

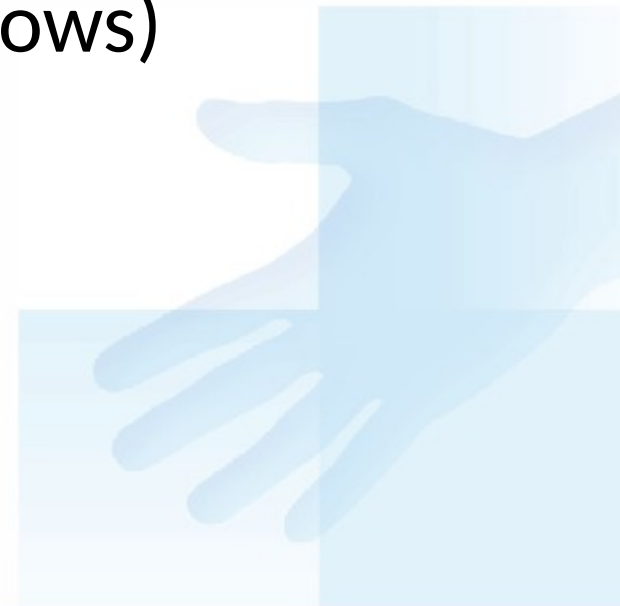
Uživatelská rozhraní



Knihovna Qt



- Trolltech (1994) v Oslu (Norsko) vytváří grafické uživatelské rozhraní (GUI) pro C++
- multi-platformová GUI C++ knihovna, určena pro vývoj aplikací (Unix/X, Windows)
- Signály a sloty



Knihovna Qt



Hlavní stránky <https://www.qt.io/>

Dokumentace <https://doc.qt.io/>



Qt Creator



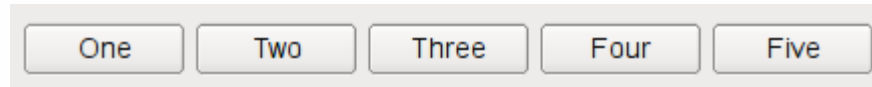
The screenshot displays the Qt Creator IDE interface. The main window shows the source code for 'converter.cpp'. The code includes various Qt widget classes and defines a 'Converter' class constructor. The constructor initializes a 'QHBoxLayout' named 'hbox' and a 'QVBoxLayout' named 'levy'. It adds a 'QRadioButtons' widget to 'hbox' and a 'QGroupBox' named 'smerGroup' to 'levy'. The 'smerGroup' contains a 'QHBoxLayout' named 'vbox' which holds the 'cToF' widget. Finally, 'inOutBox' is initialized as a 'QHBoxLayout'.

```
1  #include <QLabel>
2  #include <QLineEdit>
3  #include <QTableView>
4  #include <QHBoxLayout>
5  #include <QVBoxLayout>
6  #include <QString>
7  #include <QButtonGroup>
8  #include <QRadioButton>
9  #include <QSpinBox>
10 #include <QGroupBox>
11 #include <QPushButton>
12 #include <QMessageBox>
13
14 #include "converter.h"
15
16 Converter::Converter(QWidget *parent)
17     : QWidget(parent)
18 {
19     QHBoxLayout *hbox = new QHBoxLayout;
20
21     QVBoxLayout *levy = new QVBoxLayout;
22
23     hbox->addLayout( levy );
24
25     const QString cToFStr = QString( "C -> F" );
26     cToF = new QRadioButton( cToFStr, this );
27     cToF->setChecked( true );
28
29     QGroupBox *smerGroup = new QGroupBox(tr("Convertor direction"));
30
31     QHBoxLayout *vbox = new QHBoxLayout;
32     vbox->addWidget( cToF );
33     vbox->addStretch(1);
34     smerGroup->setLayout( vbox );
35
36     levy->addWidget( smerGroup );
37
38     QHBoxLayout *inOutBox = new QHBoxLayout;
39     //
```

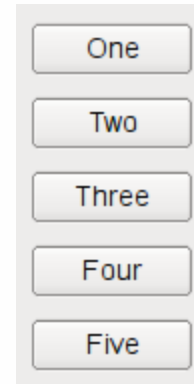
Layout



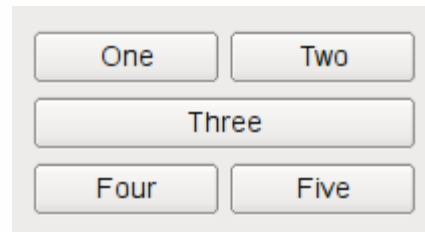
QHBoxLayout



QVBoxLayout



QGridLayout



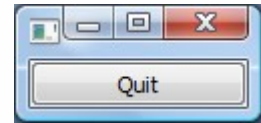
QFormLayout



<https://doc.qt.io/qt-5/layout.html>

Ukázky

```
#include <QApplication>
#include <QPushButton>
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QPushButton *button = new QPushButton("Quit");
    QObject::connect(button, SIGNAL(clicked()), &app, SLOT(quit()));
    button->show();
    return app.exec();
}
```



