Qt für C++-Anfänger

de.wikibooks.org

15. Mai 2016

Inhaltsverzeichnis

1 Übersicht

1.1 Einleitung

Dieses Tutorial richtet sich an diejenigen, die einen schnellen und einfachen Einstieg in die Qt-Programmierung suchen. Dieses Tutorial setzt Grundkenntnisse in der C++-Programmierung voraus. Neben der Programmierung soll auch das Lesen der Qt-Dokumentation geübt werden, was für die Entwicklung umfangreicher Qt-Anwendungen unverzichtbar ist.

Dieses Tutorial bezieht sich auf die Qt-Version 4, welche zur Version 3 nicht kompatibel ist. Wer jedoch mit der älteren Version von Qt gearbeitet hat, wird einen schnellen Einstieg in die neue Version finden, da sich in der grundlegenden Programmstruktur nicht viel geändert hat. Die GUI soll dabei mittels des Qt-Designers erstellt werden. Dazu wäre es vorteilhaft, wenn man sich vorher mit dem Qt-Designer befasst. Die Qt-Designer-Tutorials im Web, die ich kenne, haben alle diverse Schwächen, weswegen ich sie für ungeeignet halte, um mit ihnen ein lauffähiges Programm zu erstellen. Sie sind in erster Linie für Qt3 geschrieben, das - wie oben schon angedeutet - zu Qt4 inkompatibel ist. Außerdem wird anhand dieser Tutorials nur eine Maske erzeugt und kein lauffähiges Programm. Den Umgang mit dem Qt Designer können sie trotzdem nahebringen.

Die Befehle in diesem Tutorial beziehen sich auf den Qt-Designer unter Linux. Ich nehme an, dass sie unter Windows und anderen Betriebssystem identisch sind. Ansonsten ist das jetzt ein guter Zeitpunkt, um Linux zu installieren. ;-)

Die Anwendung, die wir in diesem Tutorial erzeugen werden, wird ein Taschenrechner zum Addieren und Subtrahieren sein. Extrem simpel, aber man hat hinterher ein Grundgerüst, welches man benutzen kann, um mit seinen eigenen Anwendungen los zu legen.

2 Das Grundgerüst

2.1 Die Erstellung des GUIs

Die grafische Benutzeroberfläche wird mit dem Tool ”Qt Designer” erstellt. Unter Linux lässt es sich mit dem Kommando ”designer” bzw. ”designer-qt4” aufrufen.

Der Taschenrechner soll später mal so aussehen:

