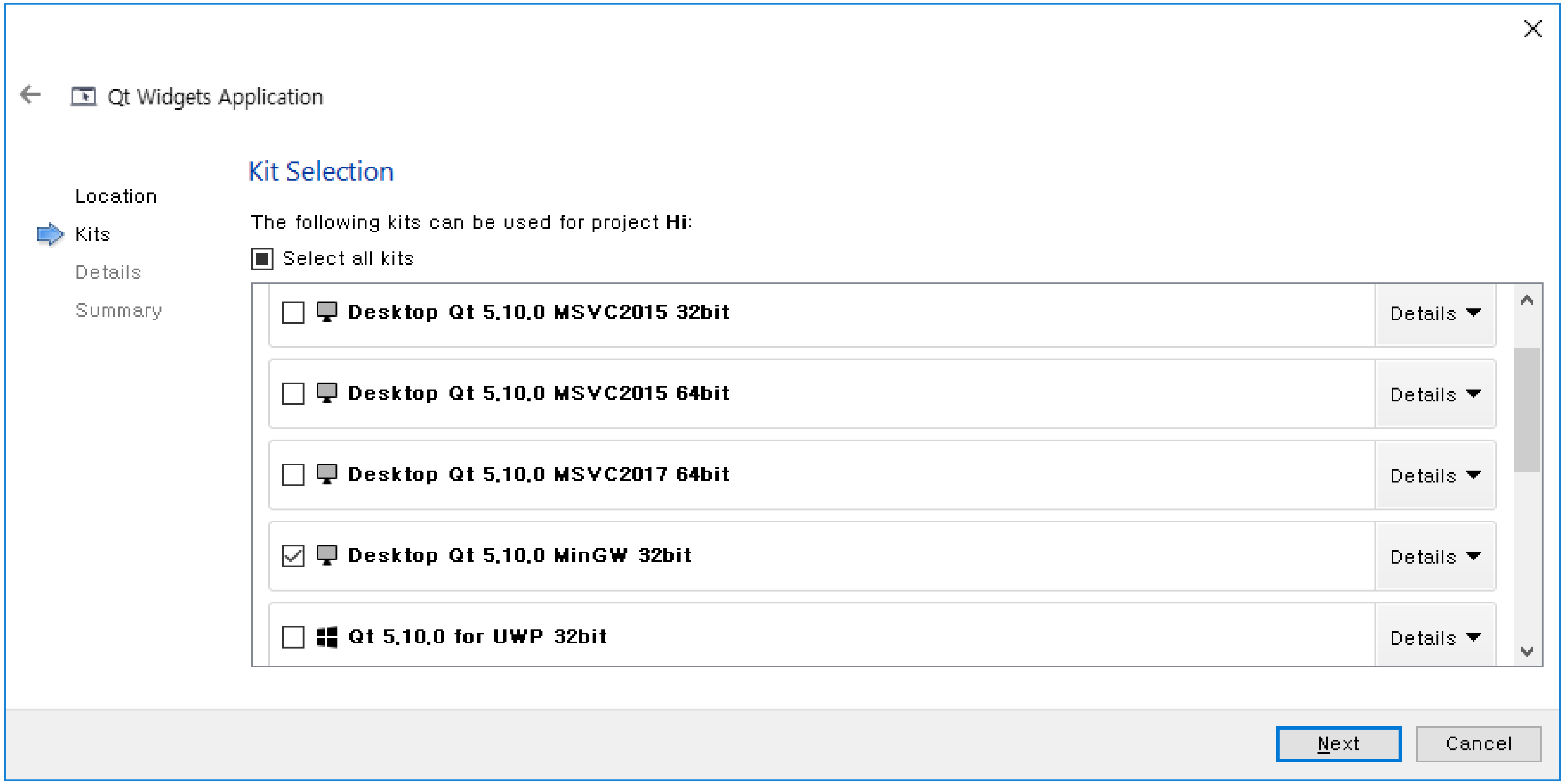
샘플 프로젝트 생성



Creates a Qt application for the desktop. Includes a Qt Designer-based main window.

Preselects a desktop Qt for building the application if available.

Supported Platforms: Desktop


Qt Widgets Application

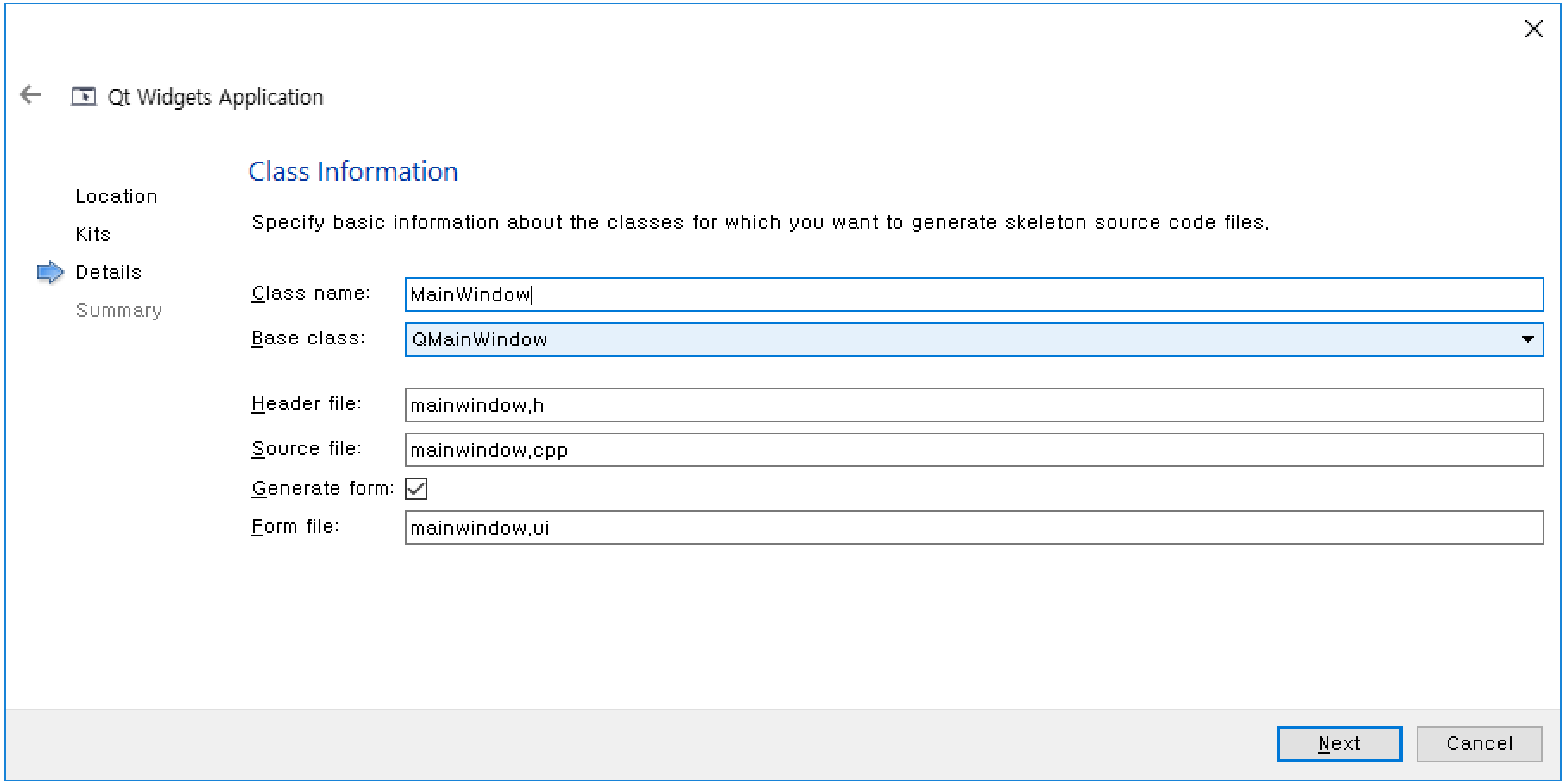
Kit Selection

Location
Kits
Details
Summary

The following kits can be used for project **Hi**:

Select all kits

- [ ] Desktop Qt 5.10.0 MSVC2015 32bit
- [ ] Desktop Qt 5.10.0 MSVC2015 64bit
- [ ] Desktop Qt 5.10.0 MSVC2017 64bit
- [✓] Desktop Qt 5.10.0 MinGW 32bit
- [ ] Qt 5.10.0 for UWP 32bit



Qt Widgets Application

Location
Kits
Details
Summary

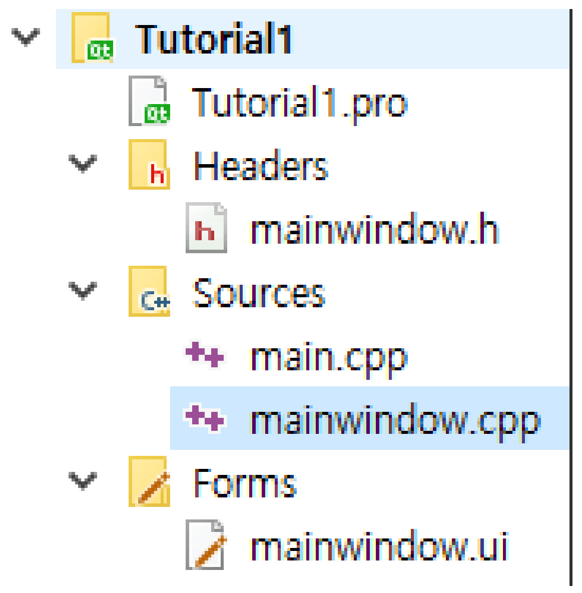
Class Information

Specify basic information about the classes for which you want to generate skeleton source code files.

