Pokročilé programování v jazyce C pro chemiky

(C3220)

Knihovna Qt – část 2

Program rozdělený do několika souborů

- Zdrojový kód programů v C++ obvykle rozdělujeme do několika souborů tak aby každá větší třída byla umístěna v samostatném souboru **.cpp*
- Název souboru volíme tak aby bylo zřejmé jakou třídu obsahuje, obvykle se shoduje s názvem třídy (např. *application.cpp*, *graphicwidget.cpp*)
- Ke každému souboru **.cpp* vytvoříme hlavičkový soubor se stejným jménem ale koncovkou *.h*
- Funkci `main()` je vhodné umístit do samostatného souboru *main.cpp*
- Pravidla pro překlad jednotlivých souborů umístíme do souboru *Makefile*
- Při práci s knihovnou Qt je *Makefile* generován pomocí nástroje *qmake*: nejdříve vygenerujeme soubor projektu (*qmake -project*) potom odpovídající *Makefile* (*qmake project_name.pro*). Toto musíme opakovat pokaždé když přidáme další soubor se zdrojovým kódem nebo vložíme nový hlavičkový soubor do některého souboru **.cpp*

Vložení hlavičkových souborů

- Hlavičkové soubory vkládáme na začátek souboru *.*cpp* pomocí direktivy předprocesoru `#include`
- Nejdříve vkládáme hlavičkové soubory knihoven, potom teprve hlavičkové soubory programu
- Hlavičkové soubory knihoven uvádíme v lomených závorkách `<>`, tyto soubory budou hledány v systémových adresářích
- Hlavičkové soubory programu uvádíme v uvozovkách `""`, tyto soubory budou hledány v adresáři kde se nachází zdrojový soubor *.*cpp*
- Vkládáme vždy pouze hlavičkové soubory těch tříd, které v daném souboru používáme

```
/********************  Soubor  application.cpp   ********************/
#include <iostream>
#include <QApplication>
#include <QHBoxLayout>
#include <QPushButton>
#include <QVBoxLayout>
#include <QWidget>

#include "graphicwidget.h"
#include "application.h"

using namespace std;
```

Struktura hlavičkového souboru

- Aby nemohlo dojít k zacyklení vkládání hlavičkových souborů, definujeme pomocí direktivy `#define` symbolickou konstantu odvozenou vhodným způsobem ze jména souboru, pomocí podmínky `#ifndef` zajistíme, že překlad se uskuteční pouze tehdy pokud ještě nebyl soubor vložen
- Na začátek hlavičkového souboru musíme vložit hlavičkové soubory tříd jejichž jména jsou v hlavičkovém souboru použita
- Do hlavičkového souboru umisťujeme zejména definice tříd

```
/********************  Soubor  application.h   ********************/
#ifndef APPLICATION_H
#define APPLICATION_H

#include <QApplication>
#include "graphicwidget.h"

class Application : public QApplication
{
  // Zde budou uvedeny cleny tridy
};

#endif
```

Makro Q_OBJECT

- U tříd knihovny Qt odvozených z `QObject` a jejich potomků (tj. např. `QApplication`, `QWidget`) vložíme na začátek definice třídy makro `Q_OBJECT` , které je nezbytné pro zajištění některých specifických vlastností knihovny Qt

```
// Ukazka definice tridy Application v souboru application.h

class Application : public QApplication
{
  Q_OBJECT

  public:
    Application(int &argc, char *argv[]);
    virtual ~Application();
    int run();

  private:
    QWidget* mainWindow;
    GraphicWidget* graphicWidget;
};
```

Interaktivní prvky v knihovně Qt

- Knihovna Qt obsahuje různé interaktivní prvky ("widgety"), které slouží pro ovládání programu uživatelem
- Pro každý interaktivní prvek existuje v Qt knihovně příslušná třída, např.:

  QPushButton – tlačítko po jehož stisnutí myší se vykoná specifikovaná operace (tlačítko obsahuje textový popisek)

  QToolButton – také tlačítko, ale místo textu obsahuje obrázek (používá se hlavně v nástrojových lištách)

