Rechnerpraktikum zu Kapitel 3

Grafische Benutzeroberflächen

“Why didn’t we provide a GUI?”

The committee couldn’t do it. [...] GUI was seen as just another large and complex library that people [...] could write themselves (in particular, that was my view). They did. The problem today is not that there is no C++ GUI library, but that there are on the order of 25 such libraries in use.

(Bjarne Stroustrup: Evolving a language in and for the real world)

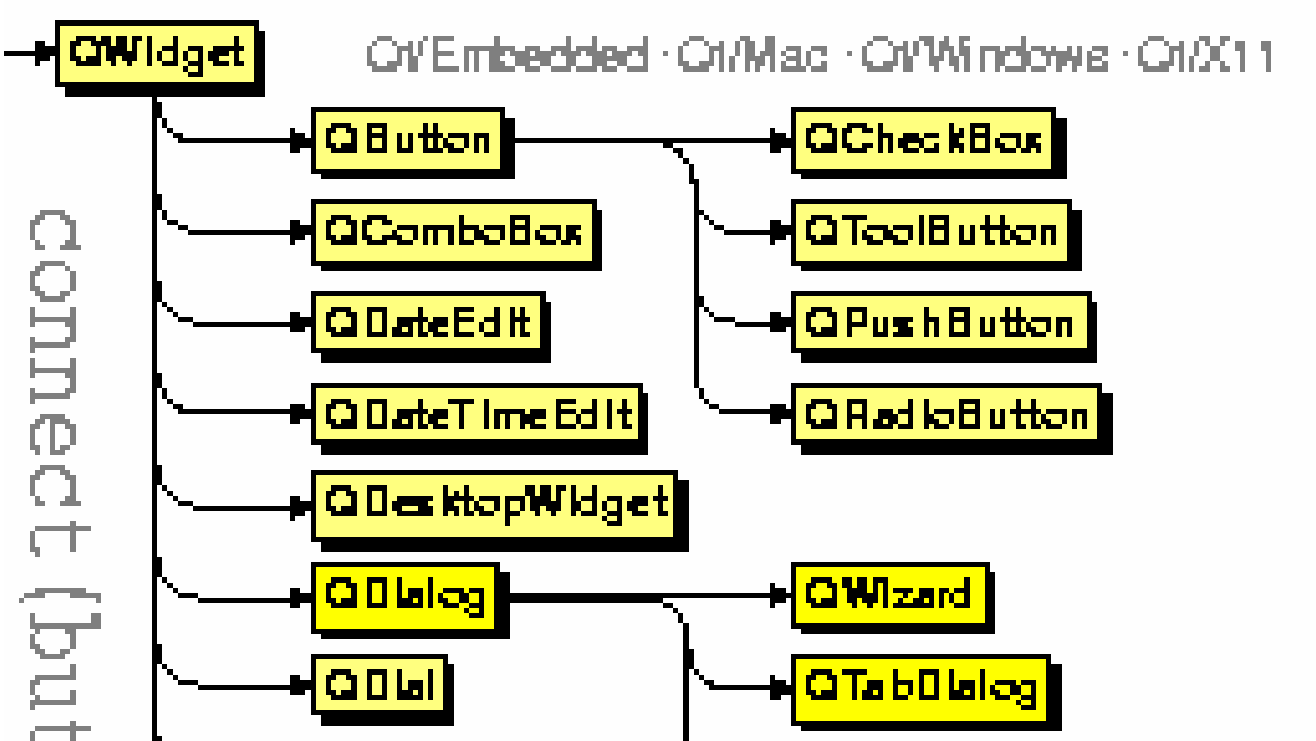
Eine dieser Bibliotheken ist Qt (vergl. http://www.qt.io)

Qt (Aussprache wie engl. “cute”) ist auf vielen Systemen, von Microsoft Windows, Mac OS X, Linux bis hin zu Smartphones verfügbar. Adobe Photoshop Elements, Mathematica, Google Earth und viele weitere Applikationen basieren auf Qt. Neben der Programmierung von Benutzeroberflächen ermöglicht Qt die Arbeit mit Datenbanken, Netzwerken und XML-Daten. Zugriffe auf parallel laufende Prozesse und Programme können ebenfalls programmiert werden.

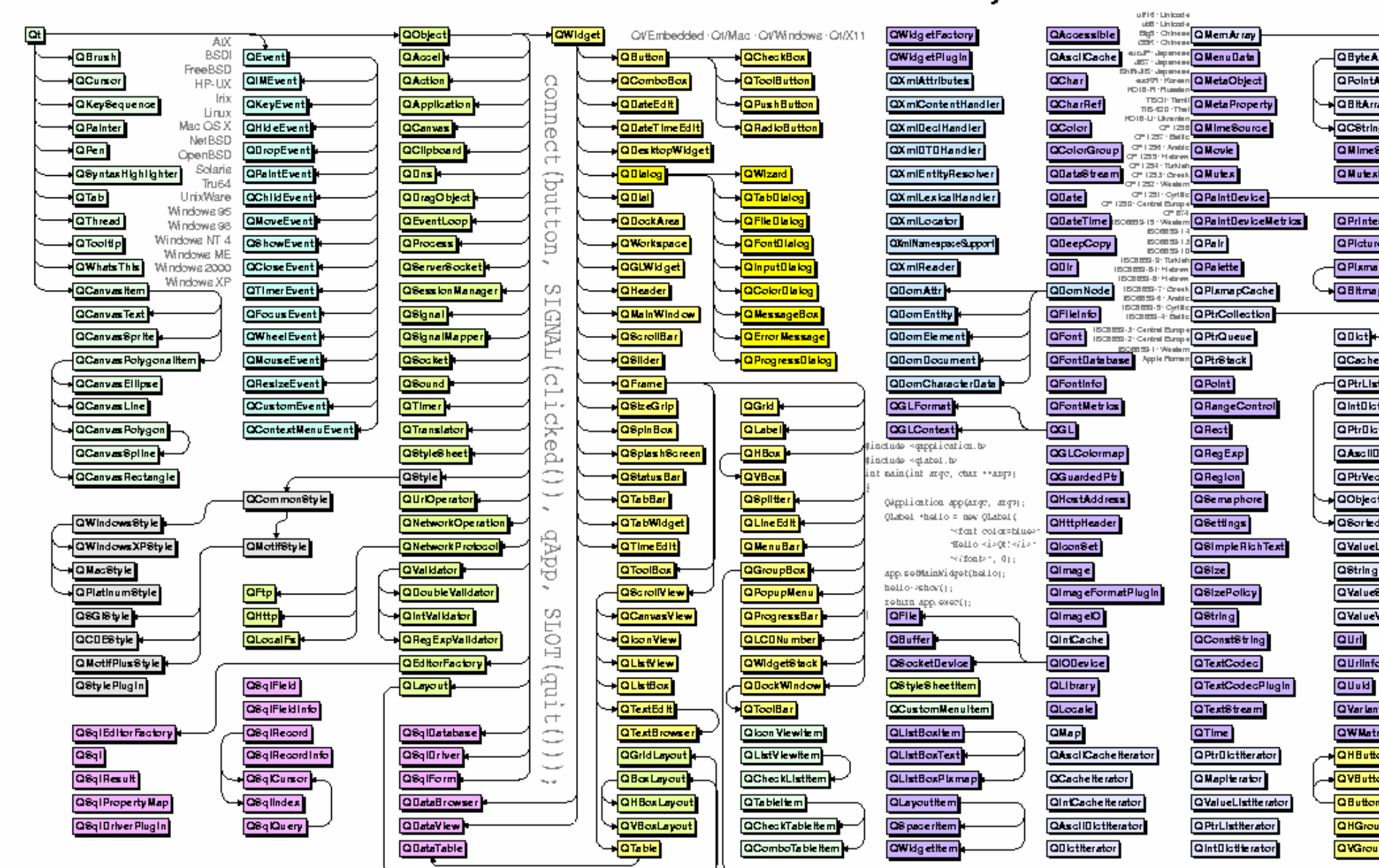
Qt-Benutzeroberflächen sind aus „Widgets“ aufgebaut.

