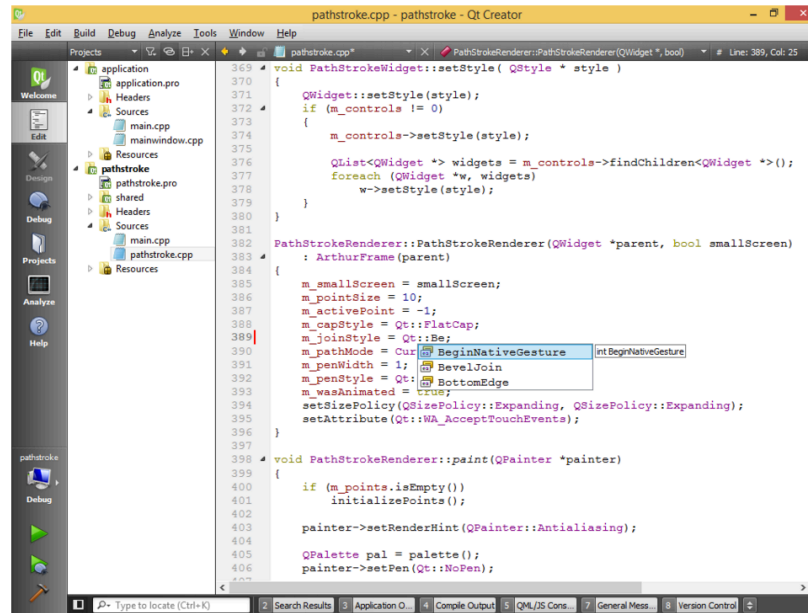


QT Programming

QT Creator (IDE)