Class name: MainWindow
Base class: QMainWindow
Header file: mainwindow.h
Source file: mainwindow.cpp
Generate form: ✓
Form file: mainwindow.ui

Next Cancel

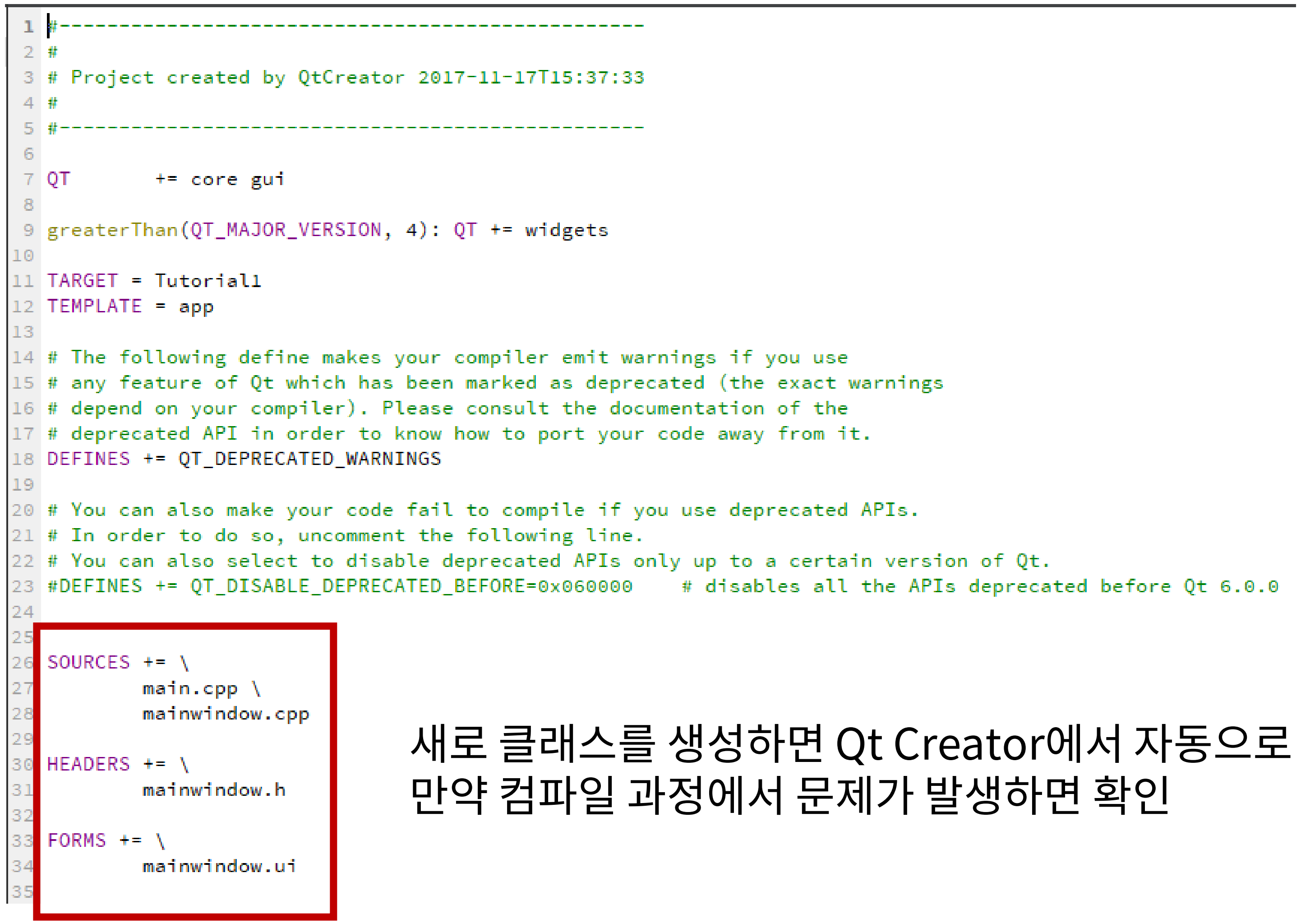
파일 설명


Tutorial1
Tutorial1.pro
Headers
mainwindow.h
Sources
main.cpp
mainwindow.cpp
Forms
mainwindow.ui

PROJECT_NAME.pro: 프로젝트 설정 파일, 일종의 makefile

WINDOW_CLASS.ui: 윈도우에 대한 UI

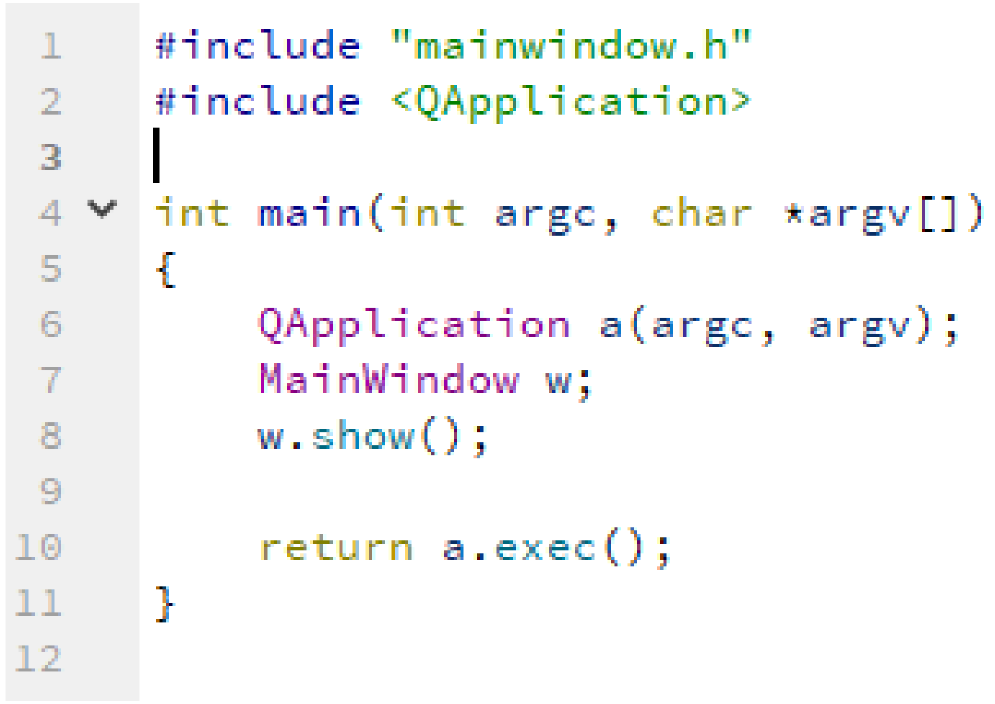
Pro 파일



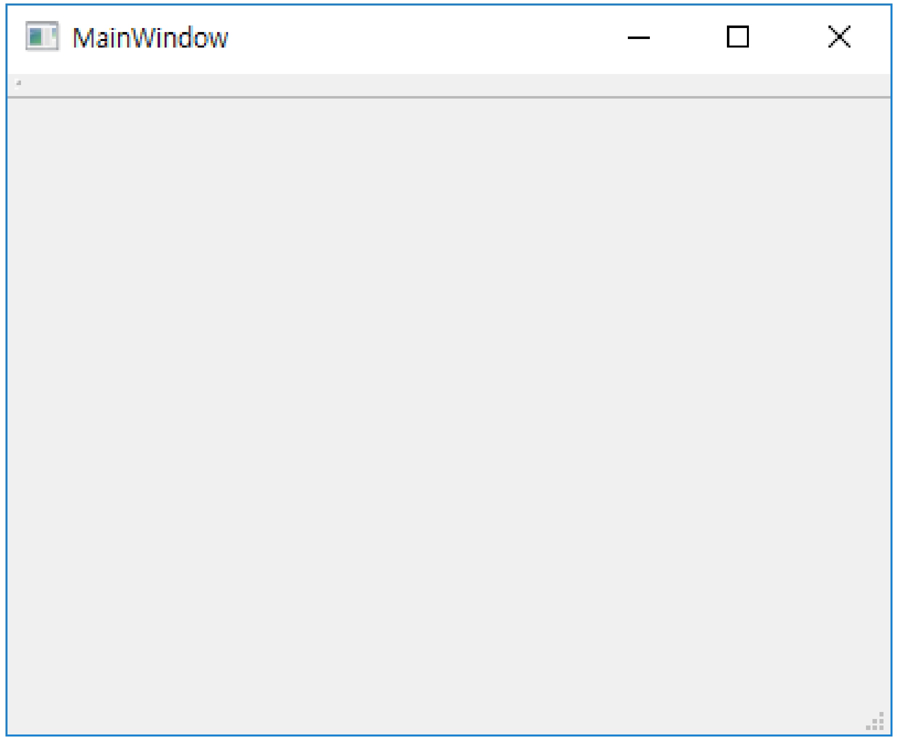
```
#-------------------------------------------------
#
# Project created by QtCreator 2017-11-17T15:37:33
#
#-------------------------------------------------

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = Tutorial1
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0


SOURCES += \
        main.cpp \
        mainwindow.cpp

HEADERS += \
        mainwindow.h

FORMS += \
        mainwindow.ui