Widgets (von engl. „window gadget“) wie Schaltflächen, Eingabefelder, Menüs, Rollleisten, Dialogfenster usw. können auf Tastatur- oder Mausereignisse reagieren und ihre grafische Darstellung auf dem Bildschirm entsprechend anpassen.

Alle Qt-Widgets sind von einer gemeinsamen Basisklasse **QWidget** abgeleitet:



The Essential Qt 3.3 Class Hierarchy



Qt is a C++ toolkit for cross-platform GUI and application development. In addition to the Qt C++ class library, the toolkit includes tools like Qt Designer that make writing applications fast and straightforward. Qt's multiplatform abilities and internationalization support means that Qt applications can reach the widest market.

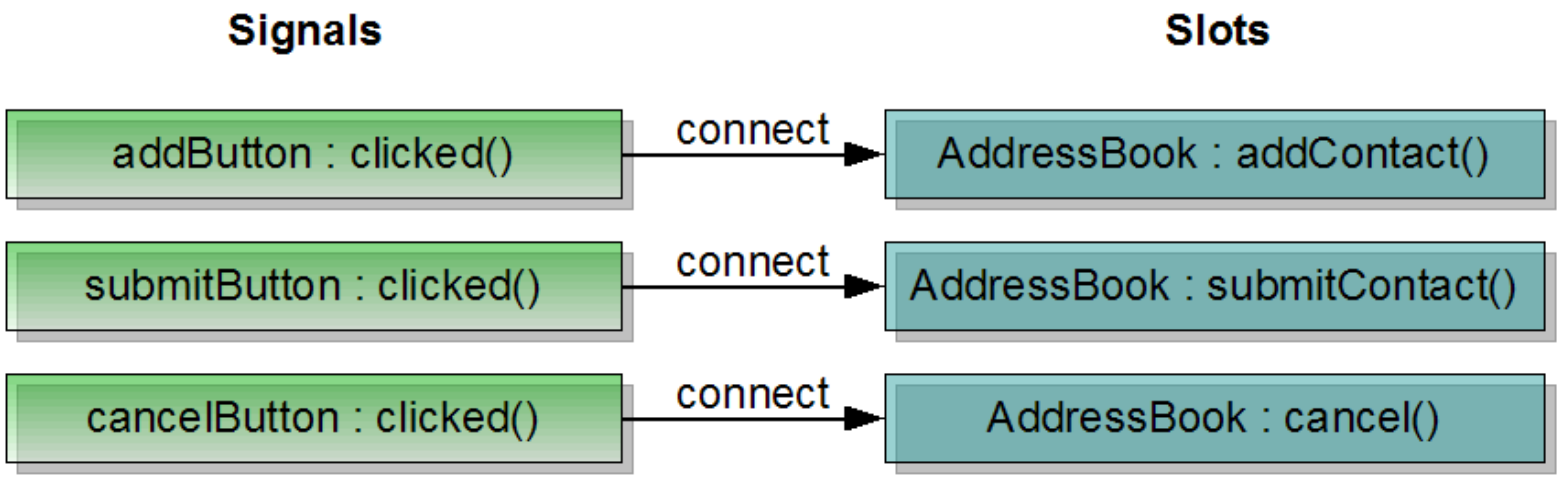
www.trolltech.com · doc.trolltech.com

Windows style · Motif style · Motif Plus style · Platinum style · SGI style · CDE style

Signals und Slots

Grafische Benutzeroberflächen werden durch Ereignisse (bei Qt **Signals** genannt) gesteuert, die zum Beispiel beim Anklicken einer Schaltfläche oder durch Mausklicks ausgelöst werden. Das Programm muss auf diese Ereignisse in der vom Anwender (!!) gewählten Reihenfolge reagieren oder ggf. auch Fehlermeldungen ausgeben.

Als Ereignisempfänger dienen spezielle Methoden (bei Qt **Slots** genannt), die mit jeweils passenden Ereignissen verknüpft sind.



(Grafik übernommen von http://doc.qt.digia.com)