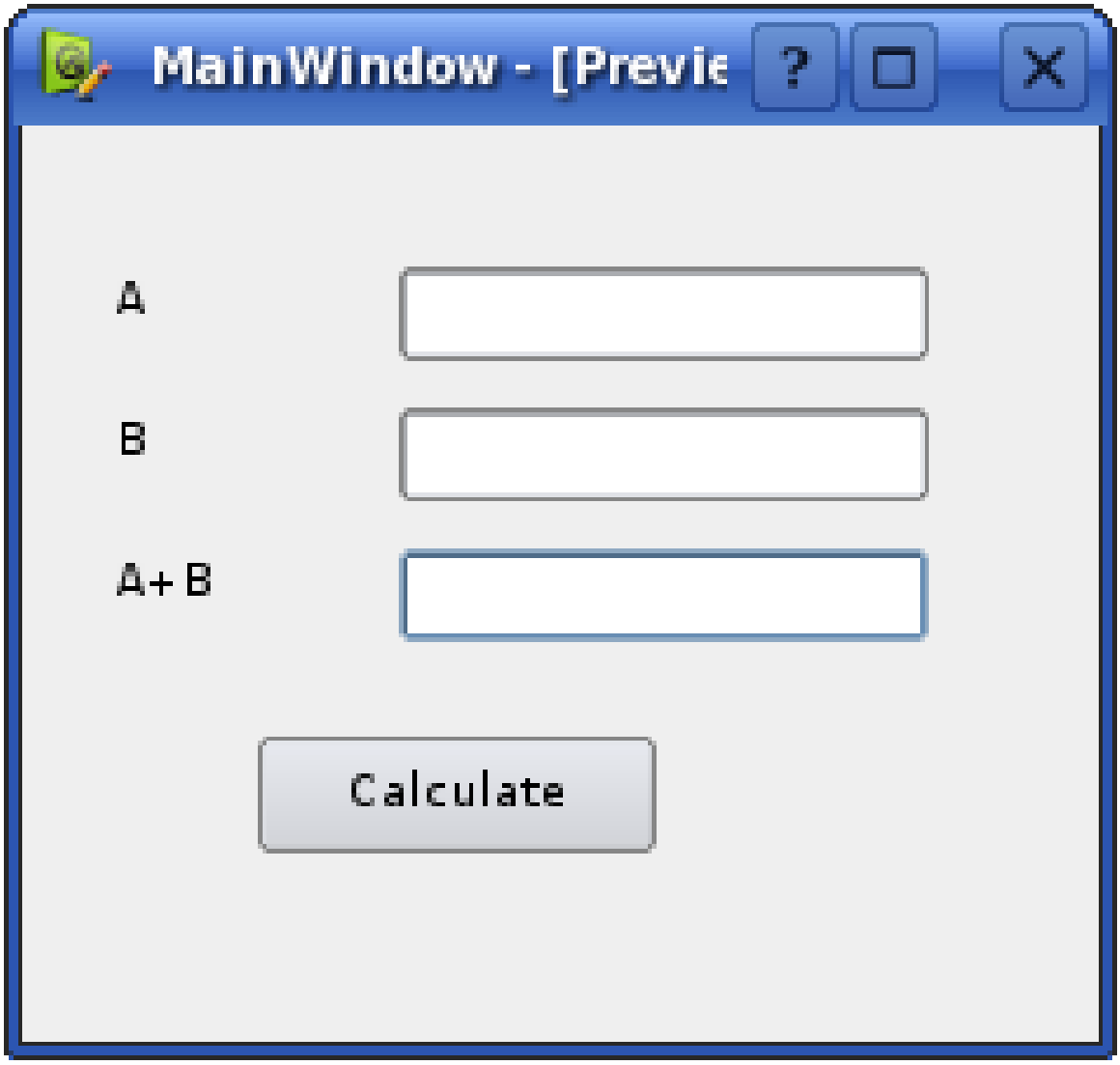

Abb. 1

Er besteht also aus einem Fenster (als Template bitte Main Window wählen) mit drei Textfeldern, drei "Eingabezeilen" und einem Knopf, um die Berechnung zu starten. Die Elemente, die ihr im Qt-Designer dazu zusammen klicken müsst, sind die folgenden, bitte auch genau mit diesen technischen Namen:

```
QLabel: LabelA, LabelB, LabelC
QLineEdit: InputA, InputB, ResultC
QPushButton: Calculate
```

Bitte auf Groß/Kleinschreibung achten. Die technischen Namen sind nicht die, die man im obigen Screenshot sieht, sondern die, die im Fenster "Property Editor" im Bereich QObject oben stehen; das Feld heißt objectName. Damit haben wir dann ein ui-File; wir speichern es unter Taschenrechner.ui in einem Verzeichnis, wo wir auch in Zukunft alle Files des Projektes speichern. Wo das Verzeichnis liegt, ist egal, es sollte aber bitte "Taschenrechner" genannt werden.

2.2 Das Code-Grundgerüst

Damit wir nun auch wirklich ein C++-Programm erstellen können, brauchen wir ein Code-Gerüst, an dem wir später weitere Arbeiten vornehmen. Das erste Gerüst wird es uns erlauben, das Programm zu übersetzen und auszuführen - allerdings noch ohne weitere Funktionalität.

Wir werden drei C++-Files haben:

```
//--- main.cpp - start ---

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

int main( int argc, char* argv[])
{
    QApplication a(argc, argv);
    Taschenrechner w;
    w.show();
    return a.exec();
}

//--- main.cpp - end ---
```

```
//--- Taschenrechner.h - start ---

#ifndef TASCHENRECHNER_H
#define TASCHENRECHNER_H

#include "ui_Taschenrechner.h"

class Taschenrechner : public QMainWindow, public Ui::MainWindow{
    Q_OBJECT

    public:
        Taschenrechner (QMainWindow *parent = 0);
        ~Taschenrechner();
```

```
};
#endif //TASCHENRECHNER_H

//--- Taschenrechner.h - end ---
```

```
//--- Taschenrechner.cpp - start ---

#include "Taschenrechner.h"

Taschenrechner::Taschenrechner(QMainWindow *parent) :
 QMainWindow(parent){
    setupUi(this);
}

Taschenrechner::~Taschenrechner(){
}

//--- Taschenrechner.cpp - end ---
```

Auch hier bitte auf Groß- und Kleinschreibung achten, insbesondere wird das Wort "Taschenrechner" immer mit einem großen "T" geschrieben. Bitte erstellt die Files so wie hier angegeben; es macht erstmal nichts, wenn man den Code hier nicht versteht; die Erklärungen im Detail folgen weiter unten.

2.3 Das erste Übersetzen

Wechsle in einer Shell in unser Projektverzeichnis und führe nacheinander folgende Befehle aus:

```
qmake -project
qmake
make
./Taschenrechner
```

Beachte: Bei manchen Distributionen heißt der Befehl qmake-qt4.

2.3.1 qmake -project

Erzeugt eine .pro-Datei; die so genannte Projekt-Datei. Der Name des Files ist der Name des Projektverzeichnisses. In dieser Datei stehen im wesentlichen alle Dateien, die zu unserem Projekt gehören. Wenn man neue Dateien erstellt, also nicht nur die bestehenden ändert, muss dieser Befehl erneut aufgerufen werden. Man kann die .pro-Datei mit einem Texteditor öffnen und bekommt eine ganz gute Vorstellung, was darin steht.

2.3.2 qmake

qmake erzeugt das Makefile.

2.3.3 make

make übersetzt am Ende unsere Quelldateien basierend auf dem Makefile. Es werden auch noch Qt-spezifische Header-Dateien erstellt. Die Datei ui_Taschenrechner.h enthält den Qt-Code für die Benutzerschnittstelle (`ui` steht für engl. *user interface*). Das Endergebnis ist eine ausführbare Datei mit dem Namen des Projekts, welche identisch mit dem Namen unseres Projektverzeichnisses ist. Ist unter Windows der Aufruf von make nicht erfolgreich, muss der komplette Pfad zur make.exe eingegeben werden: z.B. <QT-Verzeichnis>\QT\Symbian\SDKs\Symbian3Qt474\epoc32\tools\make.

2.4 Das Grundgerüst im Detail

2.4.1 Taschenrechner.h

ist das Header File für unsere Klasse Taschenrechner. In diesem Header File finden wir die so genannten Prototypen unserer Funktionen. Also die Beschreibung welche Funktionen zur Verfügung stehen, hier wird aber keine Funktion implementiert.