```

새로 클래스를 생성하면 Qt Creator에서 자동으로 추가
만약 컴파일 과정에서 문제가 발생하면 확인

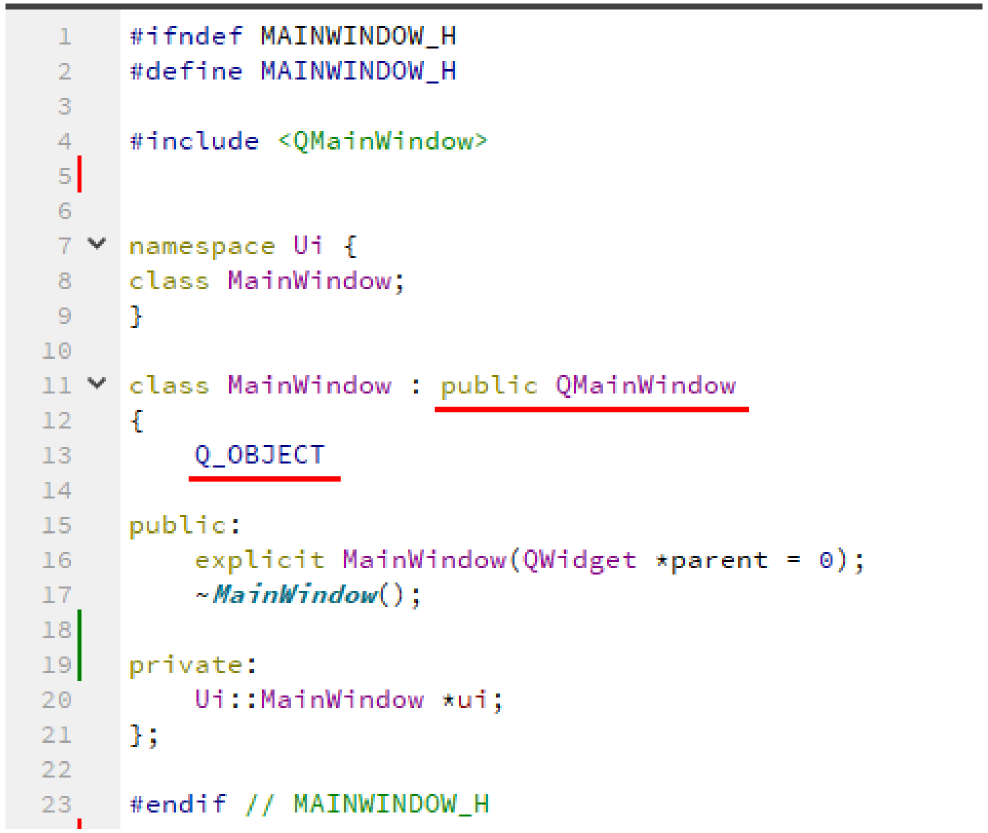
Main



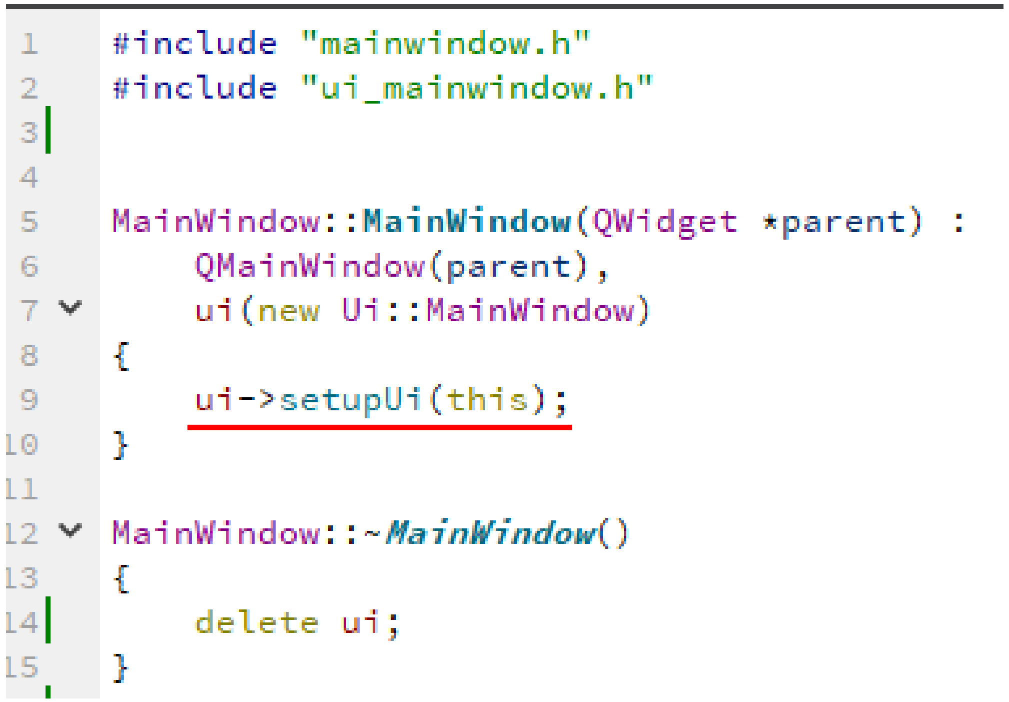
```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}

```



MainWindow



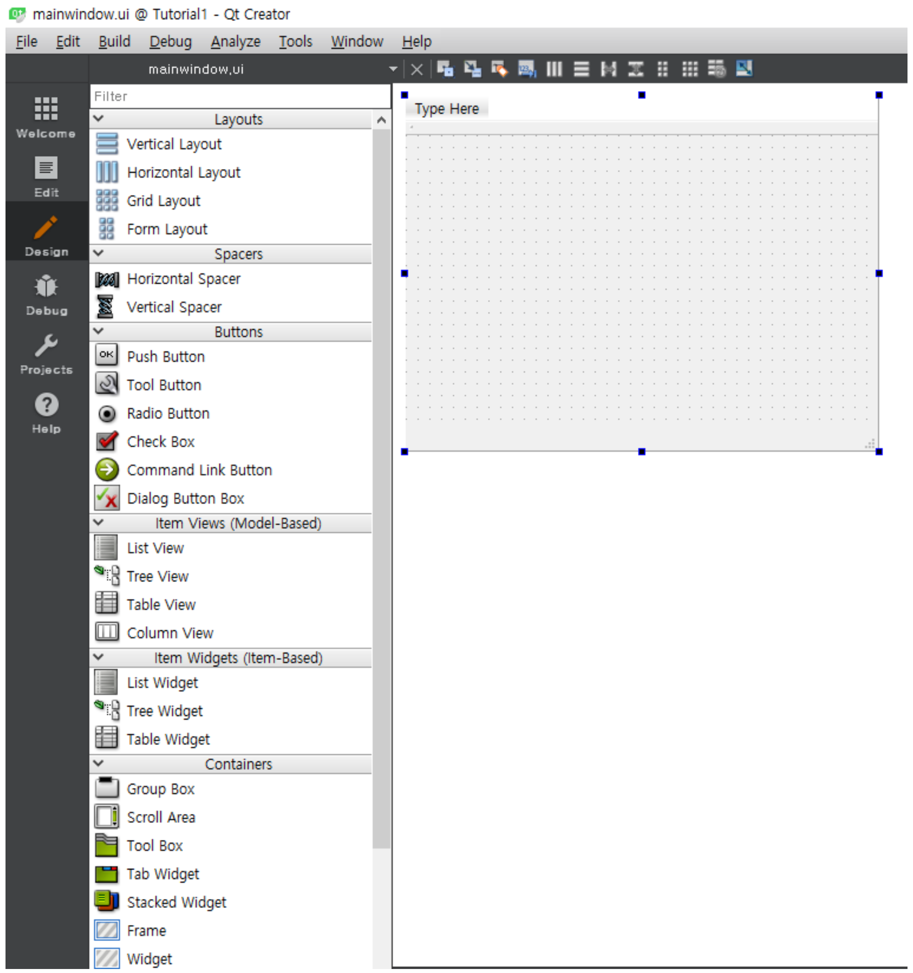
```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>


namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

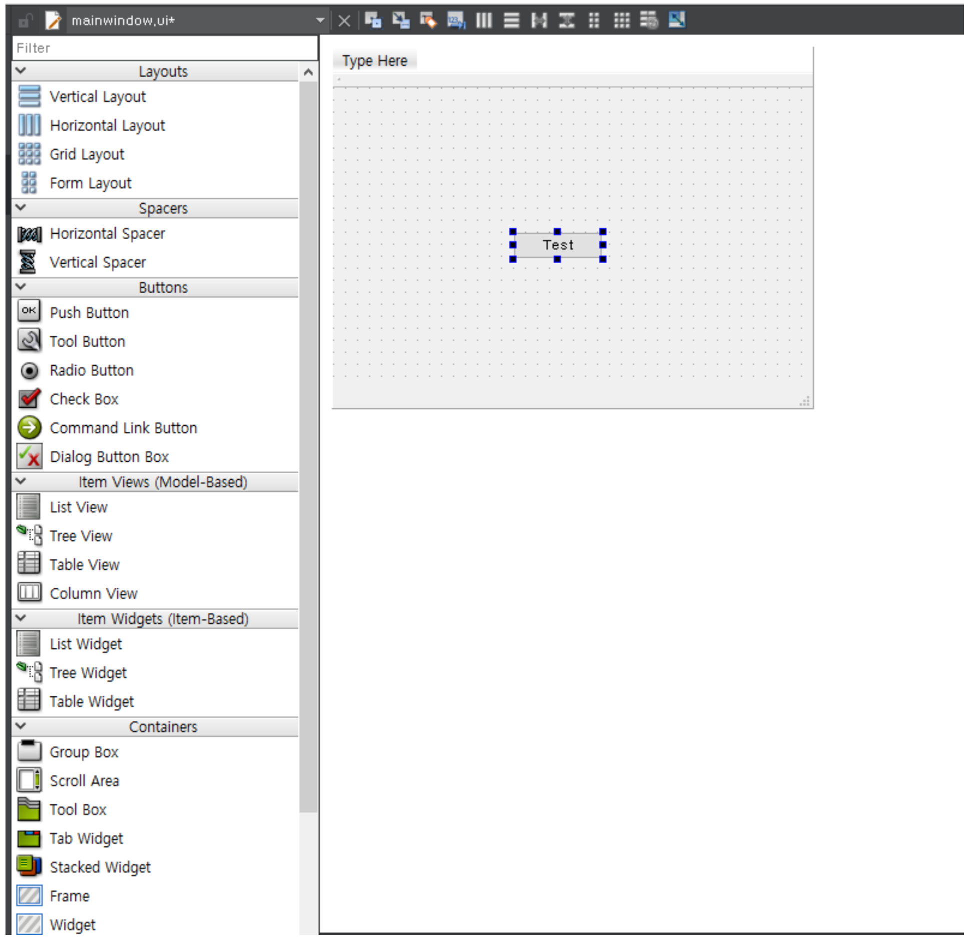


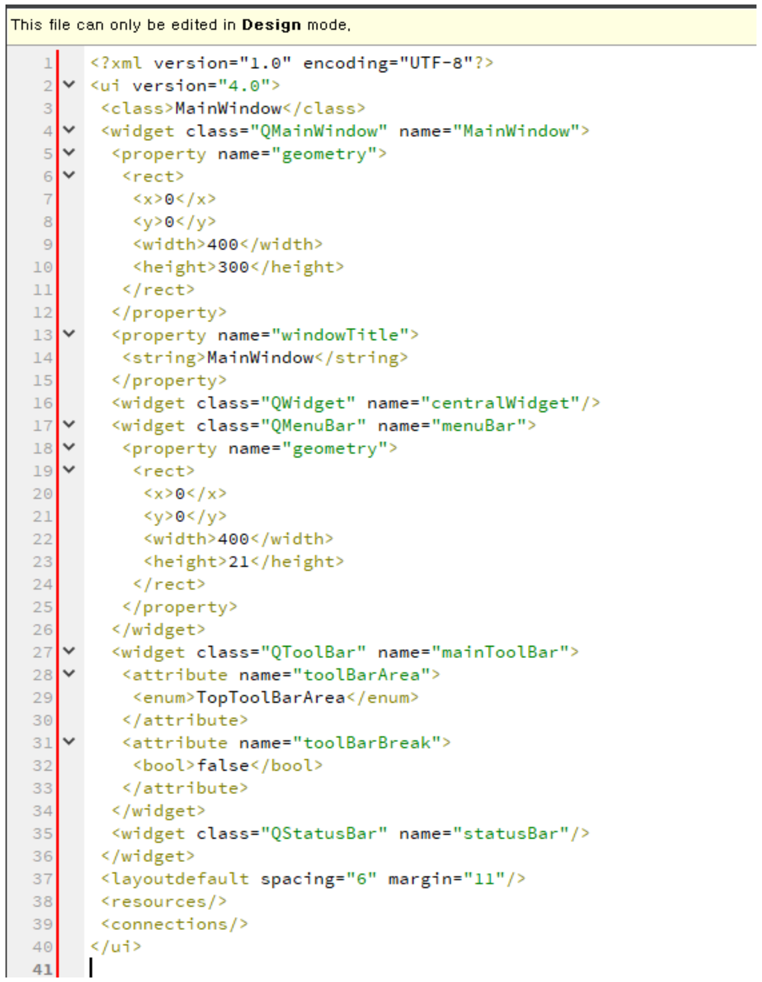
```
#include "mainwindow.h"
#include "ui_mainwindow.h"