P3.6. Einleitung

Rechnerpraktikum zu Kapitel 3

Grafische Benutzeroberflächen

Einleitung

Ein erstes Beispiel

Schwingkreisberechnung mit grafischer Oberfläche

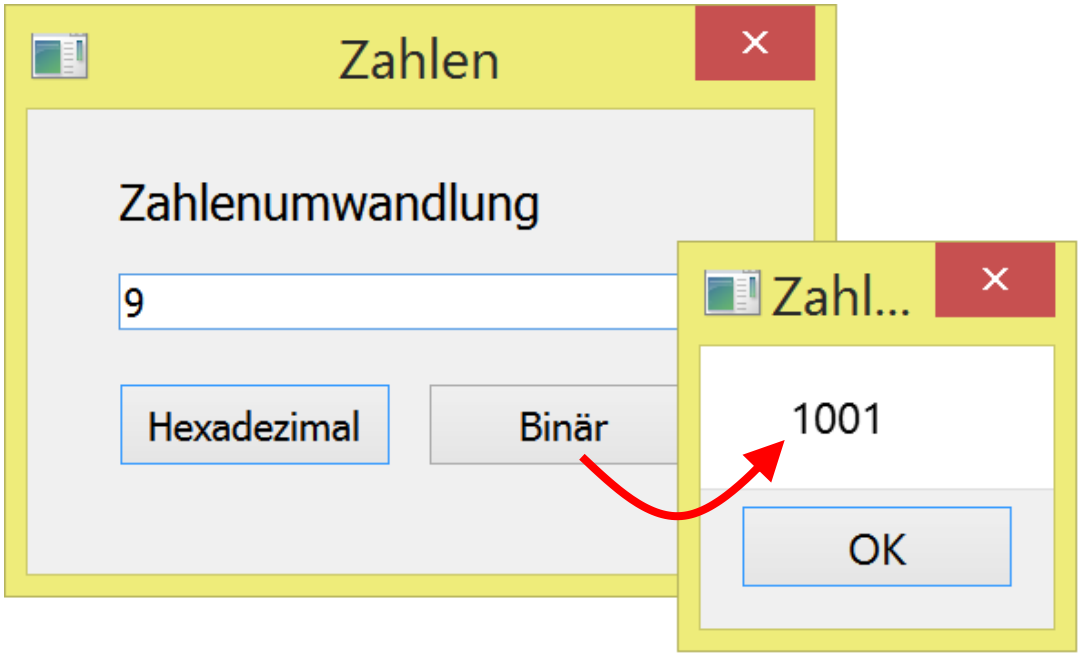
Codeschloss mit Zustandsautomat

Verkehrsampel mit Timer

Ausgabe von Binär- und Hexadezimalzahlen

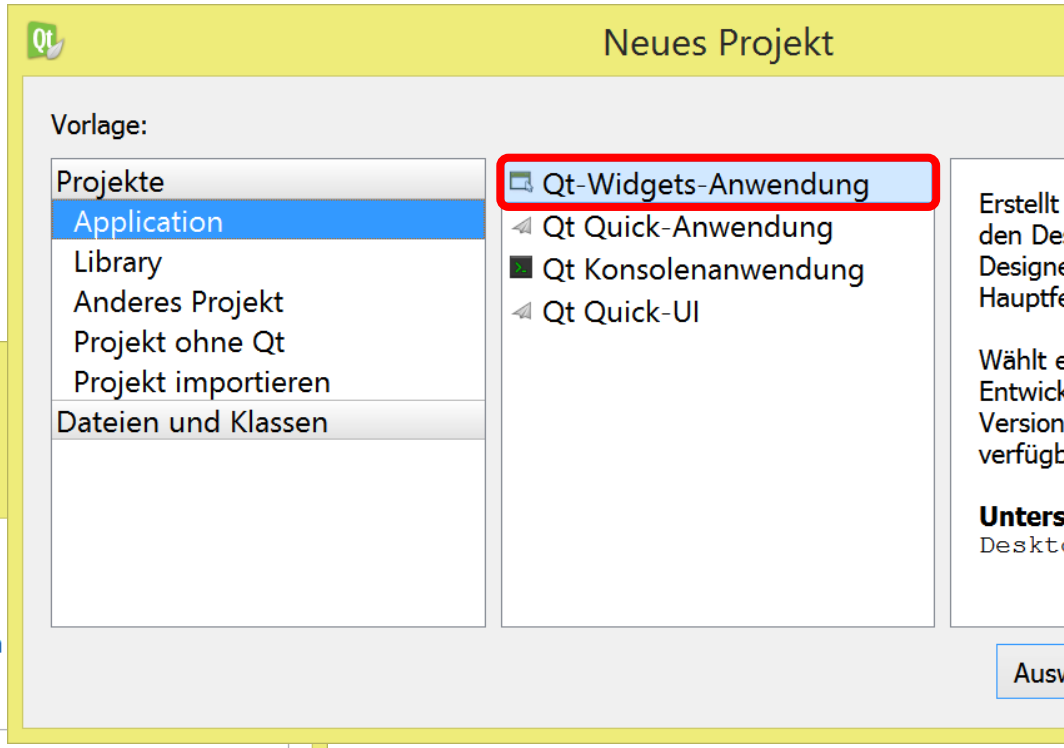
Es soll ein Programm zur Umwandlung von Dezimalzahlen in die Binär- bzw. Hexadezimaldarstellung erstellt werden:

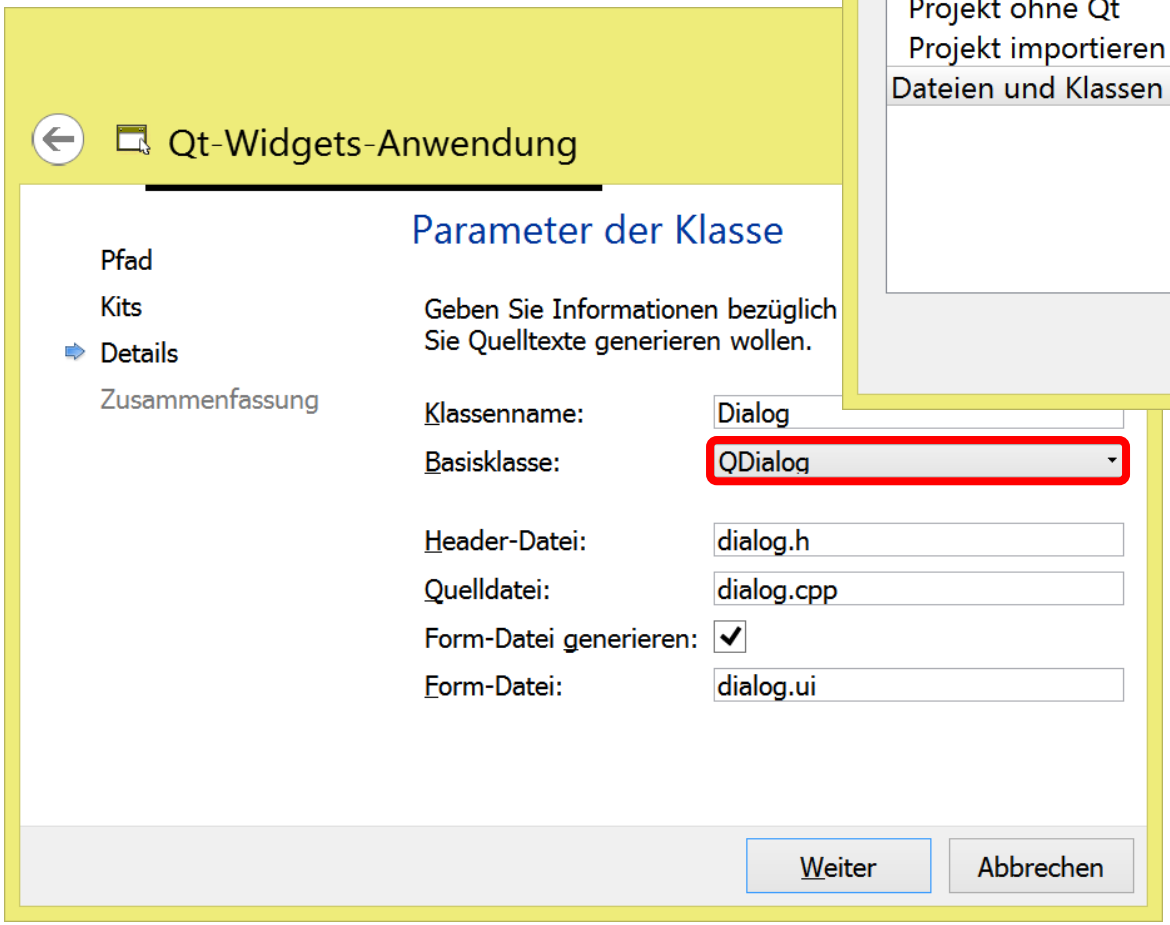
Der Anwender kann beliebige (positive, ganze) Zahlen eingeben und anschließend eine der Schaltflächen „Binär“ oder „Hexadezimal“ betätigen. Das Ergebnis wird in einem Meldungsfenster angezeigt.