  QCheckBox – prvek se dvěma stavy (vybrán / nevybrán)

  QRadioButon – prvek se dvěma stavy, používá se ve skupině několika těchto prvků z nichž je vybrán vždy jen jeden

Interaktivní prvky v knihovně Qt

- Další interaktivní prvky knihovny Qt:
  - QLabel – textový popisek
  - QListView – seznam textových položek
  - QTreeView – hiearchicky uspořádaný seznam
  - QComboBox – políčko které po kliknutí zobrazí seznam položek ze kterého lze vybírat
  - QLineEdit – políčko s jednořádkovým editovatelným textem
  - QTextEdit – políčko s víceřádkovým editovatelným textem
  - QSpinBox – políčko pro specifikaci číselné hodnoty
  - QScrollBar – posuvník (vodorovný nebo svislý)

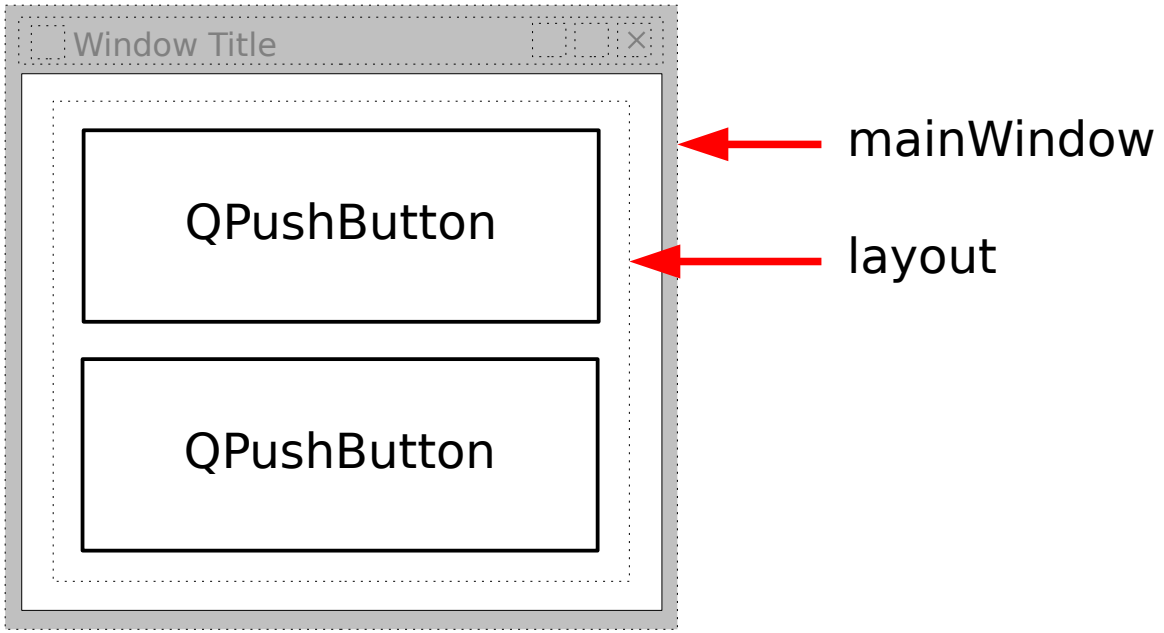
- Úplný seznam lze nalézt na:

http://doc.qt.io/qt-5/qtwidgets-index.html

http://www.informit.com/articles/article.aspx?p=1405224&seqNum=6

Rozvržení prvků v okně

- Pro automatické rozmístění interaktivních prvků v okně používáme objekty tříd `QHBoxLayout` a `QVBoxLayout`
- `QHBoxLayout` rozmisťuje objekty horizontálně, `QVBoxLayout` je rozmisťuje vertikálně
- Objekt typu `QHBoxLayout` nebo `QHBoxLayout` potom přiřadíme do okna metodou `setLayout()` třídy `QWidget`
- Pro přidávání widgetů do objekt typu `QHBoxLayout` nebo `QHBoxLayout` používáme metodu `addWidget()`
- Objekt typu `QHBoxLayout` nebo `QVBoxLayout` zajistí nastavení pozice a velikost prvků a také mezery mezi nimi