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

MainWindow::~MainWindow()
{
    delete ui;
}
```

Ui 파일

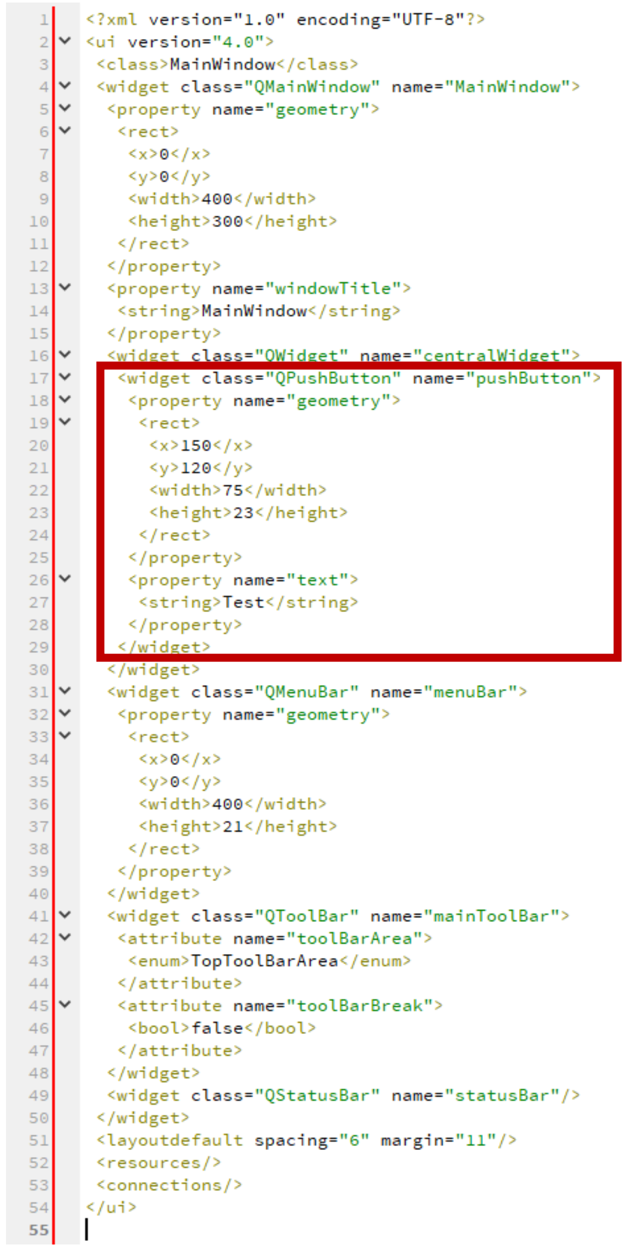
Ui 파일을 선택하면 Design 창으로 전환
왼쪽 리스트에서 원하는 위젯을 오른쪽 화면에 넣으면 메인 윈도우 디자인 변화
아래 그림은 Push Button을 넣고, 더블클릭하여 Text를 "Test"로 바꾸었음





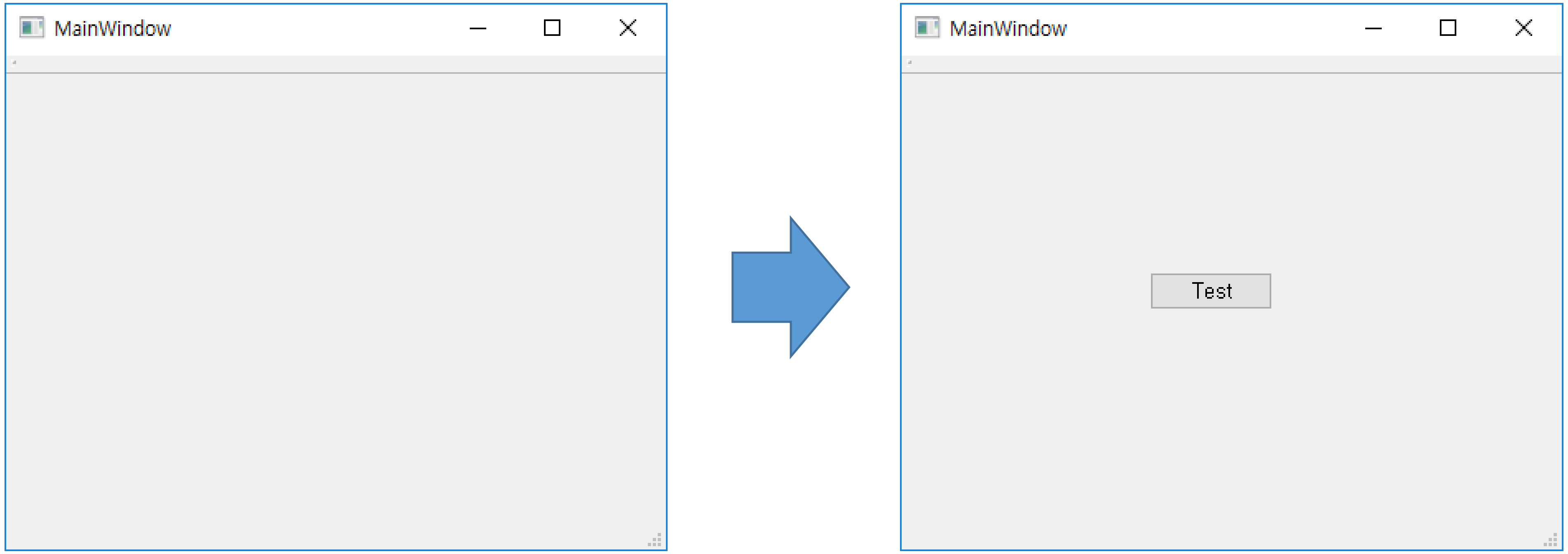

This file can only be edited in **Design** mode.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>300</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget"/>
  <widget class="QMenuBar" name="menuBar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>400</width>
     <height>21</height>
    </rect>
   </property>
  </widget>
  <widget class="QToolBar" name="mainToolBar">
   <attribute name="toolBarArea">
    <enum>TopToolBarArea</enum>
   </attribute>
   <attribute name="toolBarBreak">
    <bool>false</bool>
   </attribute>
  </widget>
  <widget class="QStatusBar" name="statusBar"/>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>300</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <widget class="QPushButton" name="pushButton">
    <property name="geometry">
     <rect>
      <x>150</x>
      <y>120</y>
      <width>75</width>
      <height>23</height>
     </rect>
    </property>
    <property name="text">
     <string>Test</string>
    </property>
   </widget>
  </widget>
  <widget class="QMenuBar" name="menuBar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>400</width>
     <height>21</height>
    </rect>
   </property>
  </widget>
  <widget class="QToolBar" name="mainToolBar">
   <attribute name="toolBarArea">
    <enum>TopToolBarArea</enum>
   </attribute>
   <attribute name="toolBarBreak">
    <bool>false</bool>
   </attribute>
  </widget>
  <widget class="QStatusBar" name="statusBar"/>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```

버튼을 넣어보고 실행



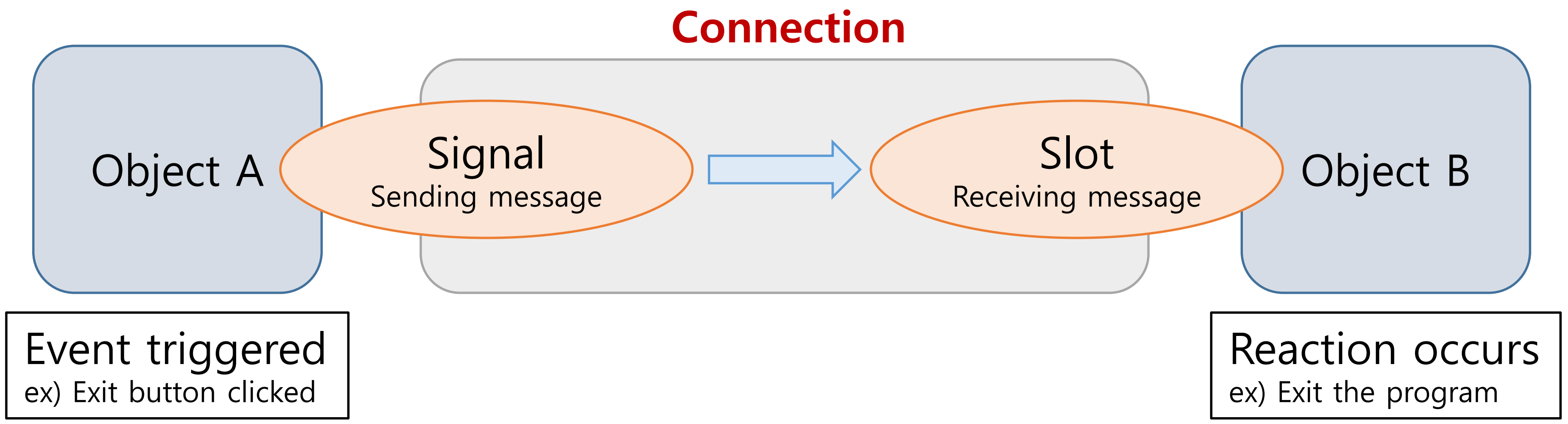
지금은 버튼을 눌러도 아무 일이 발생하지 않음

Signal & Slot

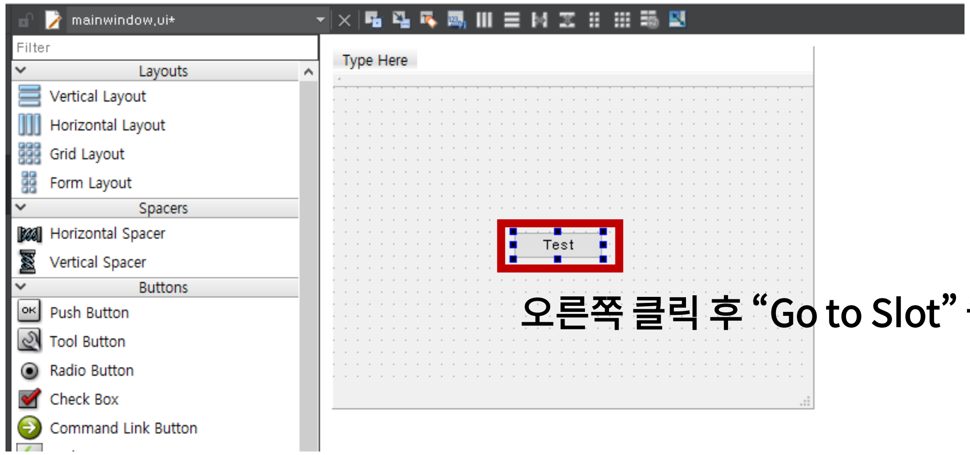
Event-Driven 방식

객체와 객체를 연결

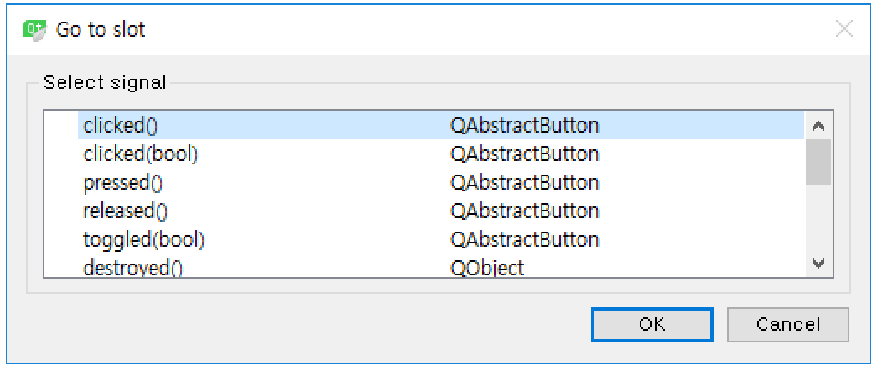
A 객체에서 신호(Signal)가 발생하면, B 객체의 슬롯에서 특정한 업무 진행



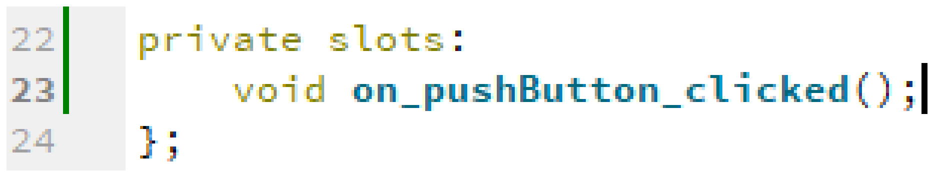
Slot



오른쪽 클릭 후 “Go to Slot” 클릭

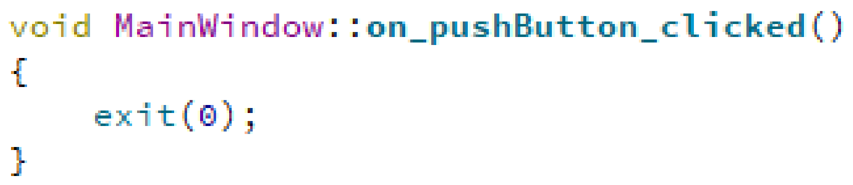


clicked()	QAbstractButton
clicked(bool)	QAbstractButton
pressed()	QAbstractButton
released()	QAbstractButton
toggled(bool)	QAbstractButton
destroyed()	QObject