P3.9. Ein erstes Beispiel

Erstellen Sie ein neues Qt-Widgets-Anwendungsprojekt.

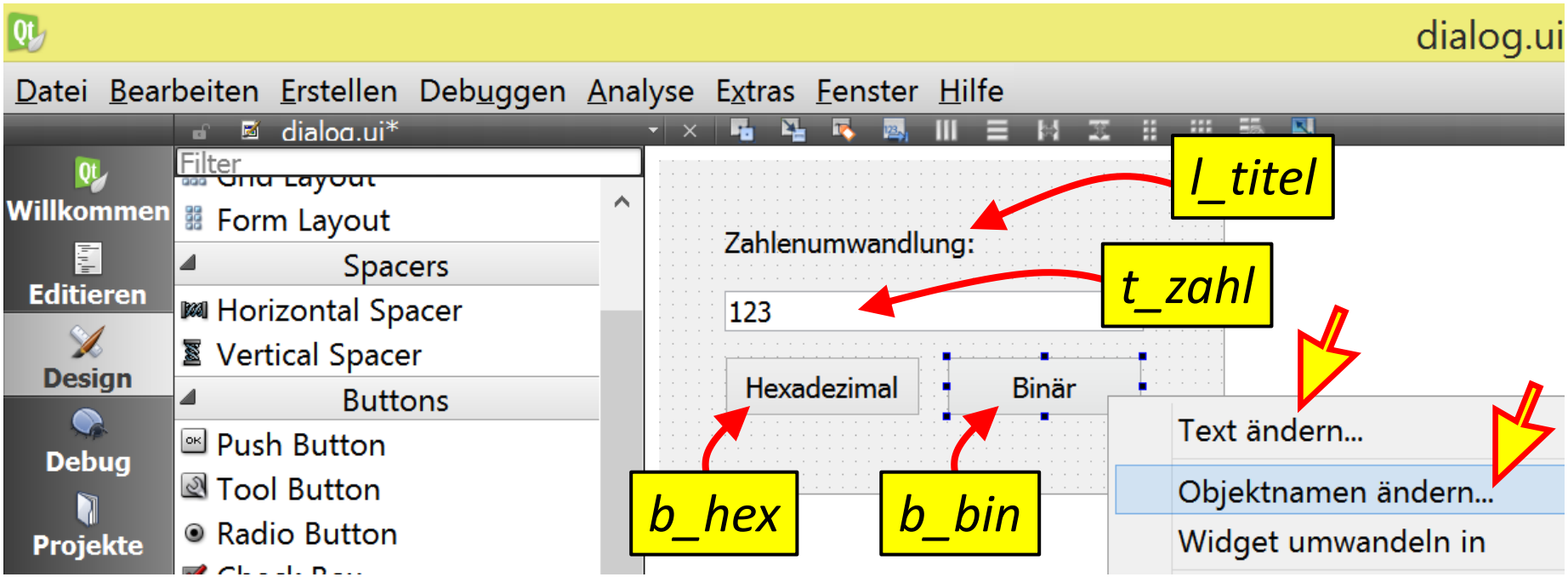




Unsere Applikation ist eine **Dialoganwendung**, wählen Sie QDialog als Basisklasse für das Hauptfenster.

P3.10. Ein erstes Beispiel

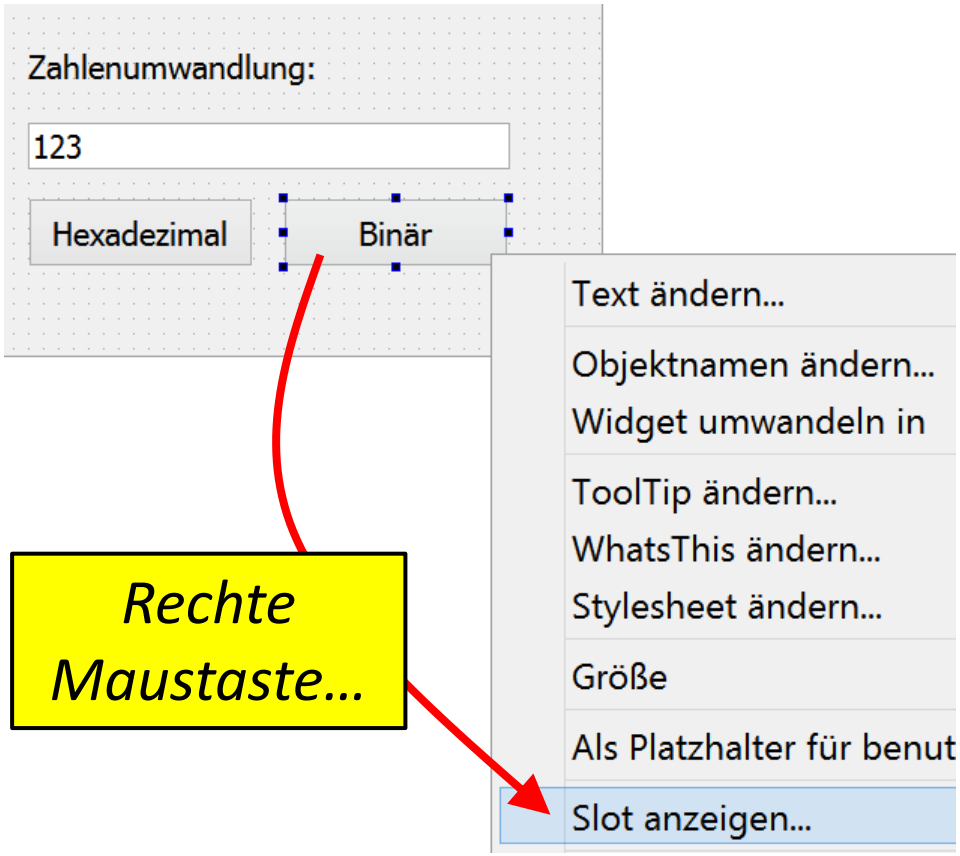
Durch Doppelklick auf den Dateinamen „dialog.ui“ öffnet sich der grafische Dialogeditor. Hier fügen Sie die Überschrift (**Label**), das Eingabefeld (**Line Edit**) und die beiden Schaltflächen (**Push Button**) zum Dialog hinzu.