- For qt development, use Qt Creator IDE

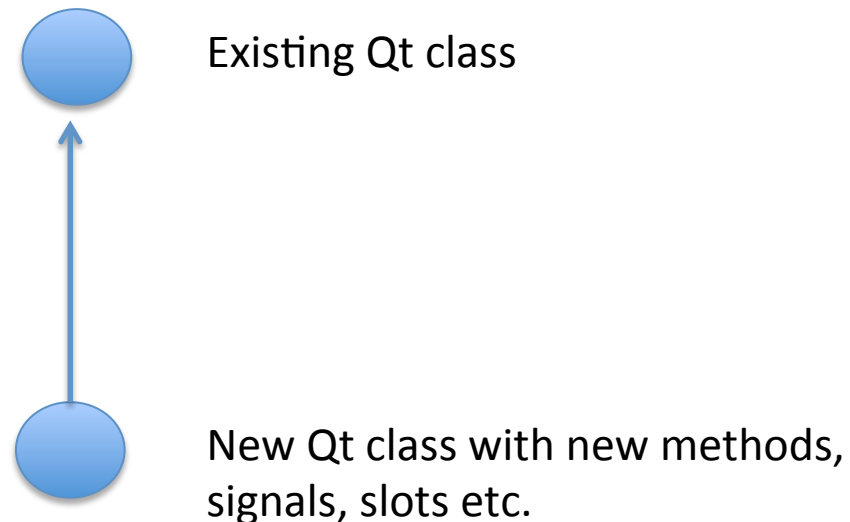
<https://www.qt.io/ide/>



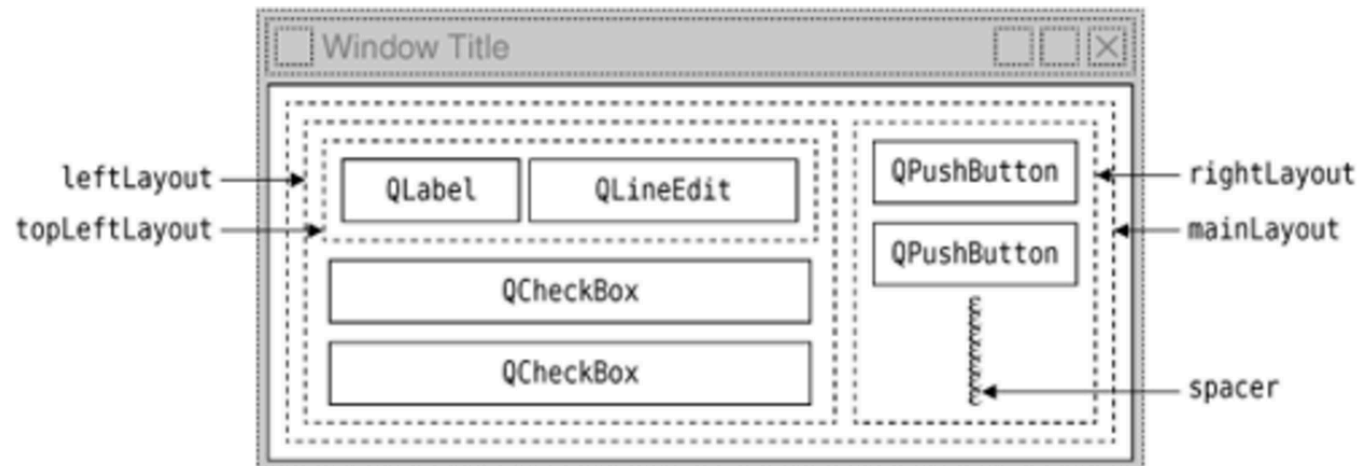
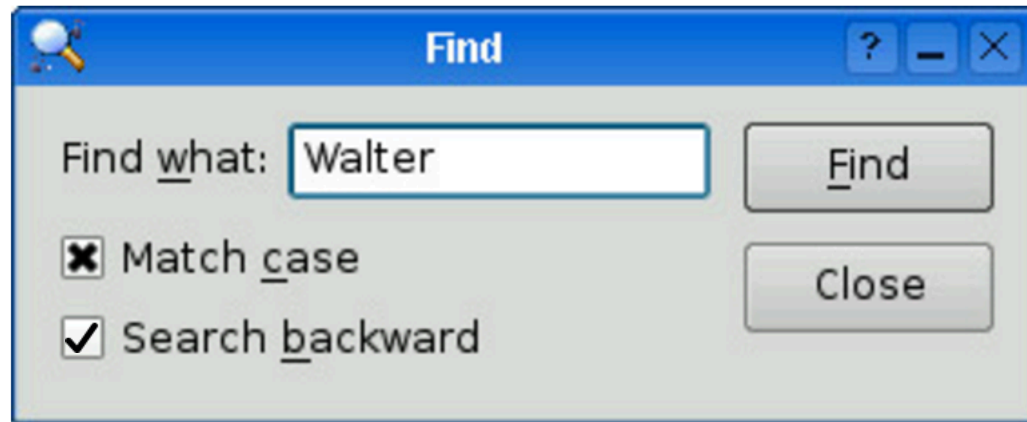
- Or alternatively, command line tools like qmake

Sub-Classing

- To customize widgets, we can pass parameters or call methods to modify appearance and behaviour.
- If these are not sufficient or are not available for specific customizations, we can subclass existing classes to create customized or new widgets.