```
private slots:
    void on_pushButton_clicked();
};
```

MainWindow.h

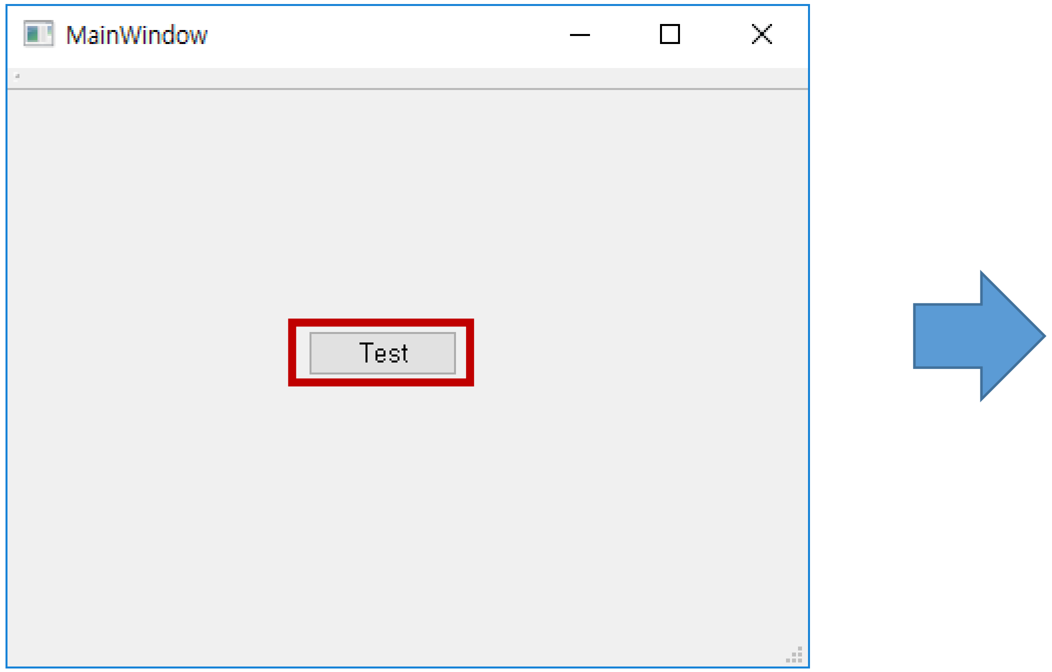


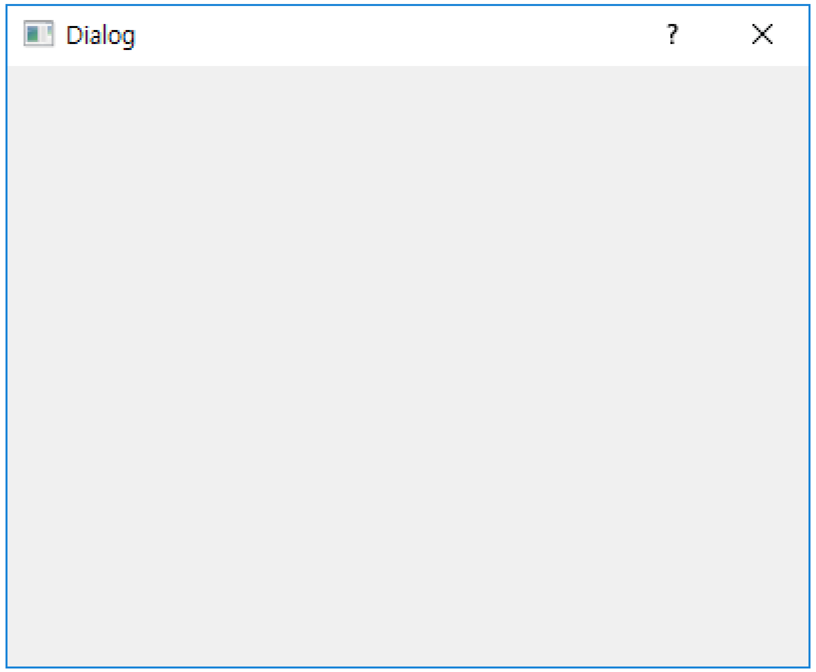
```
void MainWindow::on_pushButton_clicked()
{
    exit(0);
}
```

MainWindow.cpp

실행해서 버튼을 눌러보면 프로그램이 종료됨

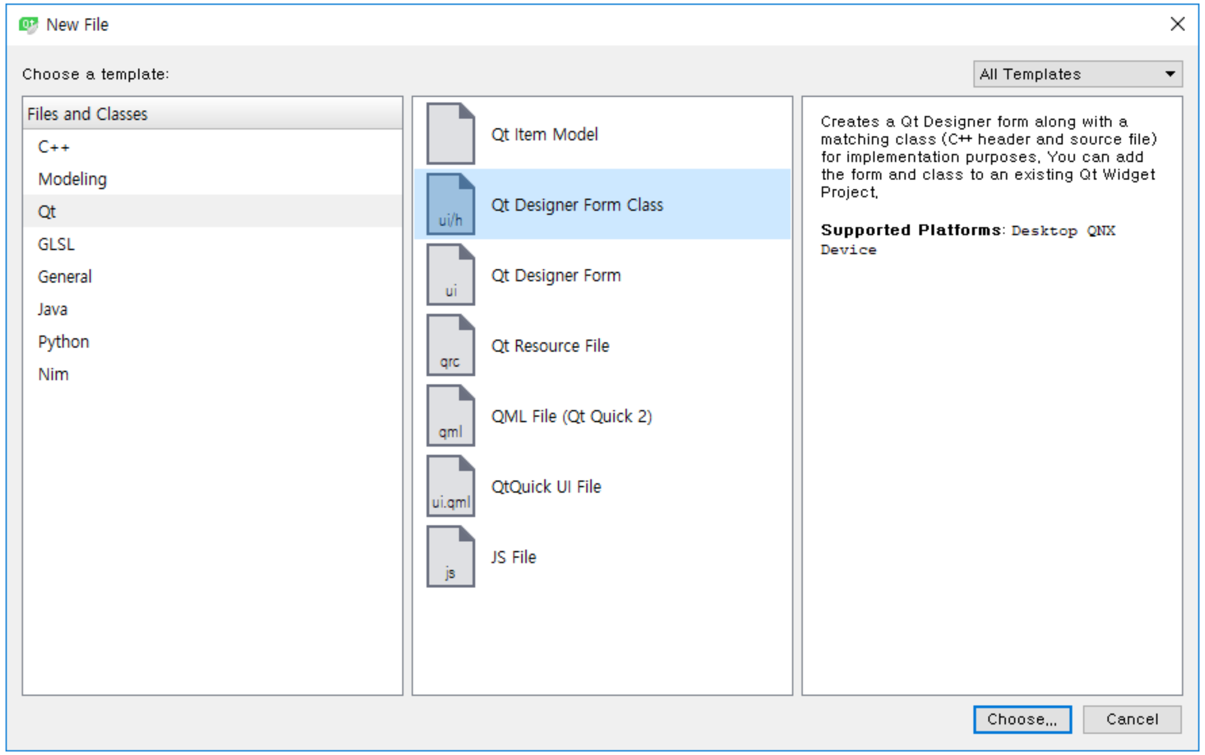
응용: 버튼을 누르면 새로운 화면 띄우기





1. 새로운 화면에 대한 클래스 정의
2. 새 화면 객체 생성
3. show() 메소드 사용

새로운 화면 만들기


New File

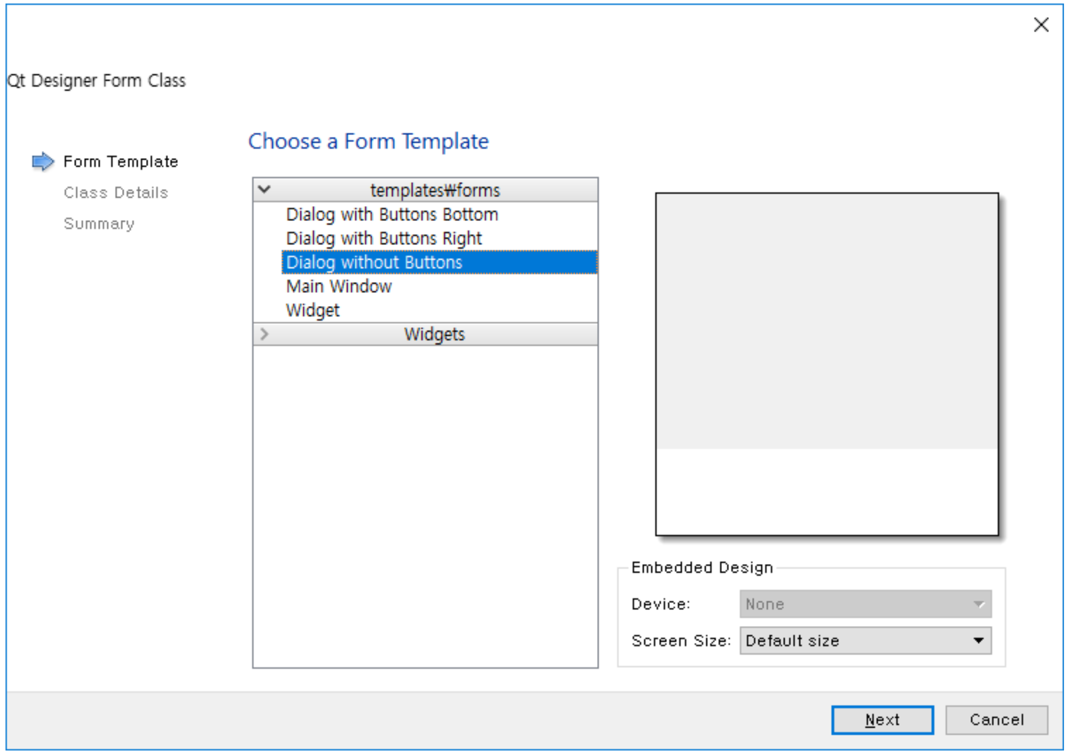
Choose a template: All Templates

Files and Classes
- C++
- Modeling
- Qt
- GLSL
- General
- Java
- Python
- Nim

- Qt Item Model
- Qt Designer Form Class
- Qt Designer Form
- Qt Resource File
- QML File (Qt Quick 2)
- QtQuick UI File
- JS File