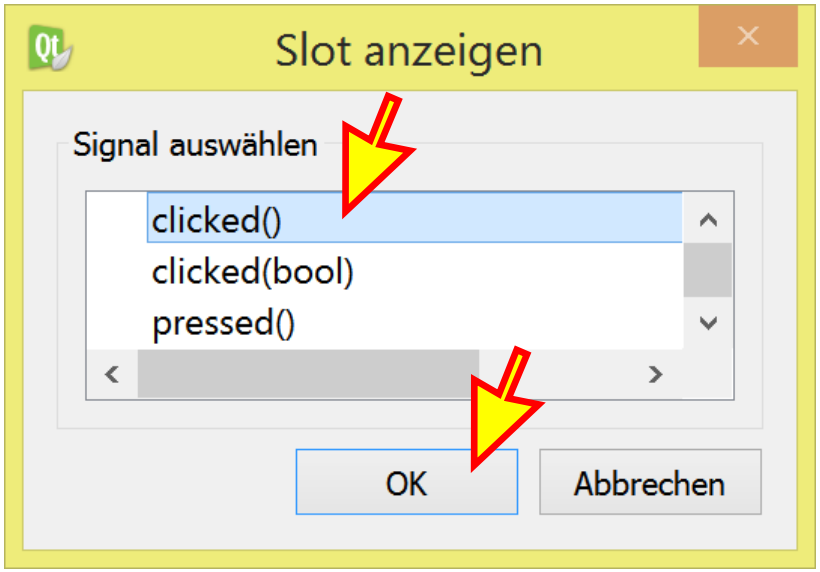


Denken Sie daran, jedem Widget einen **Objektnamen** zu geben und ggf. den **angezeigten Text** zu ändern (siehe Abbildung).

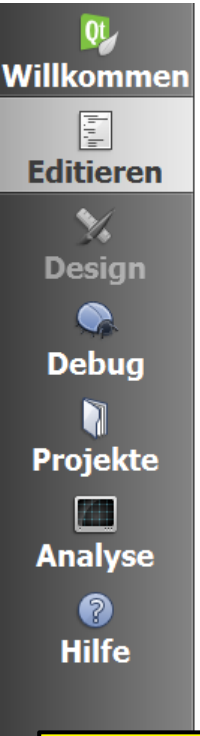
P3.11. Ein erstes Beispiel

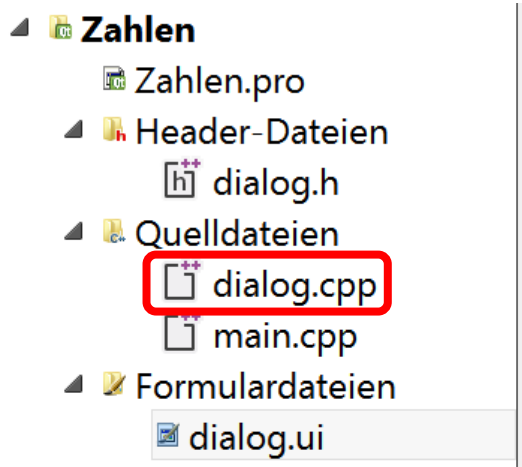
Wenn der Anwender eine der beiden Schaltfläche anklickt, werden **Signals** („clicked“) ausgelöst. Diese **Signals** werden mit passenden **Slots** (den Methoden „on_b_bin_clicked“ und „on_b_hex_clicked“) verbunden, wo die entsprechenden Reaktionen programmiert sind.





P3.12. Ein erstes Beispiel






```
1  #include "dialog.h"
2  #include "ui_dialog.h"
3  #include <QMessageBox>
4
5  void Dialog::on_b_bin_clicked()
6  {
7      QMessageBox msgBox;
8      msgBox.setText("Binär");
9      msgBox.exec();
10 }
11
12 void Dialog::on_b_hex_clicked()
13 {
14     QMessageBox msgBox;
15     msgBox.setText("Hexadezimal");
16     msgBox.exec();
17 }
18
19 Dialog::Dialog(QWidget *parent) :
20     QDialog(parent),
21     ui(new Ui::Dialog)
22 {
23     ui->setupUi(this);
```

In den Slots werden zunächst lediglich Meldungsfenster dargestellt.

Kompilieren und starten Sie nun die „nullte“ Version Ihrer Applikation!

P3.13. Ein erstes Beispiel

Nachdem Sie die „nullte Version“ Ihrer Applikation erstellt und sich von der korrekten Funktion überzeugt haben, schauen Sie sich einmal in Ruhe die vom Qt Creator (mehr oder weniger) automatisch erstellten Dateien an und versuchen Sie, den Programmaufbau zu verstehen.

main.cpp:

Das Hauptprogramm erzeugt eine Instanz der Klasse **Dialog** und zeigt das Dialogfenster auf dem Bildschirm an.

dialog.h:

Hier finden Sie die Deklaration der Klasse **Dialog**, die zunächst das Verhalten und alle Eigenschaften eines „normalen“ Dialogfensters von ihrer Basisklasse **QDialog** erbt. Hinzu kommen die Widgets (etwas versteckt in der Datei **ui_dialog.h**) und weitere Methoden und Attribute, die in Ihrer konkreten Anwendung benötigt werden.

dialog.cpp:

Hier sind alle Methoden der Klasse **Dialog** definiert.

P3.14. Ein erstes Beispiel

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN MÜNCHEN

dialog.h

```
#ifndef DIALOG_H
#defin

#include <QDialog>
#include <string>
using namespace std;

namespace Ui {
class Dialog;
}

cl                                    g
{


pu                                    *pa

private slots:
    void on_b_bin_clicked();
    void on_b_hex_clicked();

private:
    Ui::Dialog *ui;
    string binaer(double num);
};

#endif // DIALOG_H
```

Aufgabe:

Programmieren Sie nun die Umwandlung der eigegebenen Zahl ins Binärformat…!

dialog.cpp

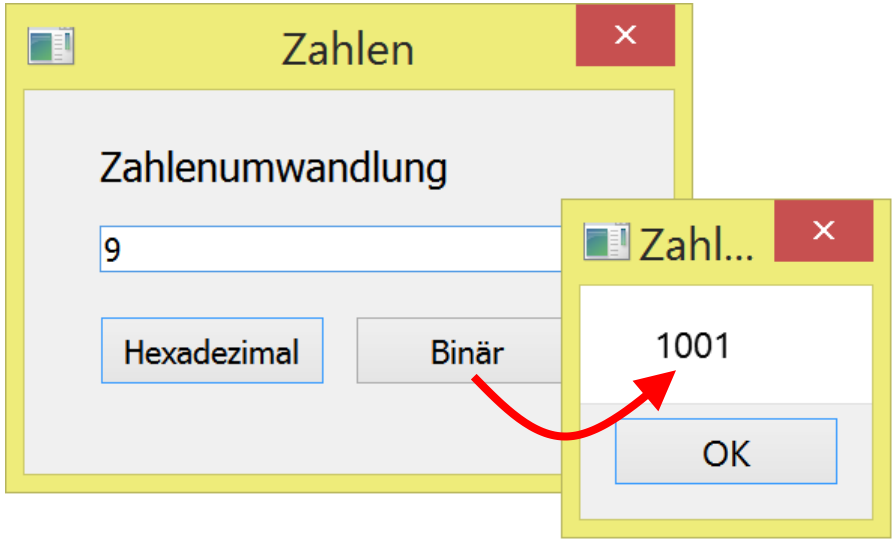
```
#include "dialog.h"
#include "ui_d
#include <QMes
#include <math.h>

string Dialog::binaer(double num)
{
    string result;
    int pot = (int)log2(num);
    for(int i = pot; i >= 0; --i)
    {
        if(num >= pow(2, i))
            result += "1", num -= pow(2, i);
        else
            result += "0";
    }
    return result;
}

void Dialog::on_b_bin_clicked()
{
    double zahl = ui->t_zahl->text().toDouble();
    string bin_zahl = binaer(zahl);

    QMessageBox msgBox;
    msgBox.setText(bin_zahl.c_str());
    msgBox.exec();
}
```

P3.15. Ein erstes Beispiel

Haben Sie bemerkt, wie Sie über das **Attribut „ui“** zu den einzelnen Widgets des Dialogfensters gelangen?