Nun zu den Details:

```
03 #ifndef TASCHENRECHNER_H
04 #define TASCHENRECHNER_H
<...>
15 #endif //TASCHENRECHNER_H
```

sorgt dafür, dass der Header nur einmal eingebunden wird. Dabei handelt es sich um sogenannte *Includewächter*. Das sind Makros, die vom Präprozessor abgearbeitet werden, bevor der Compiler das Programm übersetzt. In Zeile 03 wird nachgeschaut, ob das Flag TASCHENRECHNER_H gesetzt ist. Ist es nicht gesetzt, wird der ganze Code bis zur Zeile 15 zum Übersetzen "freigegeben". Das `#endif` in Zeile 15 schließt diese if-Bedingung ab. `#ifndef` steht hier für "if not defined". Gesetzt wird ein solches Flag über die Präprozessor-Anweisung `#define` in Zeile 04. Beim allerersten Auffinden des obigen Blocks ist das Flag TASCHENRECHNER_H also noch nicht gesetzt und wird abgearbeitet. Durch diese Abarbeitung wird das Flag in Zeile 04 nun gesetzt. Trifft der Präprozessor nun ein weiteres Mal auf diesen Code, wird dieser einfach nicht mehr beachtet, was auch nicht mehr nötig ist, da alle Prototypen schon bekannt sind. Somit wird also verhindert, dass Definitionen von Compiler mehrfach verarbeitet werden, was zu Fehlern beim Übersetzen führen würde. Die Form der Namensgebung dieses Flags (als eine Ableitung aus dem Dateinamen) ist eine weit verbreitete Konvention. Im Prinzip würde es jeder andere Name auch tun – was aber nicht zu empfehlen ist. Es würde Verwechslungen wahrscheinlicher machen.

```
06 #include "ui_Taschenrechner.h"
```

Dieses Header-File wurde von make als erstes erzeugt, wenn man dort rein schaut, erkennt man sehr leicht, dass hier alle UI Elemente zu finden sind.

```
08 class Taschenrechner : public QMainWindow, public Ui::MainWindow {
09        Q_OBJECT
10
11 public:
```

```
12        Taschenrechner (QMainWindow *parent = 0);
13        ~Taschenrechner();
14 };
```

Dies ist die Klassendeklaration unserer Hauptklasse. In Zeile 08 wird der Klassenname "Taschenrechner" angegeben; außerdem zeigt dies an, dass unsere Klasse abgeleitet ist von QMainWindow ": public QMainWindow". Wollen wir beispielsweise einen QDialog erstellen, wäre hier die Ableitung auf QDialog zu ändern. In "ui_Taschenrechner.h" findet die Definition der Taschenrechnermaske "MainWindow" im Namensraum "Ui" statt. "Ui::MainWindow" ist seinerseits nur ein Container bzw. eine Ableitung der eigentlichen Maskendefinition in "Ui_MainWindow".

Zeile 09("Q_OBJECT") ist ein weiteres Makro, welches immer bei Qt in der Klasse, die Signals und Slots[1] behandelt, angegeben werden muss. Diese werden dann durch den Meta-Object Compiler[2] ausgewertet. Vergisst man das Makro, werden Signale und Slots nicht behandelt. Unsere Klasse hat zur Zeit zwei Methoden deklariert. Beide sind "public", können also von einer beliebigen Funktion aufgerufen werden. Bei den Methoden handelt es sich um den Konstruktor und Destruktor. Die Methode, die den gleichen Namen hat wie die Klasse selbst (Zeile 12), ist immer der Konstruktor und die mit dem gleichen Namen, aber einer vorangestellten Tilde (~), ist immer der Destruktor (Zeile 13). Diese Methoden werden immer beim Anlegen oder Zerstören eines Objekts der Klasse aufgerufen. Dem Konstruktor wird hier noch ein Parameter übergeben, der dem Objekt ein Elternwidget zuweist. In unserem Beispiel ist der Standard bei nicht-übergebenem Argument, *parent = 0*.

2.4.2 Taschenrechner.cpp

```
01 #include "Taschenrechner.h"
02
03 Taschenrechner::Taschenrechner(QMainWindow *parent) : QMainWindow(parent){
04         setupUi(this);
05 }
06
07 Taschenrechner::~Taschenrechner(){
08 }
```

Hier werden die deklarierten Funktionen aus der Headerdatei wirklich implementiert. In Zeile 01 wird das Header-File inkludiert. Damit ist bekannt, welche Funktionen es geben wird. In Zeile 03 - 05 wird der Konstruktor implementiert. Das ": QMainWindow(parent)" nach dem Konstruktor sorgt dafür, dass der Konstruktor der Klasse QMainWindow, von der unsere Klasse abgeleitet worden ist, vor dem Aufruf mit dem angegebenen Argument aufgerufen wird. Die einzige Anweisung, die für den Konstruktor implementiert ist, ist in Zeile 04 gegeben. Es handelt sich um eine Qt-spezifische Funktion, die in dem File ui_Taschenrechner.h zu finden ist. Der Aufruf dieser Funktion legt letztendlich das UI an.

In den Zeilen 07 und 08 ist der Destruktor definiert; dieser ist leer. Es werden keine speziellen Funktionen ausgeführt. Um die Speicherfreigabe nach dem Zerstören des Objekts kümmert sich Qt von selbst. Man beachte hier, dass das Fehlen des Returntyps (bspw.

1 Kapitel 4 auf Seite 13

2 http://en.wikipedia.org/wiki/Qt_(toolkit)#Metaobject_compiler

”void”) spezifisch für Konstruktoren und Destruktoren ist! Dies hat nichts mit Qt zu tun, sondern tritt bei vielen objektorientierten Programmiersprachen auf.