Creates a Qt Designer form along with a matching class (C++ header and source file) for implementation purposes. You can add the form and class to an existing Qt Widget Project.

Supported Platforms: Desktop QNX Device


Qt Designer Form Class

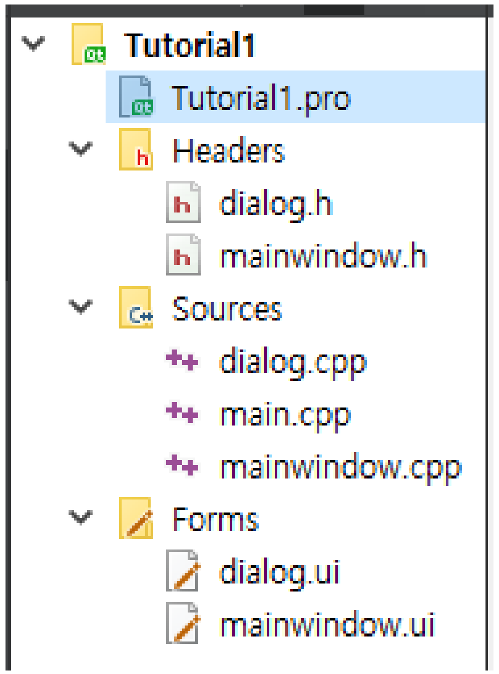
Choose a Form Template

- Form Template
- Class Details
- Summary

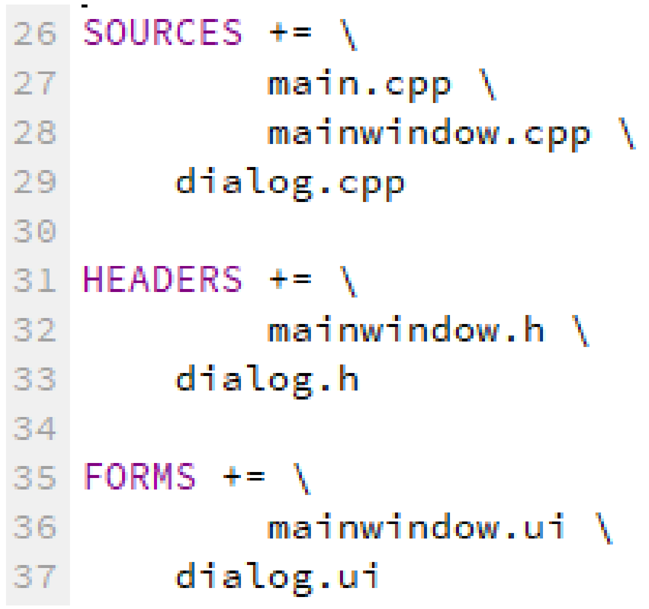
templates₩forms
- Dialog with Buttons Bottom
- Dialog with Buttons Right
- Dialog without Buttons
- Main Window
- Widget

Widgets

Embedded Design
Device: None
Screen Size: Default size

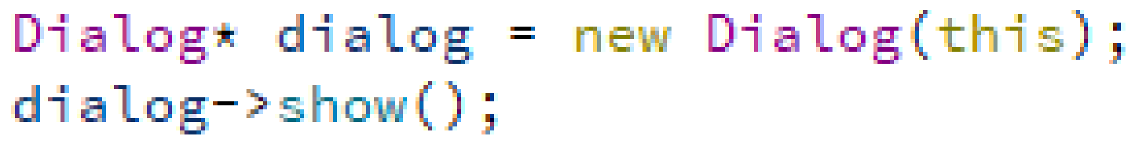


Tutorial1
Tutorial1.pro
Headers
dialog.h
mainwindow.h
Sources
dialog.cpp
main.cpp
mainwindow.cpp
Forms
dialog.ui
mainwindow.ui