Für jedes Widget hat der Qt Creator eine eigene Variable angelegt. Der Variablenname entspricht dem von Ihnen im grafischen Dialogeditor eingegebenen Namen.

Testen Sie Ihre Applikation. Was passiert bei der Eingabe negativer Zahlen? Und bei der Eingabe von Kommazahlen? Können Sie die Applikation durch Eingabefehler zum Absturz bringen?



Die nächsten Schritte, vielleicht als Hausaufgabe?

- Eingabefehler erkennen, ggf. Fehlermeldung anzeigen
- Umwandlung ins Hexadezimalformat programmieren

Rechnerpraktikum zu Kapitel 3

Grafische Benutzeroberflächen

Einleitung

Ein erstes Beispiel

Schwingkreisberechnung mit grafischer Oberfläche

Codeschloss mit Zustandsautomat

Verkehrsampel mit Timer

P3.17. Schwingkreisberechnung

Aufgabe:

Im Rechnerpraktikum zu Kapitel 1 haben Sie ein Programm zur Simulation eines elektrischen Schwingkreises erstellt. Programmieren Sie nun eine grafische Applikation zur Schwingkreisberechnung:

Die Abbildung auf Folie 6 zeigt, wie Ihre Benutzeroberfläche aussehen könnte…

- Der Anwender soll den Widerstand R, die Induktivität L und die Kapazität C über die grafische Oberfläche eingeben und durch Druck auf die Schaltfläche „Simulation“ die Berechnung starten.
- Der von Ihnen im Rahmen von Kapitel 1 erstellte Quelltext kann und soll natürlich möglichst „recycelt“ werden – nur die grafische Oberfläche muss neu entworfen werden.
- Die Bibliotheken **chart.c/.h** und **matrix.c/.h/.hpp** können ebenfalls benutzt werden.

Rechnerpraktikum zu Kapitel 3

Grafische Benutzeroberflächen

Einleitung

Ein erstes Beispiel

Schwingkreisberechnung mit grafischer Oberfläche

Codeschloss mit Zustandsautomat

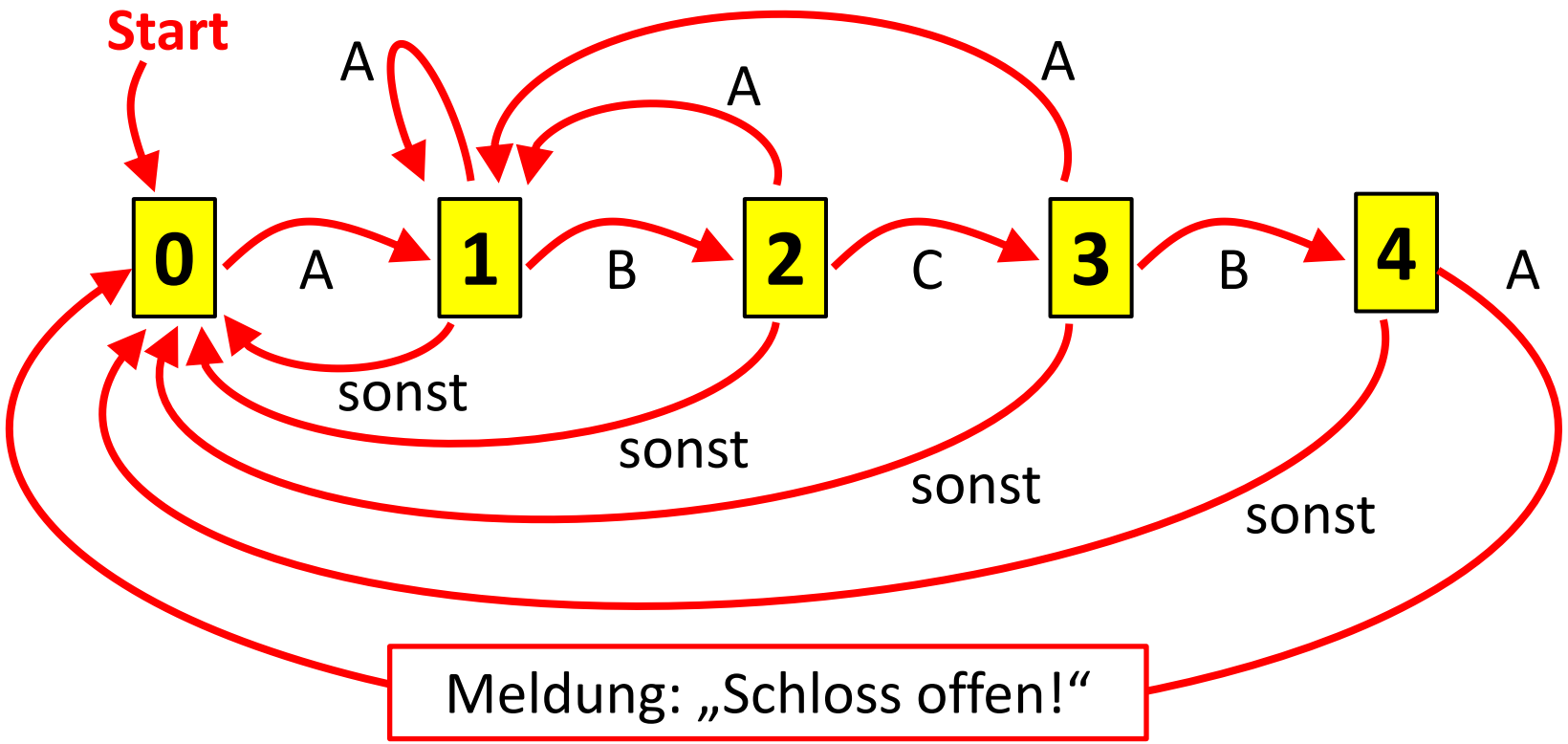
Verkehrsampel mit Timer

P3.19. Codeschloss

Aufgabe:

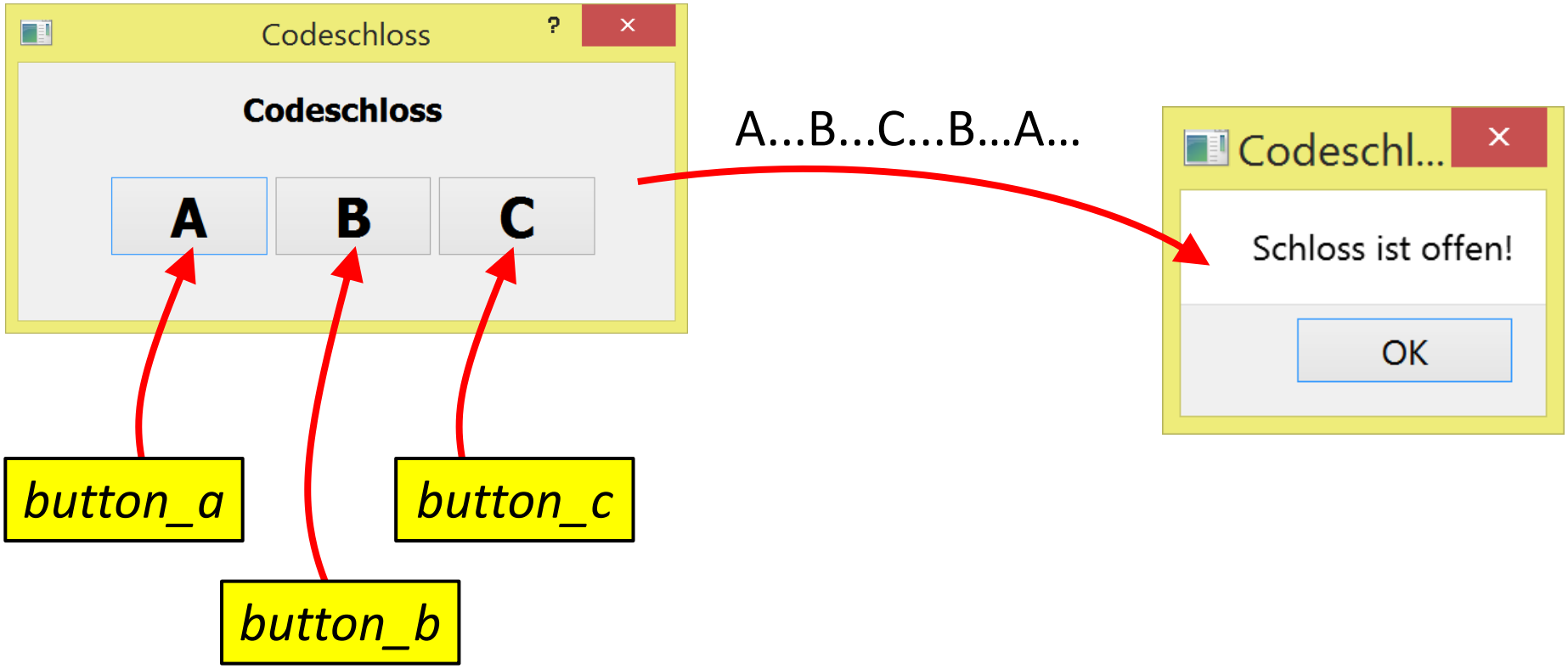
Programmieren Sie ein Codeschloss mit drei Tasten „A“, „B“ und „C“. Der Code zum Öffnen des Schlosses lautet „ABCBA“.

Das Codeschloss kann durch den folgenden **Zustandsautomaten** mit sechs verschiedenen Zuständen beschrieben werden:



P3.20. Codeschloss

Erzeugen Sie ein neues Qt-Widgets-Anwendungsprojekt und gestalten Sie zunächst eine einfache Benutzeroberfläche. Vergessen Sie nicht, allen Widgets einen eindeutigen Namen zu geben:



P3.21. Codeschloss

Fügen Sie in der Datei **dialog.h** die folgenden Zeilen hinzu:

```
private:
    Ui::Dialog *ui;
    int state;
    void aktion(char taste);
};

#endif // DIALOG_H
```

Fügen Sie in der Datei **dialog.cpp** die folgende Zeile innerhalb des Konstruktors hinzu:

```
Dialog::Dialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Dialog)
{
    ui->setupUi(this);
    state = 0;  // Startzustand
}
```

P3.22. Codeschloss

Fügen Sie nun mit dem grafischen Dialogeditor die drei **Slots** für die Schaltflächen „A“, „B“ und „C“ zum Dialog hinzu. Falls Sie unsicher sind, schauen Sie dazu nochmals auf Folie 11 nach…

```
void Dialog::on_button_a_clicked()
{
    aktion('a');
}

void Dialog::on_button_b_clicked()
{
    aktion('b');
}

void Dialog::on_button_c_clicked()
{
    aktion('c');
}
```

Der eigentliche Ablauf des Zustandsautomaten wird in der Methode **void Dialog::aktion(char taste);** implementiert.

```
void Dialog::aktion(char taste)
{
    // Zustandsautomat für Codeschloss
    switch(state)
    {
    case 0: if(taste == 'a') state = 1;
            break;
    case 1: ...
            break;
    case 2: ...
            break;
    case 3: ...
            break;
    case 4: if(taste == 'a')
            {
                // QMessageBox anzeigen:
                // „Schloss wurde geöffnet!“
                ...
            }
            state = 0;
            break;
    }
}
```

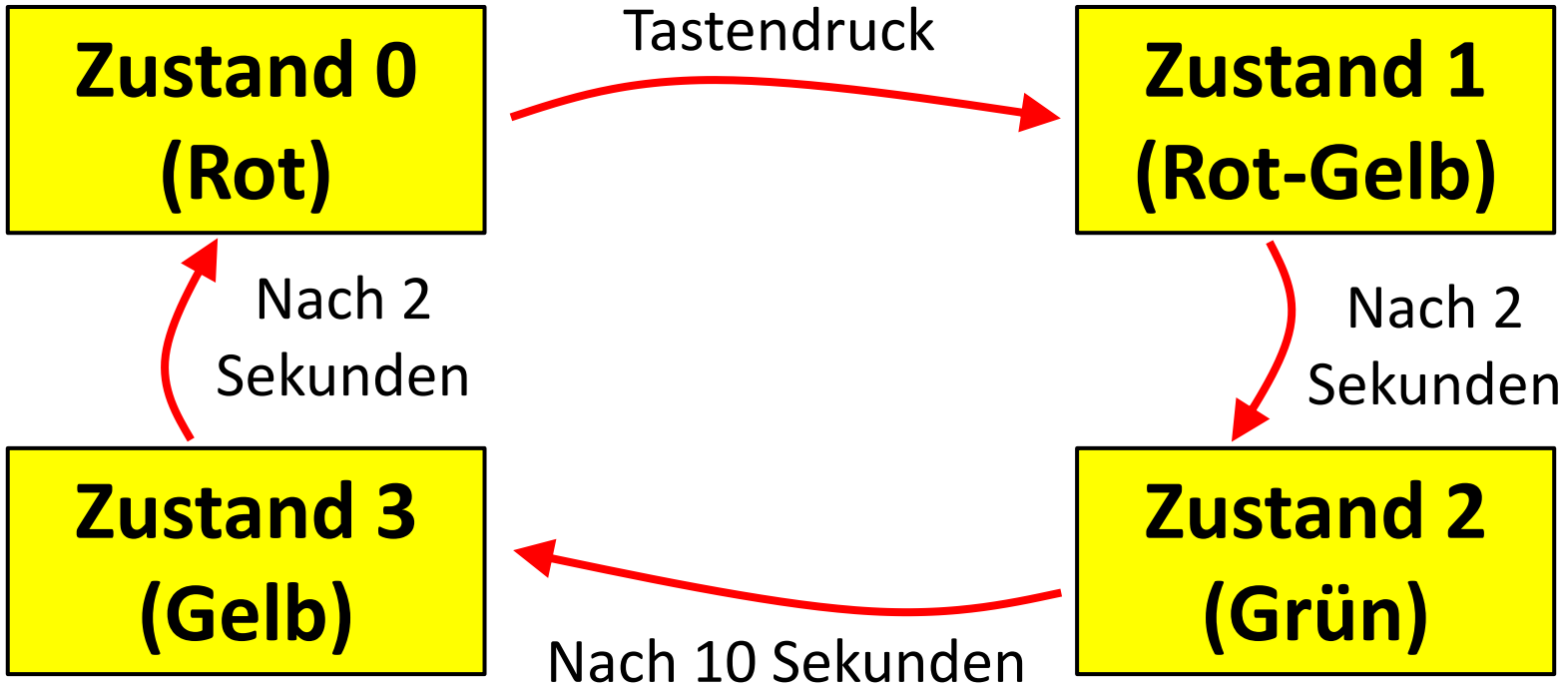
*Wenn das Codeschloss geöffnet wurde, erfolgt eine Meldung auf dem Bildschirm. Danach wird mittels **state = 0;** der Anfangszustand wiederhergestellt…*