2.4.3 main.cpp

```
01 #include "Taschenrechner.h"
02 #include <QApplication>
03
04 int main( int argc, char* argv[]){
05        QApplication a(argc, argv);
06        Taschenrechner w;
07        w.show();
08        return a.exec();
09 }
```

Die Main-Funktion, der Startpunkt eines C++ Programms, enthält nicht sehr viel. Es wird sich auch im Laufe der Zeit nicht viel an dem Aussehen ändern. Zeile 01 bindet das Headerfile ein, damit die Klasse ”Taschenrechner” bekannt ist. Das wird in Zeile 06 wichtig sein. Zeile 04 und 09 stellen das Gerüst der Main-Funktion dar. In Zeile 05 wird ein Qt Objekt ”QApplication” (durch Zeile 02 eingebunden) erstellt und die Kommandozeilenparamter werden an dieses Objekt übergeben. Der Objektname ist einfach ”a”. In Zeile 06 wird eine Instanz der Klasse ”Taschenrechner” erstellt, diese Instanz wird ”w” genannt, (kommt wohl von Window oder Widget) man könnte aber jeden beliebigen Namen wählen. In Zeile 07 rufen wir die Funktion show() auf. Diese Funktion haben wir nicht explizit implementiert, durch die Ableitung von QMainWindow steht uns aber diese Funktion zur Verfügung. QMainWindow hat diesen ”Slot” von QWidget geerbt. Diese Funktion sorgt dafür, dass das Widget angezeigt wird. Sprich: Ließe man sie weg, würde das Programm immer noch ”funktionieren”; man würde bloß nichts sehen. Zeile 08 übergibt die Kontrolle des Programms. Die Kontrolle heißt dabei, dass der Aufruf a.exec() Qt anweist, auf Events zu hören. Ohne diese Anweisung wäre also keine Userinteraktion möglich. a.exec() wird erst beendet, wenn das Qt-Programm beendet wird. Das heißt also, dass das ”return” in Zeile 08 erst ausgeführt wird, wenn das Qt-Programm beendet wird.

3 Erste UI-Erweiterungen

Als erste Erweiterung sollen am UI ein paar Veränderungen vorgenommen werden - ohne den Qt-Designer. In den Inputfeldern von A und B soll ein Startwert von ”0” vorgegeben werden.

Dazu muss als erstes herausgefunden werden, wie man in dem QLineEdit-Objekt einen Text eintragen kann. Dazu schaut man in der QLineEdit-Dokumentation[1] nach einer Funktion, die sich so anhört, als ob man sie gebrauchen könnte. Nach einiger Zeit hat man darin etwas Übung. Schaut man in die Dokumentation, findet man zwei Funktionen, die passen könnten, setText() und insert(). Beide würden das machen, was wir hier erreichen wollen. Wir wählen setText(). Die Syntax der Funktion ist laut Dokumentation:

```
void setText ( const QString & )
```

Die Funktion liefert keinen Rückgabewert, da dieser mit ”void” angegeben ist. Als Input-Parameter wird ein ”const QString &” erwartet. Man erkennt leicht, dass vom Typ her ein QString, das ist im wesentlichen ein normaler String, erwartet wird. Bleibt noch zu klären, was ”const” und ”&” bedeuten. Das ”const” gibt an, dass sich der Wert der Variable innerhalb der Funktion nicht ändert bzw. nicht geändert werden darf. Würde man sie innerhalb der Funktion versuchen zu verändern, würde es einen Compiler-Error geben. Da wir die Funktion nicht selber schreiben, sondern sie von Qt vorgegeben ist, muss einen das ”const” nicht weiter stören. Ferner erlaubt das ”const”, dass man explizite Werte, also z.B. ”Ich bin ein String”, direkt mitgeben kann. Ohne das ”const” wäre das nicht möglich. Das ”&” gibt an, dass man ein so genanntes ”Passing by reference” macht. Man gibt beim Aufruf der Funktion nur die Variable an, aber implizit wird die Adresse, an der die Variable steht, weitergereicht. Sprich, sollte die Variable innerhalb der Funktion verändert werden, so hat diese Änderung auch außerhalb der Funktion Gültigkeit. Da wir oben gesehen haben, dass die Variable nicht geändert wird, erscheint das erstmal als rein theoretisches Konstrukt. Hintergrund ist, dass es so schneller in der Ausführung ist, als wenn man ein ”Passing by value” machen würde. Kurz zusammen gefasst sagt uns obiges Argument einfach nur, dass man ein QString-Objekt übergeben muss.

Als nächstes muss man sich überlegen, wie der Befehl konkret aussieht:

```
InputA -> setText("0");
```

InputA ist der Name, den wir dem QLineEdit-Objekt im Designer gegeben haben. InputA ist aber nur ein Pointer auf diese Instanz, will man eine Funktion einer Instanz aufrufen, nutzt man den Operator ”->”.

1 http://qt-project.org/doc/qt-4.8/qlineedit.html

Jetzt ist nur noch die Frage offen, wo der Code für InputA und InputB eingefügt werden muss. Da die Eigenschaften von Anfang an gegeben sein sollen, ist der geeignete Platz der Konstruktor. Die einzige Datei, welche wir ändern müssen, ist also "Taschenrechner.cpp".

```
//--- Taschenrechner.cpp - start ---

#include "Taschenrechner.h"

Taschenrechner::Taschenrechner(QMainWindow *parent) : QMainWindow(parent){
        setupUi(this);
        InputA -> setText("0");
        InputB -> setText("0");
}

Taschenrechner::~Taschenrechner(){
}

//--- Taschenrechner.cpp - end ---
```

Nun einfach erneut "make" ausführen und das Programm starten, um die Änderung zu sehen. Als nächstes könnte man die beiden neuen Zeilen gegen diese austauschen:

```
        InputA -> insert("0");
        InputB -> insert("0");
```

4 Signale und Slots

Signale und Slots sind ein Mechanismus von Qt, wie sich verschiedene GUI-Elemente oder Aktionen unterhalten können. Jemand sendet ein Signal aus und ein anderer empfängt dieses. Ein Signal kann z.B. beim Drücken eines Buttons ausgesendet werden. Ein oder mehrere Empfänger, die so genannten Slots, empfangen das Signal und rufen daraufhin eine entsprechende Funktion auf, die z.B. irgendeine Berechnung startet. Als erstes müssen wir uns überlegen, wie man Signale und Slots miteinander verbindet.