```
SOURCES += \
        main.cpp \
        mainwindow.cpp \
    dialog.cpp

HEADERS += \
        mainwindow.h \
    dialog.h

FORMS += \
        mainwindow.ui \
    dialog.ui
```

응용: 버튼을 누르면 새로운 화면 띄우기

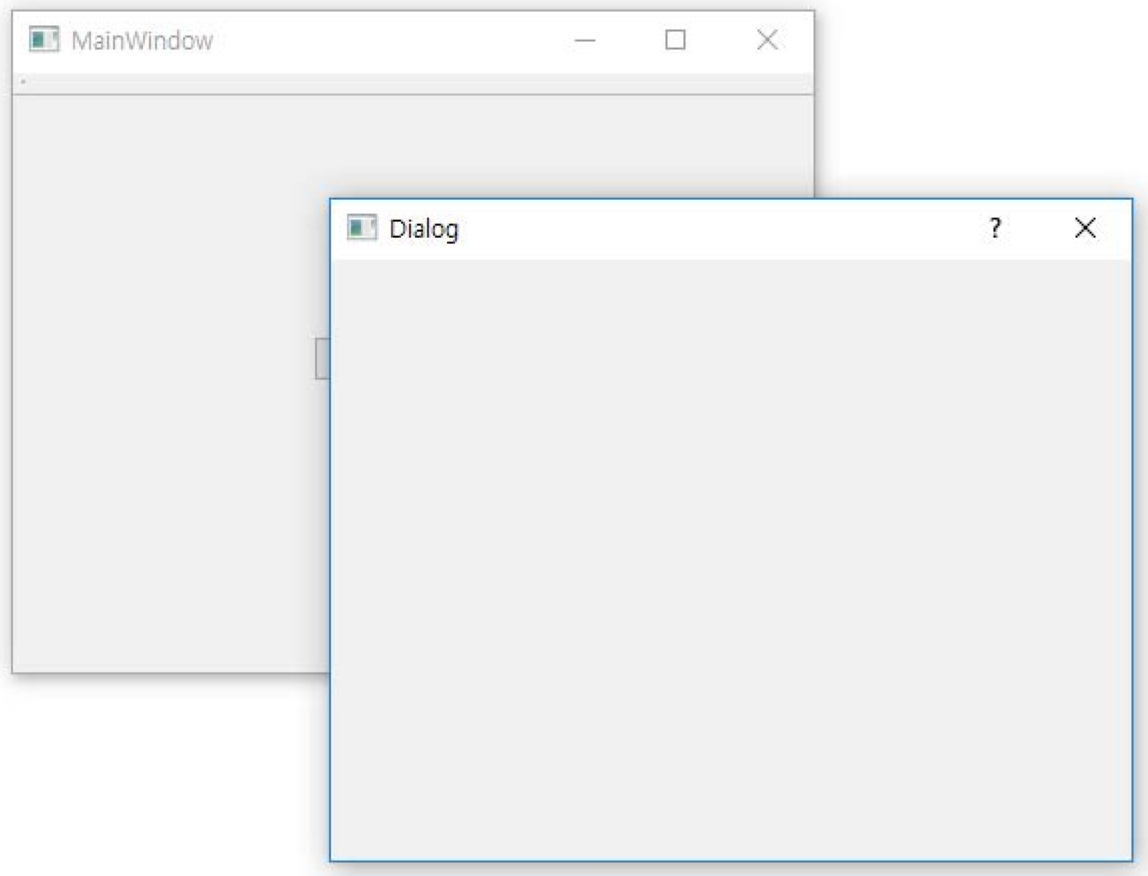


```
Dialog* dialog = new Dialog(this);
dialog->show();
```

- 위의 코드처럼 새로 만든 윈도우에 대해 객체를 생성하고, show() 메소드를 사용해 새로운 화면을 띄울 수 있습니다.

- 이제 버튼을 누르면 새로운 화면이 나타나도록 구현해보세요

응용: 버튼을 누르면 새로운 화면 띄우기



새로운 화면이 나타났지만, 이전 화면이 남아 있음.

이전 화면을 없애고 싶으면, hide()를 사용

QTimer

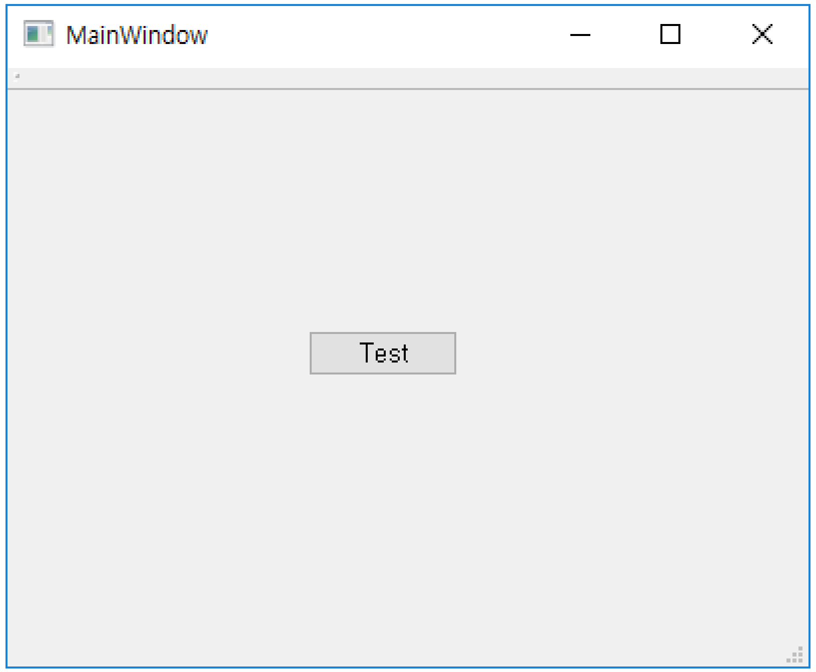
- 일정 시간 간격으로 timeout() 시그널을 보낼 수 있는 클래스

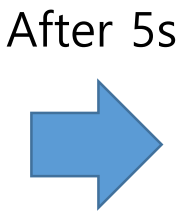
```
QTimer* timer = new QTimer();

connect (객체 A (timer), SIGNAL(timeout()), 객체 B, SLOT(FUNTION()));

timer->start(2000);        ← ms 단위
```

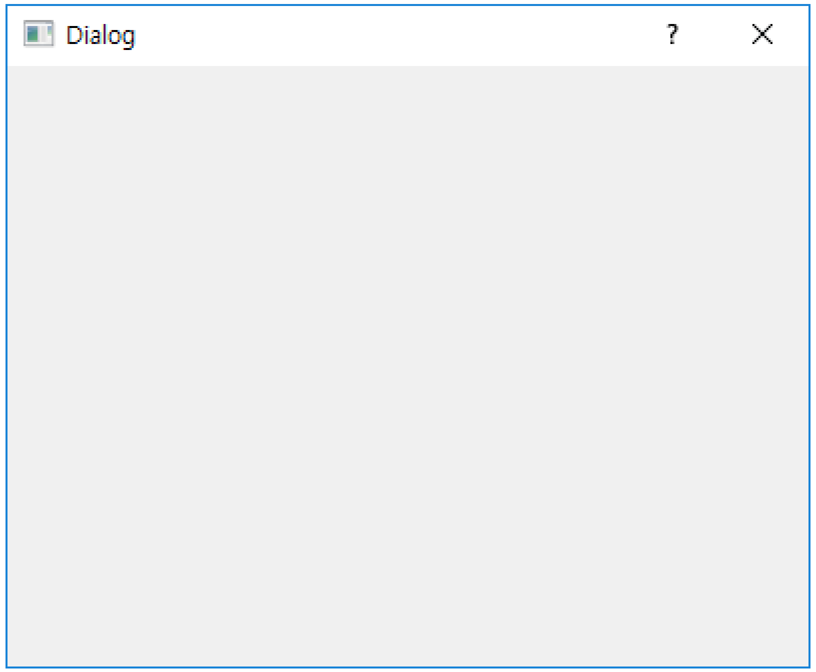
Class 정의에 Public slots에 함수의 형태로 슬롯 정의