Rozvržení prvků v okně – příklad 1

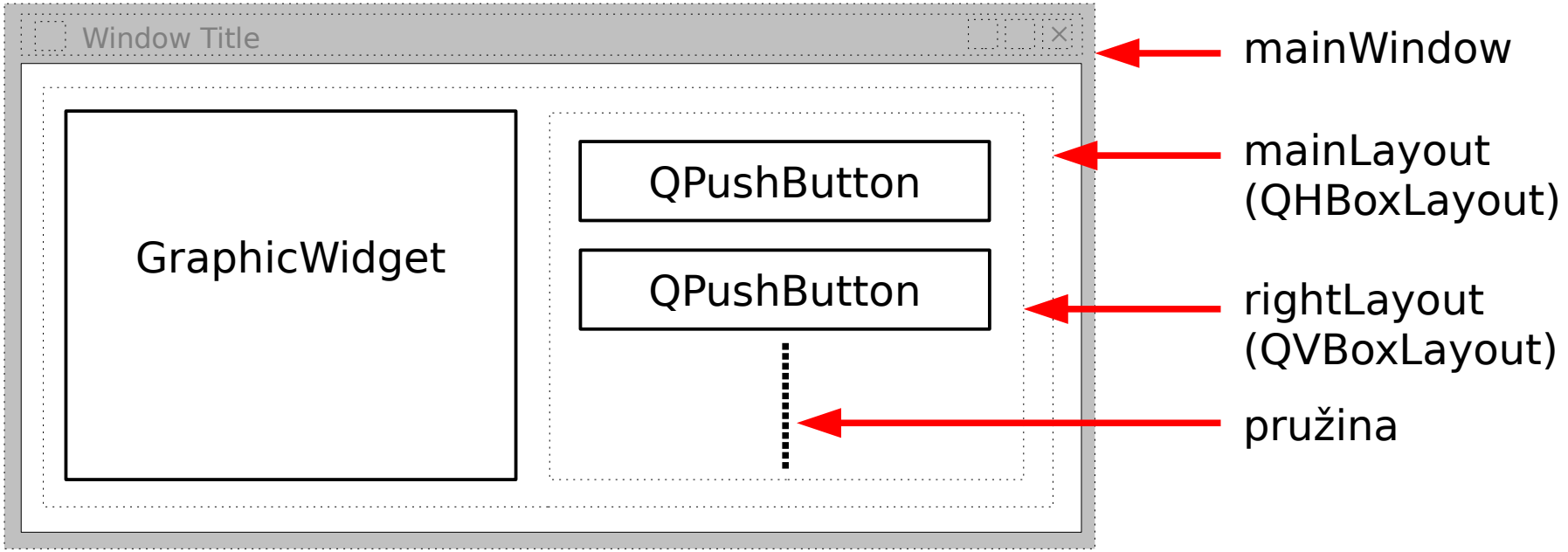
```
// Program vytvori okno a do nej budou vlozeny dve tlacitka
// usporadana vertikalne nad sebou pomoci objektu QVBoxLayout
mainWindow = new QWidget;
mainWindow->setWindowTitle("Program vytvoreny v Qt!");

  // Vytvorime dve tlacitka
QPushButton* button1 = new QPushButton("Button 1");
QPushButton* button2 = new QPushButton("Button 2");
  // Vytvorime objekt QVBoxLayout ktery bude rozmistovat tlacitka
  // vertikalne nad sebou, pozice a velikost se nastavi
  // automaticky
QVBoxLayout *layout = new QVBoxLayout();
  // Tlacitka pridame pomoci metody addWidget()
layout->addWidget(button1);
layout->addWidget(button2);

  // Do hlavniho okna nastavime prislusny layout
mainWindow->setLayout(layout);
```