Diese Verbindung wird über das connect-Statement hergestellt. connect ist dabei ein Qt-spezifisches Makro, welches vom Präprozessor in echtes C++ umgesetzt wird. Die Syntax sieht wie folgt aus:

```
connect(Calculate, SIGNAL(clicked()), this, SLOT(addAB()));
```

”Calculate” ist das Qt-Objekt, das ein Signal aussendet. Calculate ist der Name des QPush-Button, den wir im Qt Designer festgelegt hatten.

”SIGNAL(clicked())” bezeichnet das Signal, welches abgefangen werden soll. Objekte können unterschiedliche Signale aussenden. In der Qt-Dokumentation können die für das jeweilige Qt-Objekt verfügbaren Signale nachgeschlagen werden.

”this” bezeichnet die Instanz der Klasse, deren aufzurufende Methode im Folgenden angegeben wird.

”SLOT(addAB())” bezeichnet die Funktion, die aufgerufen werden soll. Für diese Funktion muss natürlich, wie für jede andere C++ Funktion auch, eine Deklaration erstellt werden.

Dieser connect-Befehl muss im Konstruktor unserer Klasse stehen, damit er gleich am Anfang ausgeführt wird. Die Deklaration des Slots addAB() findet im Headerfile statt. Das Headerfile sieht damit folgendermaßen aus:

```
//--- Taschenrechner.h - start ---

#ifndef TASCHENRECHNER_H
#define TASCHENRECHNER_H

#include "ui_Taschenrechner.h"

class Taschenrechner : public QMainWindow, public Ui::MainWindow {

Q_OBJECT

public:
    Taschenrechner(QMainWindow *parent = 0);
    ~Taschenrechner();
private slots:
    void addAB();
};
```

```
#endif //TASCHENRECHNER_H

//--- Taschenrechner.h - end ---
```

Wie man sieht, findet die Deklaration der Funktion addAB() innerhalb der Klasse statt. Es handelt sich also um eine Memberfunktion. Es wird noch angegeben, dass es sich um eine private Funktion handelt. Mit dem Makro ”slots” wird gesagt, dass es sich bei dieser Funktion um einen Slot handelt. Die Datei Taschenrechner.cpp mit der eigentlichen Implementation sieht dann so aus:

```
//--- Taschenrechner.cpp - start ---

#include "Taschenrechner.h"

Taschenrechner::Taschenrechner(QMainWindow *parent) : QMainWindow(parent) {
    setupUi(this);
    InputA -> setText("0");
    InputB -> setText("0");

    connect(Calculate, SIGNAL(clicked()), this, SLOT(addAB()));
}

Taschenrechner::~Taschenrechner() { }

void Taschenrechner::addAB() {
    //Hier wird im nächsten Kapitel die Funktion implementiert
}

//--- Taschenrechner.cpp - end ---
```

Alle Slots und Funktionen, die benötigt werden, werden also einfach in der Datei Taschenrechner.cpp nacheinander aufgelistet. Mit der Angabe ”Taschenrechner::” wird mitgeteilt, dass es sich um eine Funktion der Klasse Taschenrechner handelt. Der Funktionskopf muss ansonsten genau so aussehen wie im zugehörigen Headerfile.

Wenn man das Programm nun übersetzt und ausführt, wird man erst einmal keine Veränderung feststellen, da die Funktion addAB() noch leer ist. Als letztes sei angemerkt, dass die Funktion addAB() keinen Wert zurück gibt (void). Ein ”return;” schadet zwar nicht, man kann sich die Tipparbeit aber sparen. Nur, wenn es mehrere Ausstiege aus der Funktion gibt, was zum Beispiel bei Schleifen der Fall sein kann, muss ein return verwendet werden.

4.1 Siehe auch

- Signals and Slots[1] beim Qt Project

1 http://qt-project.org/doc/qt-4.8/signalsandslots.html

5 Die erste Version

5.1 Die Funktion addAB()

Jetzt muss noch die Funktion addAB() implementiert werden und unser Taschenrechner ist in der ersten Version fertig. Die Implementation dieser Funktion besteht im Wesentlichen im Aufruf von Qt Funktionen. Hierbei kann man also ein bisschen üben, sich in der Qt Dokumentation zurecht zu finden.

5.1.1 Auslesen der Felder

Als erstes müssen wir die Eingabefelder auslesen. Die Eingabefelder sind QLineEdit Objekte. Wir suchen also eine Funktion in der Doku zu QLineEdit, die passen könnte. Wir finden die passenden Funktionen displayText() und text(). Beide Funktionen sind gleich, ein Unterschied ergibt sich lediglich in dem Fall, dass das Eingabefeld zur Eingabe von Passworten verwendet wird. Wir entscheiden uns für text(). Die Syntax für text() ist:

```
QString text() const
```

Man ruft die Funktion also ohne Parameterübergabe auf und bekommt einen QString zurück. Das ”const” am Ende der Syntaxdefinition besagt, daß das Objekt, auf das man die Funktion anwendet (also QLineEdit und damit der darin gespeicherte Text) nicht verändert wird. Wie schon beim Setzen der Default-Werte wird die Funktion über den Operator ”->” aufgerufen, da es sich bei InputA um einen Pointer handelt.

```
QString a;
a = InputA -> text();
```

Jetzt haben wir in dem QString a den eingegebenen Text stehen. Analoges kann man für InputB machen. Um mit den Zahlen rechnen zu können, müssen wir aus dem QString ein Double machen.