Rechnerpraktikum zu Kapitel 3

Grafische Benutzeroberflächen

P3.25. Verkehrsampel

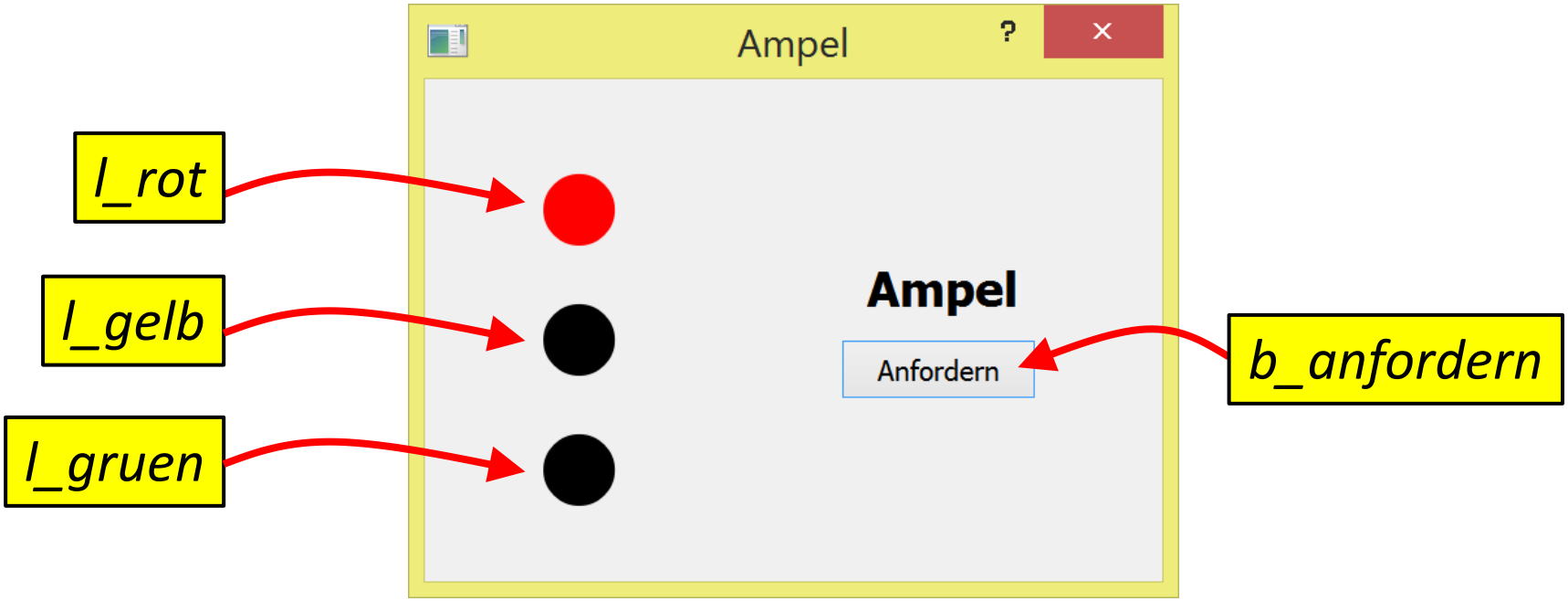
Auch eine Verkehrsampel kann als Zustandsautomat programmiert werden. In diesem Beispiel steht die Ampel solange auf „Rot“, bis eine Taste betätigt wird. Die Ampel durchläuft anschließend **durch einen Timer gesteuert** automatisch die Zustände „Rot-Gelb“, „Grün“ usw. und bleibt schließlich wieder auf „Rot“ stehen.



P3.26. Verkehrsampel

Aufgabe:

Erzeugen Sie ein neues Qt-Widgets-Anwendungsprojekt und gestalten Sie zunächst die Benutzeroberfläche der Verkehrsampel (mit Knopf). Vergessen Sie nicht, allen Widgets einen eindeutigen Namen zu geben:



Anschließend implementieren Sie das Verhalten der Verkehrsampel, den zeitlichen Ablauf, die roten/gelben/grünen „Lampen“…

P3.27. Verkehrsampel

Folgende Teilprobleme müssen gelöst werden:

1. Wie programmiert man einen Timer?
2. Wie färbt man die „Lampen“ der Verkehrsampel ein?
3. Wie sieht die Struktur des Gesamtprogramms aus, wo ist der Zustandsautomat implementiert?

Teilproblem 1: Timer, alle 1000 msec wird ein Ereignis generiert

```
Dialog::Dialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Dialog)
{
    ui->setupUi(this);
    state = 0;
    startTimer(1000);
}

void Dialog::timerEvent(QTimerEvent *)
{
```

dialog.cpp

```
private:
    Ui::Dialog *ui;
    int state;
    int sekunden;
    void timerEvent(QTimerEvent *);
};
```

dialog.h

```
void Dialog::timerEvent(QTimerEvent *)
{
    switch(state)
    {
    case 0:
        ui->l_rot->setStyleSheet("QLabel { color : red; }");
        ui->l_gelb->setStyleSheet("QLabel { color : black; }");
        ui->l_gruen->setStyleSheet("QLabel { color : black; }");
        break;

    case 1:
        ui->l_gelb->setStyleSheet("QLabel { color : yellow; }");
        if(sekunden++ == 2) { state = 2; sekunden = 0; }
        break;

    case 2: ...
    case 3: ...
    }
}

void Dialog::on_b_anfordern_clicked()
{
    if(state == 0) { state = 1; sekunden = 0; }
}
```

Diese Methode wird automatisch alle 1000 msec aufgerufen.

Teilproblem Nr. 2, Einfärben der „Lampen“
Teilproblem Nr. 3, Zustandsautomat

Dies ist der Slot für die Schaltfläche „Anfordern“