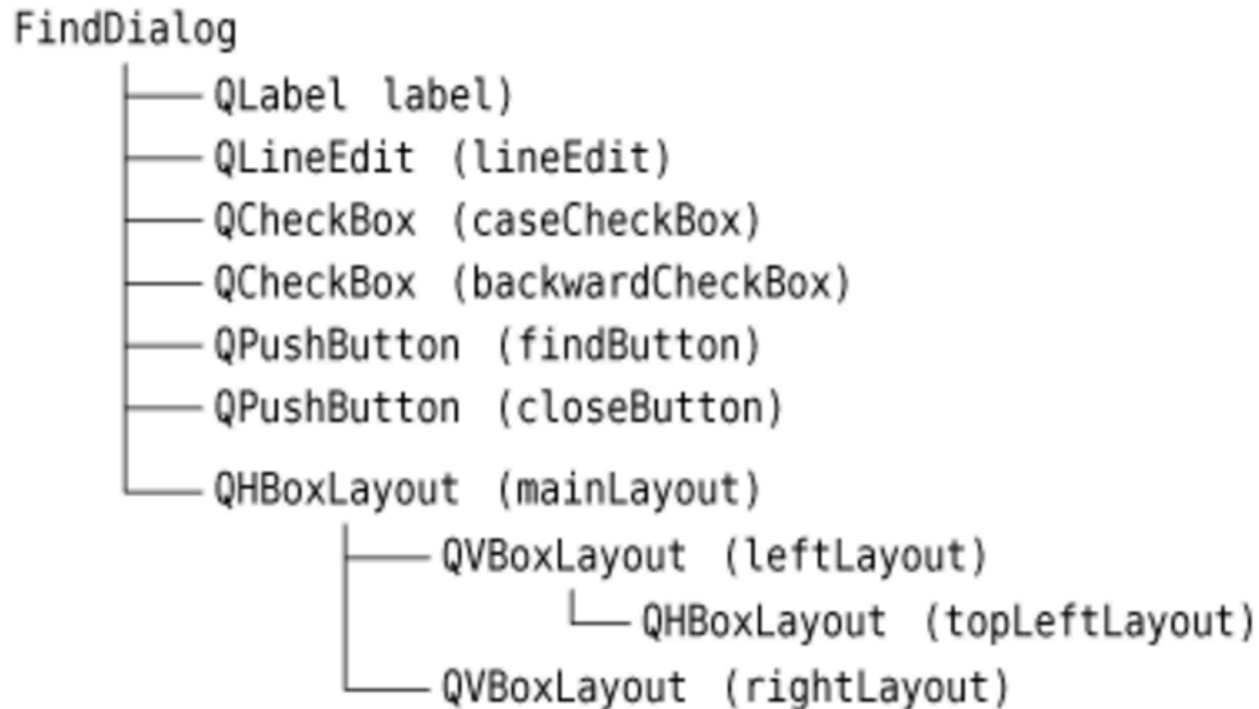


Sub-classing QDialog

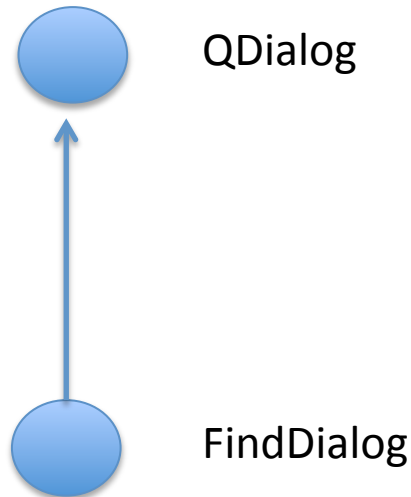


Sub-classing QDialog

- FindDialog's parent-child relationship



Sub-classing QDialog



For newly created sub-class (FindDialog), prepare two source codes files:

FindDialog.h

FindDialog.cpp

Sub-classing QDialog

- main.c (main program using FindDialog)

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

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    FindDialog *dialog = new FindDialog;
    dialog->show();
    return app.exec();
}
```

Sub-classing QDialog

finddialog.h file

```
#ifndef FINDDIALOG_H
#define FINDDIALOG_H

#include <QDialog>

class QCheckBox;
class QLabel;
class QLineEdit;
class QPushButton;

class FindDialog : public QDialog
{
    Q_OBJECT

public:
    FindDialog(QWidget *parent = 0);
```

```
signals:
    void findNext(const QString &str,
Qt::CaseSensitivity cs);
    void findPrevious(const QString &str,
Qt::CaseSensitivity cs);

private slots:
    void findClicked();
    void enableFindButton(const QString &text);

private:
    QLabel *label;
    QLineEdit *lineEdit;
    QCheckBox *caseCheckBox;
    QCheckBox *backwardCheckBox;
    QPushButton *findButton;
    QPushButton *closeButton;
};
#endif
```


Sub-classing QDialog

finddialog.cpp file

```
#include <QtGui>
#include "finddialog.h"

FindDialog::FindDialog(QWidget *parent) : QDialog(parent)
{
    label = new QLabel(tr("Find &what:"));
    lineEdit = new QLineEdit;
    label->setBuddy(lineEdit);

    caseCheckBox = new QCheckBox(tr("Match &case"));
    backwardCheckBox = new QCheckBox(tr("Search &backward"));

    findButton = new QPushButton(tr("&Find"));
    findButton->setDefault(true);
    findButton->setEnabled(false);
    closeButton = new QPushButton(tr("Close"));

    connect(lineEdit, SIGNAL(textChanged(const QString &)),
           this, SLOT(enableFindButton(const QString &)));
    connect(findButton, SIGNAL(clicked()),
           this, SLOT(findClicked()));
    connect(closeButton, SIGNAL(clicked()),
           this, SLOT(close()));
```

Sub-classing QDialog

finddialog.cpp file

```
#include <QtGui>
#include "finddialog.h"

FindDialog::FindDialog(QWidget *parent) : QDialog(parent)
{
    label = new QLabel(tr("Find &what:"));
    lineEdit = new QLineEdit;
    label->setBuddy(lineEdit);

    caseCheckBox = new QCheckBox(tr("Match &case"));
    backwardCheckBox = new QCheckBox(tr("Search &backward"));

    findButton = new QPushButton(tr("&Find"));
    findButton->setDefault(true);
    findButton->setEnabled(false);
    closeButton = new QPushButton(tr("Close"));

    connect(lineEdit, SIGNAL(textChanged(const QString &)),
           this, SLOT(enableFindButton(const QString &)));
    connect(findButton, SIGNAL(clicked()),
           this, SLOT(findClicked()));
    connect(closeButton, SIGNAL(clicked()),
           this, SLOT(close()));
```