5.1.2 QString in Double wandeln

Um eine entsprechende Funktion zu finden, müssen wir jetzt die Doku zu QString nach einer passenden Funktion durchsuchen. Die passende Funktion ist toDouble(). Die Syntax hier ist etwas komplizierter:

```
double QString::toDouble ( bool * ok = 0 ) const
```

Man sieht direkt, dass der Rückgabewert ein double ist. Als Parameter zum Aufruf kann ein Pointer (*) auf eine boolsche Variable mitgegeben werden, die den internen Namen ”ok”

bekommt. Der Parameter ist optional. Das heißt, wenn kein Parameter vorhanden ist, wird der Default-Wert ”0” übergeben. In dieser boolschen Variable finden wir später das Ergebnis, ob die Konvertierung also geklappt hat (ok=true) oder nicht (ok=false). Das ”const” besagt wieder, dass das Objekt, auf das die Funktion angewendet wird, nicht verändert wird. In diesem Fall existiert also der QString, den wir in Double umwandeln, auch nach der Wandlung noch unverändert.

```
double a;
a = (InputA -> text()).toDouble();
```

Hier wird jetzt nicht der ”->” Operator verwendet, da text() einen QString zurückliefert und nicht einen Pointer auf einen QString. Wollen wir die Wandlung prüfen, so müsste der Code so aussehen:

```
double a;
bool ok;
a = (InputA -> text()).toDouble(&ok);
```

ok ist eine boolsche Variable, die Funktion will aber einen Pointer auf eine boolsche Variable haben, sprich die (Speicher-)Adresse, wo die boolsche Variable zu finden ist. Die Adresse einer Funktion wird über den ”&” Operator bestimmt. Wir werden die erste Version ohne Prüfung verwenden, auch wenn es schlechter Stil ist. Unser Code sieht also dann wie folgt aus:

```
double a, b, c;
a = (InputA -> text()).toDouble();
b = (InputB -> text()).toDouble();
c = a + b;
```

Als nächstes muss c in das QLineEdit Feld ResultC geschrieben werden. Das Problem ist, dass die bekannte Funktion setText() einen QString als Parameter erwartet und kein Double. Es muss also wieder eine Umwandlung vorgenommen werden.

5.1.3 Wandeln von Double nach QString

Da wir wieder etwas mit einem QString machen wollen, müssen wir wieder die Doku zu QString durchsuchen. Die Funktion, die wir benötigen, lautet: arg(). Die Funktion arg() gibt es in der Dokumentation zu QString mehrfach. Dies sind so genannte überladene Funktionen. Der Compiler sucht sich an Hand der Parameter (Anzahl und Typen) die passende Funktion heraus. Wir müssen also die richtigen Parameter übergeben. Die Funktion, die wir brauchen, ist

```
QString QString::arg ( double a, int fieldWidth = 0, char format = 'g',
                       int precision = -1,
                       const QChar & fillChar = QLatin1Char( ' ' ) ) const
```

Warum gerade diese? Nunja, das ist die einzige, die als Übergabeparameter ein Double erlaubt. Die Funktionssyntax sieht relativ kompliziert aus. Der einzige verpflichtende Parameter ist jedoch ”double a”, alle anderen Übergabeparameter sind optional, da dort ein Defaultwert angegeben ist. Hierbei ist zu beachten, dass man bei der Übergabe eines Parameters, der vom Defaultwert abweichen soll, alle anderen Parameter ”links” von ihm ebenfalls übergeben muss, selbst wenn diese ihre Defaultwerte behalten sollen. Das ”const” ganz am

Ende der Funktionsdefinition kennen wir ja bereits: Es besagt, dass der QString, auf den die Funktion angewendet wird, selber nicht verändert wird. Der zurückgegebene QString ist also ein neu erzeugter QString.

double a: Der Wert, der umgewandelt werden soll.

int fieldWidth: Der Wert gibt an, wie viele Stellen mit dem Parameter fillChar aufgefüllt werden. (Da es sich um eine überladene Funktion handelt, ist der Parameter nicht an dieser Stelle in der Doku erklärt sondern bei der arg() Funktion, wo er zum ersten Mal auftritt.)

char format: Gibt an, in welcher Schreibweise die Zahl umgewandelt werden soll, also z.B. mit Exponenten.

int precision: Gibt an, wie viele Stellen nach dem Komma dargestellt werden sollen.

const QChar &: Gibt das Zeichen an, welches zum Auffüllen verwendet wird, wenn field-Width ungleich Null ist.

Wir wollen Format ”f” (floating point) verwenden mit 4 Nachkommastellen. Unser Aufruf ist also arg(a+b,0,’f’,4). Damit ist unser Code wie folgt:

```
double a, b;
a = (InputA -> text()).toDouble();
b = (InputB -> text()).toDouble();
ResultC -> setText(QString("%1").arg(a+b,0,'f',4));
```

%1 ist ein Platzhalter, welcher durch die arg Funktion ersetzt wird. Auch diese Erklärung findet man weiter oben in der Doku, wo sie zum ersten Mal benutzt wird. Es lohnt sich also bei Funktionen, die mehrfach vorkommen, auch die Dokumentation von Versionen anzuschauen, die man eigentlich nicht braucht. Was wir nicht vergessen dürfen ist das Einbinden der QString libary in der die Funktion enthalten ist. Das Einbinden erfolgt hierbei mit #include <QString>