```
public slots:
        void FUNCTION();
```

응용: 실행 후 5초가 지나면 새로운 화면 띄우기







Hint: Slot 정의 -> Qtimer 객체 생성 -> Connect -> Qtimer 시작

QMessageBox

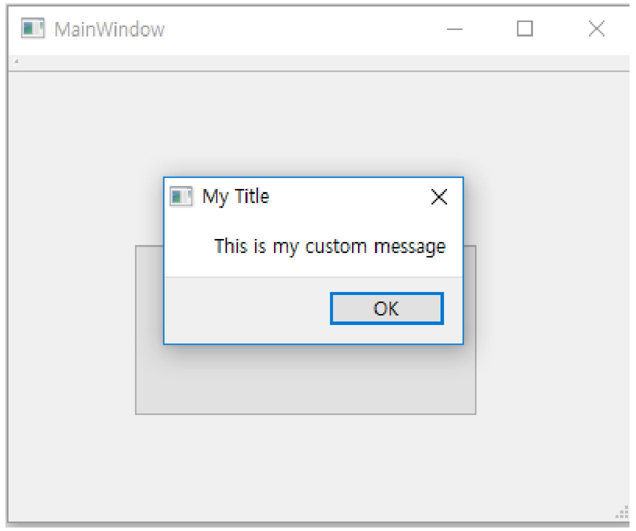
- The QMessageBox class provides a modal dialog for informing the user or for asking the user a question and receiving an answer

- Message type

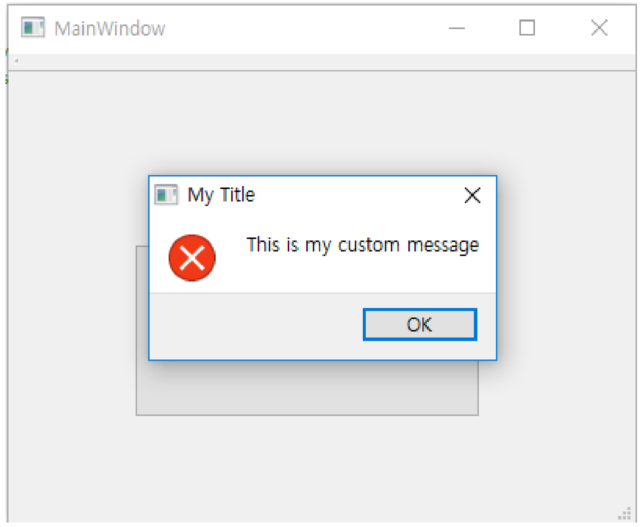
❓	Question	For asking a question during normal operations.
ℹ	Information	For reporting information about normal operations.
⚠	Warning	For reporting non-critical errors.
❌	Critical	For reporting critical errors.

QMessageBox 예제

```
QMessageBox::about(this, "My Title", "This is my custom message");
```

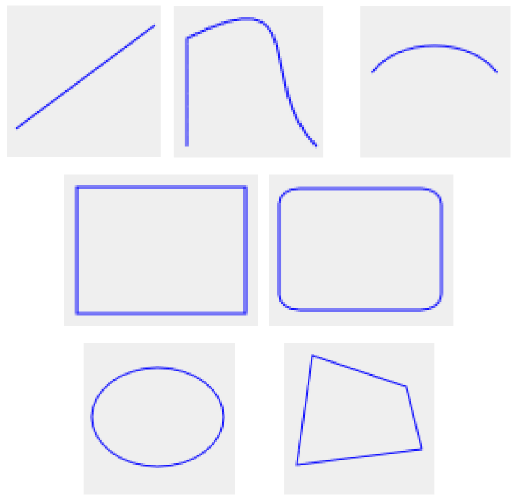


```
QMessageBox::critical(this, "My Title", "This is my custom message");
```



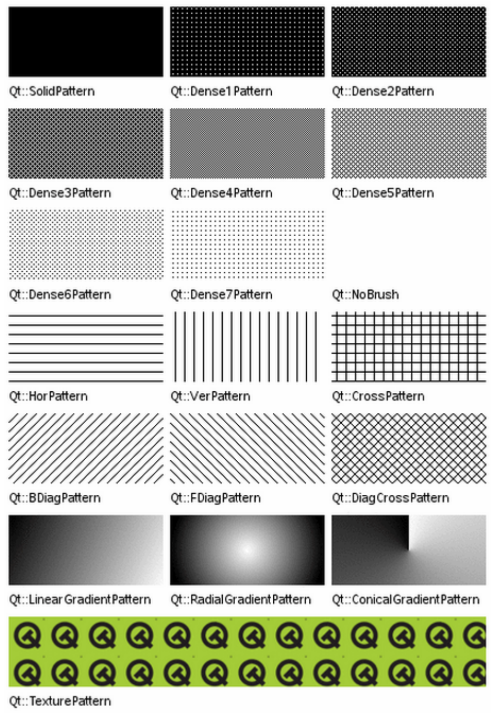
Classes for Painting

- QPainter
  - Performs low-level painting on widgets and other paint devices
  - It can draw from simple lines to complex shapes
    - drawPoint()
    - drawLine(), drawPath(), drawArc()
    - drawRect(), drawRoundRect()
    - drawEllipse()
    - drawPolygon()



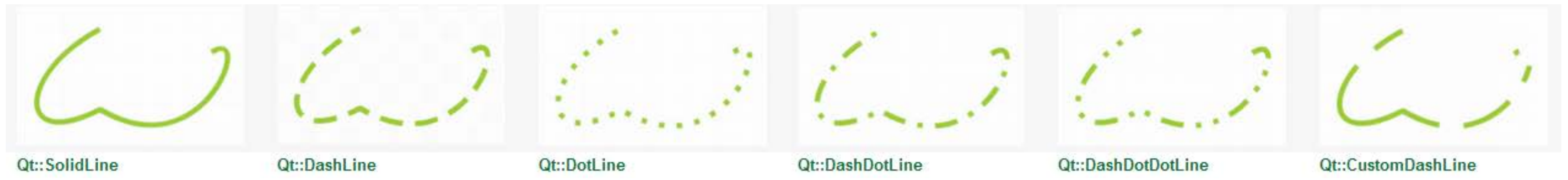
Classes for Painting

- QBrush
  - Defines the fill pattern of shapes drawn by Qpainter
    - Style: defines the fill pattern
    - Color: defines the color of the fill pattern
    - Gradient: defines the gradient fill
    - Texture: defines the pixmap

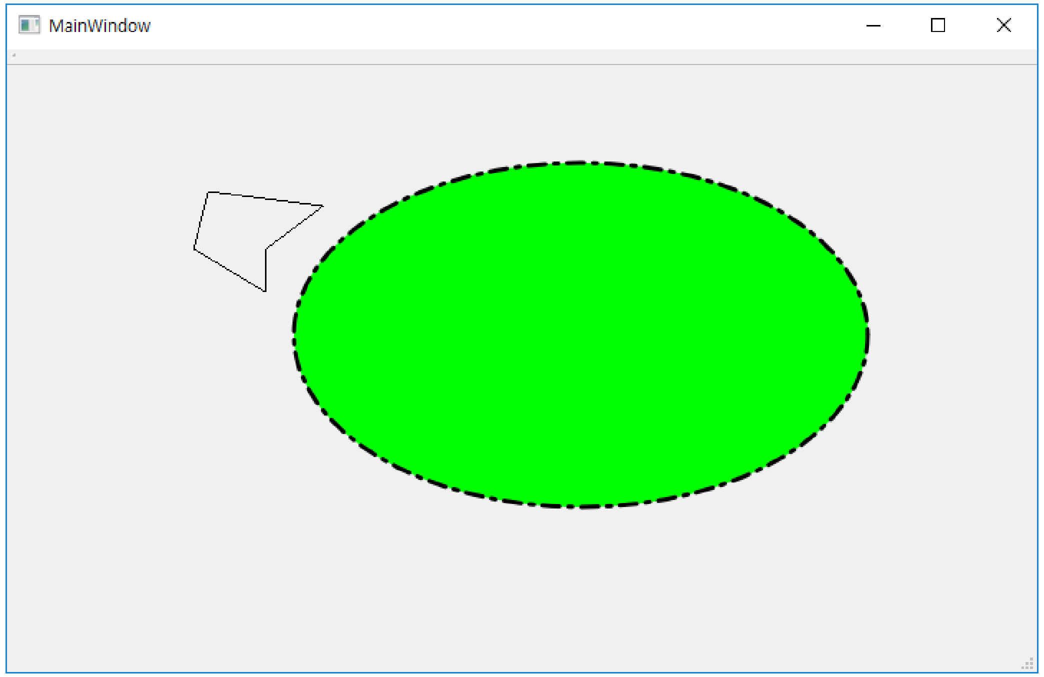


Classes for Painting

- QPen
  - Defines how a QPainter should draw lines
    - Style: defines the line type
    - Brush: fill strokes generated with the pen
    - Width: defines the line width



```cpp
#include <QMainWindow>
#include <QtGui>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

protected:
    void paintEvent(QPaintEvent *event);

private:
    Ui::MainWindow *ui;
};
```



```cpp

void MainWindow::paintEvent(QPaintEvent *event)
{
    QPainter painter(this);

    //draw a polygon
    QPolygon polygon;
    polygon << QPoint(130, 140) << QPoint(180, 170)
             << QPoint(180, 140) << QPoint(220, 110)
             << QPoint(140, 100);
     painter.drawPolygon(polygon);

     //draw an ellipse
     painter.setRenderHint(QPainter::Antialiasing, true);
     painter.setPen(QPen(Qt::black, 3, Qt::DashDotLine, Qt::RoundCap));
     painter.setBrush(QBrush(Qt::green, Qt::SolidPattern));
     painter.drawEllipse(200, 80, 400, 240);

}
```

Qt Tutorial#2?

- QGraphicsItem → 동적으로 화면 내에 그리고 지우기
- 키 입력
- 충돌 체크