Rozvržení prvků v okně

- Objekty typu `QHBoxLayout` nebo `QVBoxLayout` lze vkládat do sebe pomocí medody `addLayout()`
- Kombinací objektů typu `QHBoxLayout` nebo `QVBoxLayout` můžeme vytvořit i složitější rozmístění objektů
- Při roztažení okna jsou objekty rozmisťovány tak aby byly centrované a mezery mezi nimi proporcionální, toto chování lze ovlivnit vložením "pružiny" metodou `addStretch()`



Rozvržení prvků v okně – příklad 2

```
// Program vytvori okno a do nej bude vlozen widget
// typu GraphicWidget a na pravo budou dve tlacitka nad sebou
mainWindow = new QWidget;
mainWindow->setWindowTitle("Program vytvoreny v Qt!");

graphicWidget = new GraphicWidget;
  // Pro objekt test_widget nastavime minimalni velikost
graphicWidget->setMinimumSize(300, 350);
QPushButton *buttonHide = new QPushButton("Hide");
QPushButton *buttonShow = new QPushButton("Show");

QVBoxLayout *rightLayout = new QVBoxLayout;
rightLayout->addWidget(buttonHide);
rightLayout->addWidget(buttonShow);
  // Pod tlacitka pridame pruzinu
rightLayout->addStretch();

QHBoxLayout *mainLayout = new QHBoxLayout;
mainLayout->addWidget(graphicWidget);
mainLayout->addLayout(rightLayout);

  // Do hlavniho okna nastavime prislusny layout
mainWindow->setLayout(mainLayout);

mainWindow->show();
```

Komunikace mezi objekty v Qt

- Ke komunikaci mezi objekty v Qt knihovně slouží systém tzv. sinálů a slotů (*signals and slots*)
- Signály a sloty se používá nejčastěji pro zaslání informace od interaktivního objektu (např. informace o stisknutí tlačítka) do jiného objektu (hlavního okna nebo jiného widgetu)
- Signál je metoda vytvořená v objektu od něhož signál pochází
- Slot je metoda, kterou vytvoříme ve třídě která bude zpracovávat zaslaný signál
- Signály a sloty jsou ve třídách deklarovány ve speciálních sekcích označených `signals` a `slots`
- Ve třídě `QPushButton` je definován signál `clicked()`, který je generován po stisknutí tlačítka
- Propojení mezi zaslaným signálem a slotem provedeme pomocí funkce connect:

```
QObject::connect(object1, signal, object2, slot);
```

- Podrobnější popis se nachází na:
  http://doc.qt.io/qt-5/signalsandslots.html

Komunikace mezi objekty v Qt – příklad

```
// Dklarace slotu ve tride GraphicWidget v souboru graphicwidget.h
class GraphicWidget : public QWidget
{
  Q_OBJECT
  private slots:
    // Nasledujici metody slotu budou volany po zmacknuti
    // prislusnych tlacitek Hide a Show
    void hideGraphic();
    void showGraphic();
  // Deklarace dalsich clenu tridy
};
```

```
  // Program ukazuje propojeni mezi signalem clicked() od dvou
  // tlacitek se sloty v objektu tridy GraphicWidget

  graphicWidget = new GraphicWidget;
  QPushButton* buttonHide = new QPushButton("Hide");
  QPushButton* buttonShow = new QPushButton("Show");

    // Signal clicked() z tlacitka buttonHide zpusobi volani metody
    // hideGraphic() definovane ve tride GraphicWidget
  QObject::connect(buttonHide, SIGNAL(clicked()),
    graphicWidget, SLOT(hideGraphic()));
    // Podobne pro tlacitko buttonShow bude volana metoda
    // showGraphic() definovana ve tride GraphicWidget
  QObject::connect(buttonShow, SIGNAL(clicked()),
    graphicWidget, SLOT(showGraphic()));
```

Definice slotu

- Vhodné signály jsou zpravidla již předdefinované ve třídách Qt knihovny, většinou potřebujeme definovat pouze sloty
- Metody slotu obsahují kód reagující na signál

```
// Definice metody slotu, která je zavolána po stisknutí
// tlacitka buttonHide

void GraphicWidget::hideGraphic()
{
    // Vypiseme informaci o stisknuti tlacitka na terminal
  cout << "Bylo stisknuto tlacitko Hide" << endl;

    // Do promenne displayRectangle priradime hodnotu false
    // indikujici ze obdelnik nema byt vykreslovan
  displayRectangle = false;

    // Vyvolame pozadavek na prekresleni okna metodou update()
  update();
}
```