Die Datei Taschenrechner.cpp sieht damit so aus:

```
//--- Taschenrechner.cpp - start ---

#include "Taschenrechner.h"
#include <QString>

Taschenrechner::Taschenrechner(QMainWindow *parent) : QMainWindow(parent){
        setupUi(this);
        InputA -> setText("0");
        InputB -> setText("0");

        connect(Calculate, SIGNAL (clicked()), this, SLOT(addAB()));
}

Taschenrechner::~Taschenrechner(){
}

void Taschenrechner::addAB(){
        double a, b;
        a = (InputA -> text()).toDouble();
        b = (InputB -> text()).toDouble();
        ResultC -> setText(QString("%1").arg(a+b,0,'f',4));
}

//--- Taschenrechner.cpp - end ---
```

Nach dem erneuten Übersetzen ist der Taschenrechner fertig zum Addieren. Man könnte jetzt sehr einfach eine Funktion subAB() und einen zusätzlichen Button implementieren, um die Subtraktion einzuführen. Hier soll jedoch einen Schritt weiter gegangen werden: Das Ganze soll übersichtlich in verschiedenen Quellcodedateien implementiert werden.

6 Weitere UI-Erweiterungen

6.1 UI File zur Subtraktion

Um den Taschenrechner nicht nur zur Addition, sondern auch zur Subtraktion verwenden zu können, müssen einige Änderungen vorgenommen werden. Das UI soll so aussehen, dass es ein Fenstermenü gibt, über das man die gewünschte Rechenoperation auswählen kann. Je nachdem, welche Operation gewählt wird, soll sich der Inhalt des UIs (Labels, Eingabefelder und der Button) ändern. Das Wechseln des UIs beim Auswählen des entsprechenden Menüeintrags geschieht über ein so genanntes "Stacked Widget". Ein Stacked Widget kann man sich wie einen Block Papier vorstellen und je nach dem, welche Seite gewählt wird, sieht man eine entsprechende Ansicht.

6.1.1 Änderungen im UI-File

Das bestehende[1] UI-File kann als Grundlage genommen werden. Hierin wird ein neues StackedWidget-Element erstellt. Alle schon vorhandenen UI-Elemente werden auf dessen erste Seite verschoben, danach werden die Elemente nochmals auf die zweite Seite kopiert. Dabei wird noch das Textlabel "A+B" umbenannt in "A-B". Durch das Kopieren ändern sich die technischen Namen der kopierten Objekte: Sie heißen wie die ursprünglichen Objekte, enden aber mit "_2". Will man die Objekte hier also manuell anlegen, sollte dieser Name gewählt werden. Der technische Name des Stacked Widget in unserem Tutorial ist "stackedWidget". Geht man im Qt-Designer in den Preview-Modus (Strg+R), so sieht man für das Stacked Widget ein paar kleine Pfeile, mit denen man die verschiedenen Ansichten durchblättern kann.

Als letztes muss nun noch das Menü erstellt werden. Sollte der UI Designer keine leere Menüleiste am oberen Ende des Reißbrettes anzeigen, kann man durch Klick auf eine leere Stelle im Hintergrund das Kontextmenü aufrufen und "Create Menu Bar" wählen. Einträge können direkt durch einen Mausklick auf die Menüleiste hinzugefügt werden. Das Hauptmenü soll aus den zwei Einträgen "Main" und "Choose" bestehen. Dabei enthält das erste Hauptmenü nur den Eintrag "Quit" zum Beenden des Programms und das zweite Menü die Einträge "Add" und "Sub". Die technischen Namen der Einträge sind per Default "action<Menüeintrag>", also z.B. "actionQuit". Änderungen an diesen Namen können über den Property Editor vorgenommen werden. Eventuell zeigt der Property Editor nicht die richtigen Daten an, wenn man auf den gewünschten Menüeintrag klickt. In diesem Fall müsste man über den Object Inspector gehen. Dieser kann über das Menü im Qt Designer eingeschaltet werden.

1 `https://de.wikibooks.org/wiki/Qt_f%C3%BCr_C%252B%252B_Anf%C3%A4nger%3A_Das_Grundger%C3%BCst`

Man kann mit dem neuen UI-File das Programm nun wieder übersetzen und ausführen. Man sollte die Menüleiste und deren Einträge sehen aber sonst noch keine Veränderungen.

6.1.2 Menüs mit Funktionen belegen

Nun sollen die drei Menüs mit Funktionen belegt werden. Das ist ganz einfach. Wenn man in der Doku ein wenig sucht, findet man die passenden Funktionen. Hier wird nun eine etwas kürzere Beschreibung gegeben als im Kapitel zuvor. Für jeden Menüeintrag muss ein connect String in den Konstruktor eingebaut werden. Der connect String hat die Form

```
connect(action<Menüeintrag>,SIGNAL (triggered()), this, SLOT(<Funktion>));
```

action<Menüeintrag> ist dabei der technische Name des Menüs und <Funktion> ist eine zu implementierende Funktion, die gerufen werden soll, wenn der Benutzer den entsprechenden Eintrag auswählt. Für jede dieser Funktionen muss im Header-File Taschenrechner.h eine entsprechende Deklaration eingeführt und diese in der Quelldatei Taschenrechner.cpp implementiert werden.