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

```
Widget::Widget(QWidget *parent)
```

```
    : QWidget(parent)
```

```
{
```

```
    QHBoxLayout *layout = new QHBoxLayout();
```

```
    QPushButton *btn1 = new QPushButton("Prvni");
```

```
    QPushButton *btn2 = new QPushButton("Druhy");
```

```
    connect(btn1, SIGNAL(clicked(bool)), this, SLOT(close()));
```

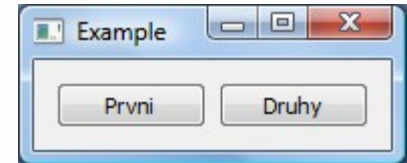
```
    connect(btn2, SIGNAL(clicked(bool)), this, SLOT(showMaximized()));
```

```
    layout->addWidget(btn1);
```

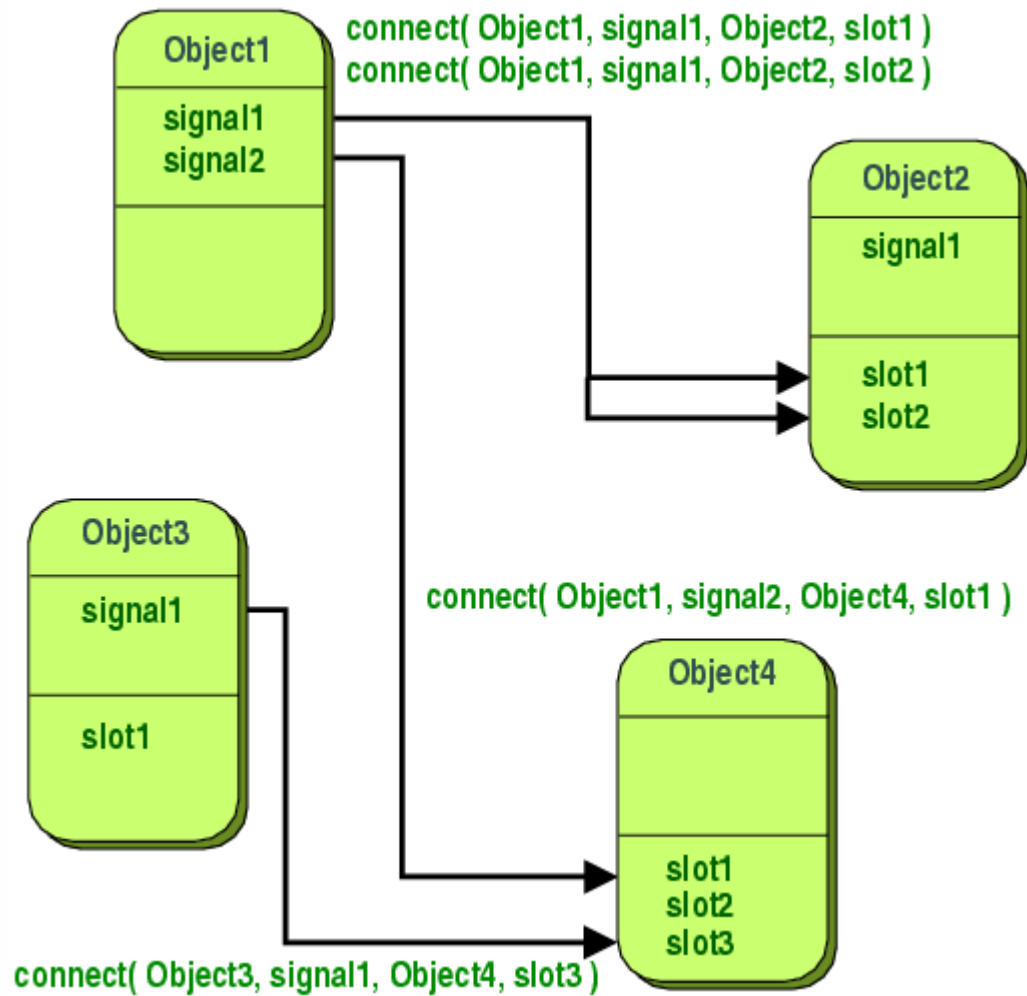
```
    layout->addWidget(btn2);
```

```
    this->setLayout(layout);
```

```
}
```



Signály a sloty



Signály a sloty



```
class Priklad1
{
public:
    Priklad1(); // konstruktor
    int hodnota() const { return _hodnota; }
    void nastavHodnotu( int );
private:
    _hodnota val;
};
```



Signály a sloty




```
class Priklad1 : public QObject {
    Q_OBJECT
public:
    Foo();
    int hodnota() const { return _hodnota; }
public slots:
    void nastavHodnotu( int );
signals:
    void hodnotaZmenena(int);
private:
    int _hodnota;
};
```