Dialogová okna v knihovně Qt

- V knihovně Qt můžeme vytvářet dialogová okna, která odvozujeme ze třídy `QDialog`
- V knihovně je předdefinováno několik nejčastěji používaných dialogových oken:

  QFileDialog – dialogové okno pro výběr souboru nebo adresáře

  QColorDialog – dialogové okno pro výběr barvy

  QFontDialog – dialogové okno pro výběr fontu

  QMessageBox – dialogové okno pro zobrazení textové informace uživateli

  QInputDialog – dialogové okno pro získání textové nebo číselné hodnoty od uživatele
- Úplný seznam se nachází na:
  http://doc.qt.io/qt-5/dialogs.html

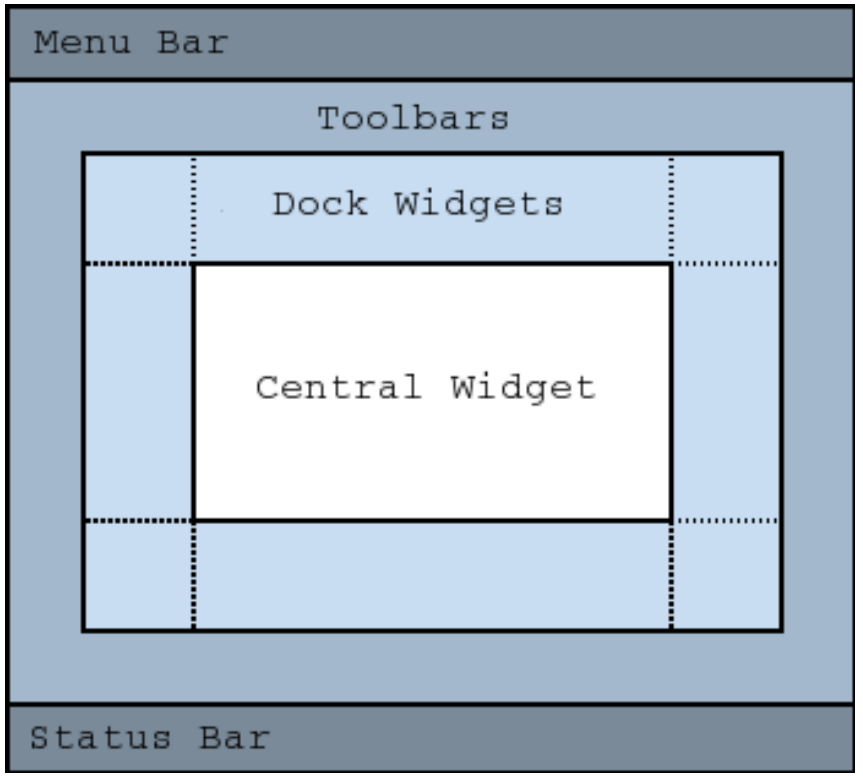
Dialogové okno pro výběr souboru

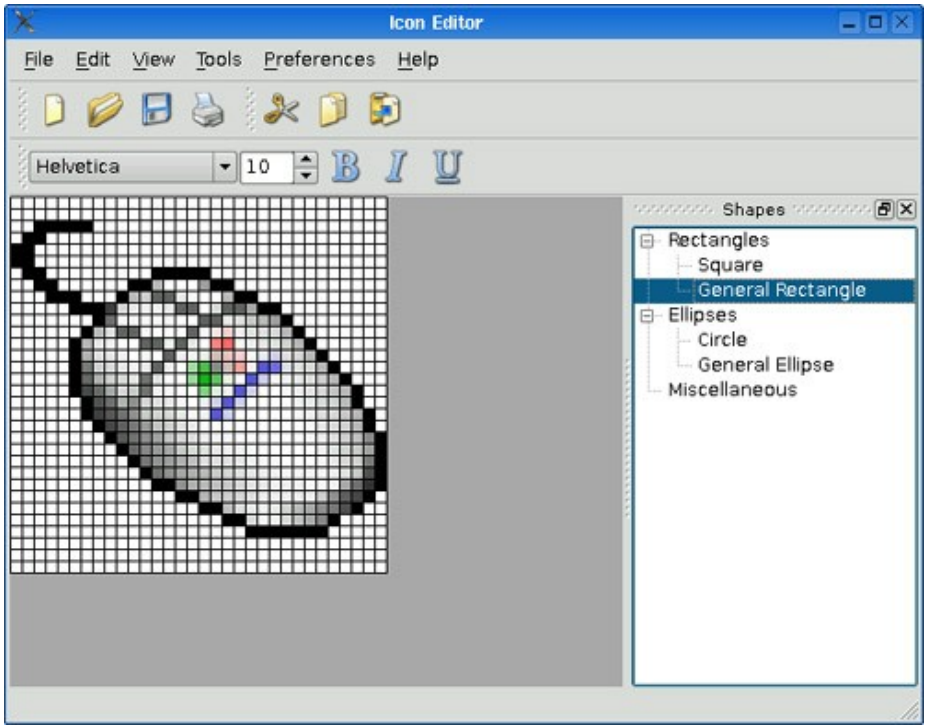
- Pro jednoduchou práci s dialogovými okny jsou v Qt knihovně předdefinovány statické metody které automaticky vytvoří příslušný objekt dialogového okna a okno zobrazí
- Dialogové okno pro vybrání souboru lze otevřít metodou `QFileDialog::getOpenFileName()`, která vrátí jméno souboru

```
  // Nasledujici metoda je zavolana po stisknuti tlacitka pro
  // otevreni souboru
void GraphicWidget::openFile()
{
    // Knihovna Qt pouziva pro retezce tridu
    // QString misto string
  QString fileName;
    // Dialogove okno pro vyber souboru otevreme nasledujici metodou,
    // ktera vrati jmeno souboru jako retezec typu QString
  fileName = QFileDialog::getOpenFileName(this, "Open", ".");
    // Pokud nebylo vybrano jmeno souboru, je retezec prazdny
  if (fileName.isEmpty()) return;
    // Standardni vystupni proudy umi pracovat jen s promennymi typu
    // string, na ktere musime konvertovat promennou fileName ktera
    // je typu QString
  string strFileName;
  strFileName = fileName.toLatin1().constData();
  cout << "Jmeno souboru: " << strFileName << endl;
}
```