```
QHBoxLayout *topLeftLayout = new QHBoxLayout;
topLeftLayout->addWidget(label);
topLeftLayout->addWidget(lineEdit);

QVBoxLayout *leftLayout = new QVBoxLayout;
leftLayout->addLayout(topLeftLayout);
leftLayout->addWidget(caseCheckBox);
leftLayout->addWidget(backwardCheckBox);

QVBoxLayout *rightLayout = new QVBoxLayout;
rightLayout->addWidget(findButton);
rightLayout->addWidget(closeButton);
rightLayout->addStretch();

QHBoxLayout *mainLayout = new QHBoxLayout;
mainLayout->addLayout(leftLayout);
mainLayout->addLayout(rightLayout);
setLayout(mainLayout);

setWindowTitle(tr("Find"));
setFixedHeight(sizeHint().height());
}
```

Sub-classing QDialog

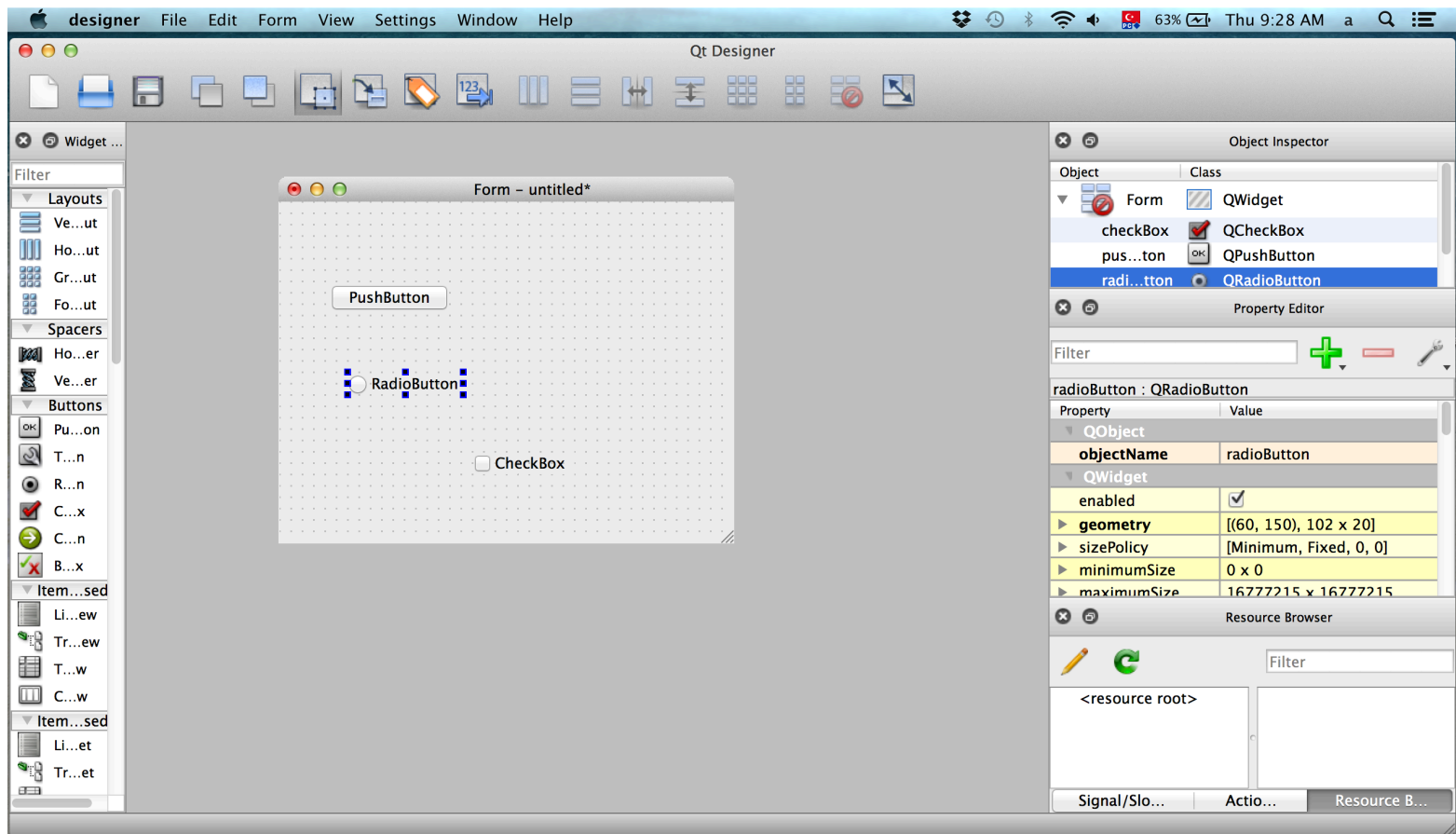
finddialog.cpp file