Signály a sloty



```
void Priklad1::nastavHodnotu( int h )  
{  
    if ( h != _hodnota ) {  
        _hodnota = h;  
        emit hodnotaZmenena(h);  
    }  
}  
  
//  signál: hodnotaZmenena
```



Signály a sloty



Příklad1 a, b; // definice dvou objektu dedících z QObject
//přirazení signalu

```
connect(&a, SIGNAL(hodnotaZmenena(int)), &b,  
      SLOT(nastavHodnotu(int)));
```

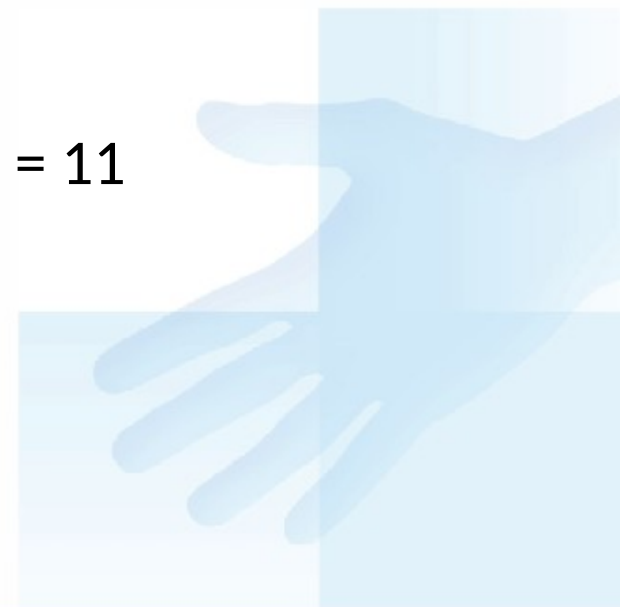
```
b.nastavHodnotu( 11 );
```

```
// a = není definováno
```

```
b = 11
```

```
a.nastavHodnotu( 79 );
```

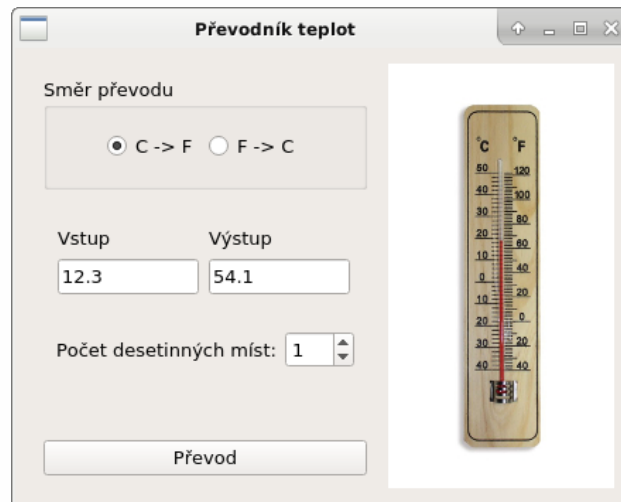
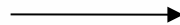
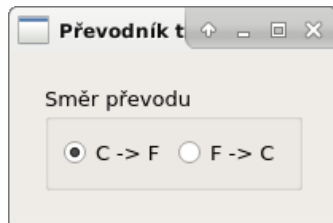
```
// a = 79    b = 79
```



Náplň cvičení:



- Vytvořte převodník teplot



Náplň cvičení:



Diagram illustrating the layout structure of a temperature converter application window titled "Převodník teplot".

The window is divided into two main sections:

- Left Panel (Control Area):** This area is managed by a **VBoxLayout** (indicated by a dashed orange border). It contains several sub-sections:
 - Směr převodu (Direction of conversion):** A **GroupBox** containing two radio buttons: C -> F and F -> C.
 - Input/Output Fields:** A **GridLayout** containing two text input fields: "Vstup" (Input) with the value "12.3" and "Výstup" (Output) with the value "54.1".
 - Decimal Places:** A **FormLayout** containing a label "Počet desetinných míst:" followed by a spin box set to "1".
 - Conversion Button:** A button labeled "Převod" (Convert).
- Right Panel (Visual Area):** This area displays a vertical thermometer graphic with two scales: Celsius (°C) on the left and Fahrenheit (°F) on the right. The Celsius scale ranges from -40 to 50, and the Fahrenheit scale ranges from -40 to 120. The current temperature shown is 12.3°C (54.1°F).

Obrázek



```
QString *imgFilename = new QString( "soubor.png" );  
QPixmap *imgPixmap = new QPixmap( *imgFilename );  
QLabel *obr = new QLabel;  
obr->setPixmap( *imgPixmap );  
hbox->addWidget( obr );
```



**Prosím, dopracujte opět funkčnost
a vzhled. Projekt si můžete
libovolně rozšířit.**

Děkuji za pozornost.