Třída `QMainWindow`

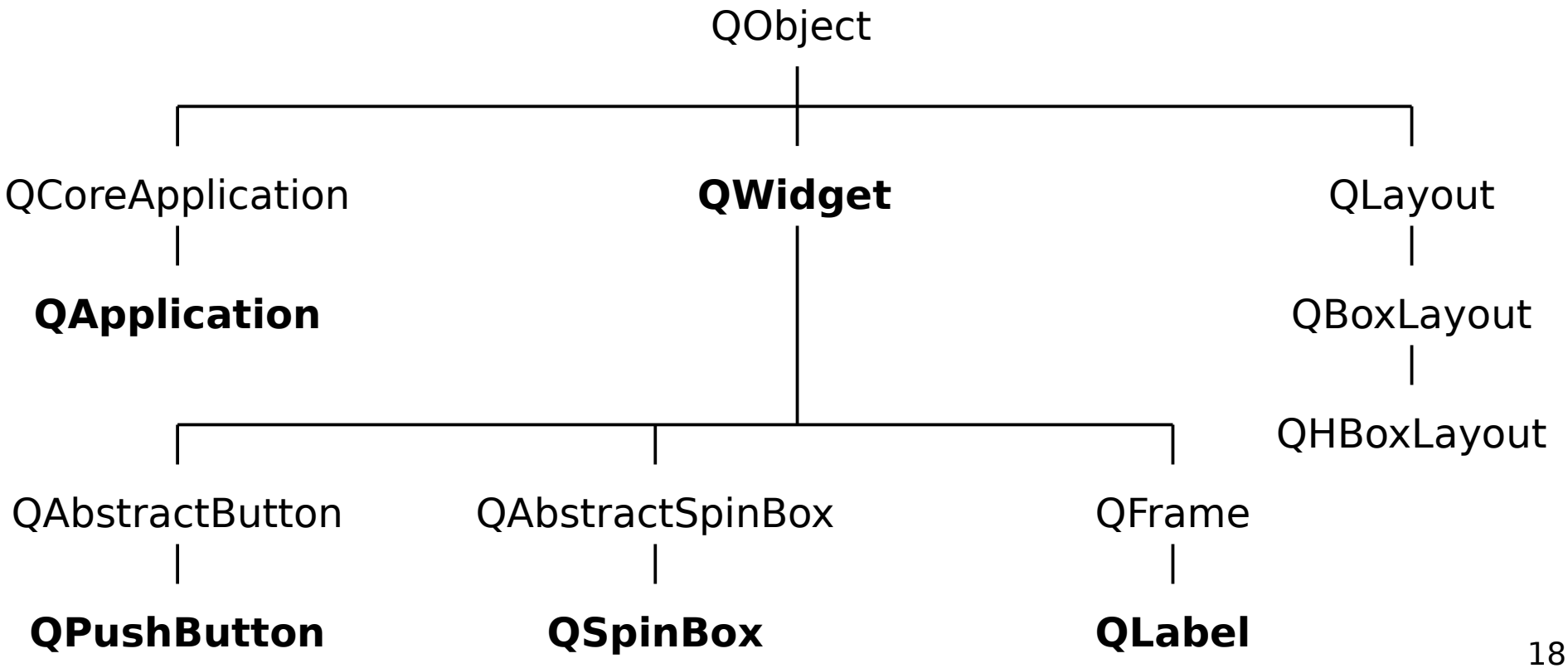
- Hlavní okno aplikace se v knihovně Qt obvykle odvozuje ze třídy `QMainWindow`, která poskytuje možnost pro vytváření menu, nástrojových lišt, informační lišty a další pokročilé funkce
- Více informací: http://doc.qt.io/qt-5/qmainwindow.html





Hierarchie tříd v knihovně Qt

- Třídy v Qt knihovně jsou zpravidla odvozeny z jiných tříd a vytváří tak hirerarchickou strukturu
- Třídy odvozené z `QObject` jsou schopny komunikace prostřednictvím systému signálů a slotů
- Podrobnější informace na:
  http://doc.qt.io/qt-5/hierarchy.html



Dodržujte následující pravidla

- V hlavičkovém souboru vždy použijte direktivy uvedené v sekci "Struktura hlavičkového souboru".
- Na začátek souborů **.cpp* vložte vždy jen hlavičkové soubory s těmi třídami, které v daném souboru opravdu používáte.
- Některé překladače obsahují chybu způsobující pád programu. V takovém případě je potřeba v souboru *Makefile* změnit na řádku začínajícím CXXFLAGS optimalizaci z -O2 na -O1 nebo -O
- Do adresáře s projektem ani jeho podadresářů neumísťujte žádné jiné soubory **.cpp* a **.h* než ty, které jsou pro projekt potřeba. Tyto soubory by totiž byly automaticky zahrnuty do souboru projektu (při jeho generování příkazem *qmake -project*) a byly by tedy i kompilovány.

Cvičení

1. Vytvořte program vycházející z programu z předchozího cvičení, který bude v hlavním okně obsahovat widget `GraphicWidget` a napravo dvě tlačítka s popisem *Hide* a *Show*. Po stisknutí prvního tlačítka dojde ke schování obdélníku (tj. okno se překreslí ale vykreslí se jen elipsa a čára). Po stisknutí druhého tlačítka se obdélník opět zobrazí. 3 body

2. Do programu přidejte třetí tlačítko po jehož stisknutí se zobrazí dialogové okno pro výběr souboru. Po vybrání souboru se jeho jméno vypíše na terminál a také se zobrazí v okně s grafikou. 2 body