```
void FindDialog::findClicked()
{
    QString text = lineEdit->text();
    Qt::CaseSensitivity cs =
        caseCheckBox->isChecked() ? Qt::CaseSensitive
                                   : Qt::CaseInsensitive;
    if (backwardCheckBox->isChecked()) {
        emit findPrevious(text, cs);
    } else {
        emit findNext(text, cs);
    }
}

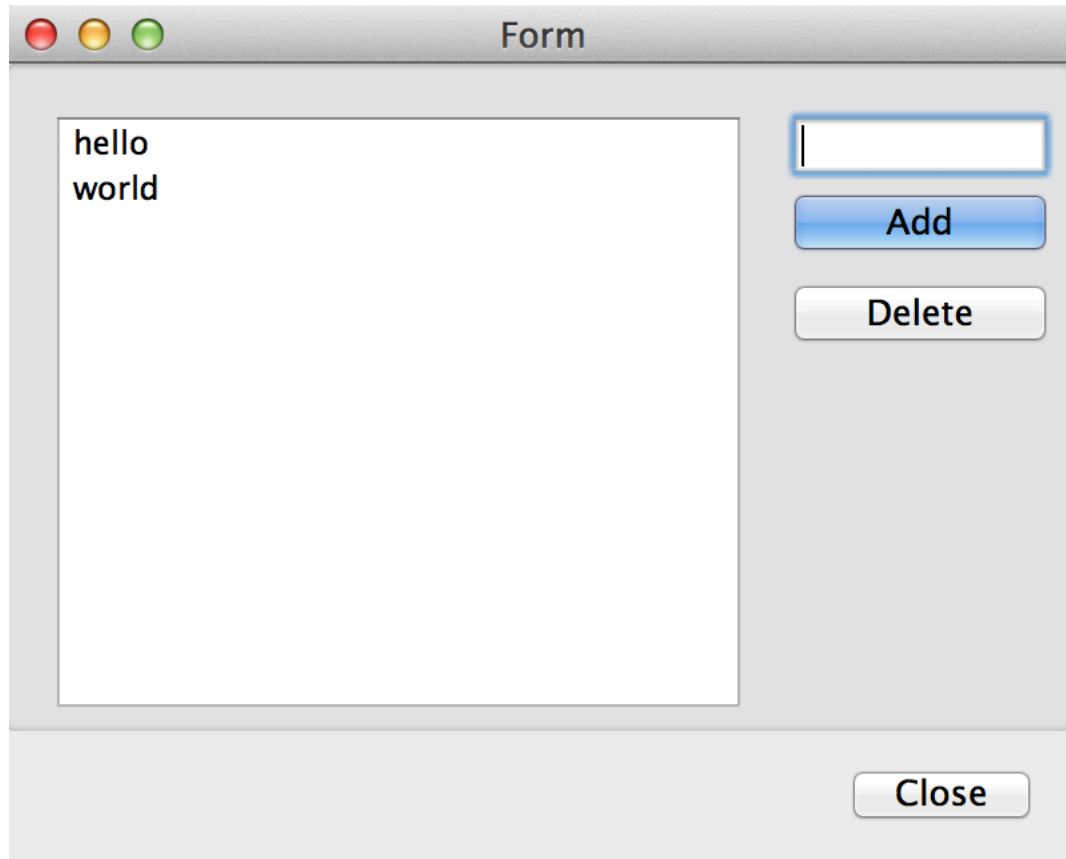
void FindDialog::enableFindButton(const QString &text)
{
    findButton->setEnabled(!text.isEmpty());
}
```

QT Designer

- Qt Designer is Qt's tool for designing and building graphical user interfaces (GUIs) from Qt components.
- You can compose and customize your widgets or dialogs in a what-you-see-is-what-you-get (WYSIWYG) manner, and test them using different styles and resolutions.



Form Designed with QT Designer



The image shows a Qt Designer window titled "Form". The window has a standard macOS-style title bar with red, yellow, and green window control buttons. The main content area is divided into two sections. On the left is a large text area containing the text "hello" and "world" on two lines. On the right is a vertical stack of controls: a text input field, a blue "Add" button, and a "Delete" button. At the bottom right of the window is a "Close" button.

Form

hello
world

Add

Delete

Close