6.1.3 Menü Quit

Der connect String sieht hier so aus

```
connect(actionQuit,SIGNAL (triggered()), this, SLOT(slotClose()));
```

Anzumerken sei hier, dass der Slot "slotClose()" genannt wurde, man mag geneigt sein, ihn einfach nur "close()" zu nennen. close() ist aber eine Funktion, die es in Qt schon gibt. Die Benutzung von close() als Slotname würde zu einem unerwünschten Verhalten führen. Generell sollte man bei der Wahl der Funktionsnamen etwas aufpassen und zu "simple" Namen meiden. Weitere Beispiele von Funktionsnamen, die nicht verwendet werden können, sind add und sub. Add und Sub könnten aber benutzt werden. Die Groß- und Kleinschreibung spielt hier also eine Rolle. Die Deklaration der Funktion slotClose() ist sehr einfach:

```
void slotClose();
```

Die Implementation der Funktion ist ebenfalls sehr simpel

```
void Taschenrechner::slotClose(){
        close();
}
```

Zur Erinnerung: ganz am Anfang[2] hatten wir in main.cpp eine Code-Zeile "return a.exec()" implementiert. Wenn jetzt close() aufgerufen wird, schließt sich das aktuelle Fenster. Ist das aktuelle Fenster das letzte Fenster, so endet a.exec() und das Programm ebenfalls. Der return-Befehl gibt den Status von a.exec() zurück. Die Funktion close() bekommt QApplication von QWidget vererbt, ist also unter QWidget zu finden. Möchte man nicht nur das Fenster schließen, sondern die ganze Applikation beenden (falls noch andere Fenster offen sind), so gibt es die Funktionen QCoreApplication::quit() und QCoreApplication::exit().

2 https://de.wikibooks.org/wiki/Qt_f%C3%BCr_C%252B%252B_Anf%C3%A4nger%3A_Das_Grundger%C3%BCst%23main.cpp

Nun kann die Applikation wieder neu übersetzt und dann geprüft werden, ob "Quit" wirklich die Anwendung schließt. Die Funktion slotClose() ist natürlich sehr kurz, da nur close() aufgerufen wird, man kann natürlich einen Slot auch direkt in dem connect String verarbeiten.

```
connect(actionQuit,SIGNAL (triggered()), qApp, SLOT(quit()));
```

qApp ist ein Makro, welches die aktuelle Instanz von QApplication herausfindet.

6.1.4 Menü Add

Der connect String sieht wie folgt aus

```
connect(actionAdd, SIGNAL(triggered()), this, SLOT(showAdd()));
```

Die Deklaration von showAdd() wie folgt

```
void showAdd();
```

Für die Implementation von showAdd() muss man sich überlegen wie man zwischen den Seiten des stacked Widget umschalten kann. In der Doku findet man die passende Funktion, damit sieht die Implementation wie folgt aus:

```
void Taschenrechner::showAdd(){
        stackedWidget -> setCurrentIndex(0);
}
```

Man beachte, daß die Zählung der Seiten bei Null beginnt.

6.1.5 Menü Sub

Das Menü Sub wird implementiert wie das Menü Add, man ersetze einfach Add durch Sub. Ferner muss darauf geachtet werden, dass der Index von Null auf Eins geändert wird, damit auch wirklich die richtige Seite angezeigt wird.

Beim folgenden Coding ist noch zu beachten, dass sich main.cpp nicht geändert hat und dass analog zum vorherigen Kapitel noch Defaultwerte für die Sub-Seite gesetzt wurden.

```
//--- Taschenrechner.h - start ---

#ifndef TASCHENRECHNER_H
#define TASCHENRECHNER_H

#include "ui_Taschenrechner.h"

class Taschenrechner : public QMainWindow, public Ui::MainWindow{
        Q_OBJECT

public:
        Taschenrechner (QMainWindow *parent = 0);
        ~Taschenrechner();
private slots:
        void addAB();
        void slotClose();
        void showAdd();
        void showSub();
```

```
};

#endif //TASCHENRECHNER_H

//--- Taschenrechner.h - end ---


//--- Taschenrechner.cpp - start ---
#include "Taschenrechner.h"

Taschenrechner::Taschenrechner(QMainWindow *parent) : QMainWindow(parent){
        setupUi(this);
        InputA -> setText("0");
        InputB -> setText("0");
        InputA_2 -> setText("0");
        InputB_2 -> setText("0");

        connect(Calculate, SIGNAL (clicked()), this, SLOT(addAB()));
        connect(actionQuit,SIGNAL (triggered()), this, SLOT(slotClose()));
        connect(actionAdd, SIGNAL(triggered()), this, SLOT(showAdd()));
        connect(actionSub, SIGNAL(triggered()), this, SLOT(showSub()));
}

Taschenrechner::~Taschenrechner(){
}

void Taschenrechner::addAB(){
        double a, b;
        a = (InputA -> text()).toDouble();
        b = (InputB -> text()).toDouble();
        ResultC -> setText(QString("%1").arg(a+b,0,'f',4));
}

void Taschenrechner::slotClose(){
        close();
}

void Taschenrechner::showAdd(){
        stackedWidget -> setCurrentIndex(0);
}

void Taschenrechner::showSub(){
        stackedWidget -> setCurrentIndex(1);
}

//--- Taschenrechner.cpp - end ---
```

Zu guter Letzt könnte man jetzt eine Funktion subAB implementieren und fertig wäre der Taschenrechner. Da dies identisch ist mit der Implementierung von addAB(), kann auf eine weitere Erklärung hier verzichtet werden. Wenn man dies macht, wird die Datei Taschenrechner.cpp sehr schnell sehr unübersichtlich (es gibt ja noch mehr Rechenarten zu implementieren). Daher soll im folgenden Kapitel der Quellcode auf verschiedene Dateien verteilt werden.

7 Aufteilung des Quellcodes

Der Quellcode soll jetzt auf mehrere Dateien verteilt werden, um eine bessere Übersicht zu behalten. Dazu sollen die Funktionen zum Addieren und zum Subtrahieren in einer jeweils eigenen Datei liegen. Ferner wird es eine Datei geben, die für übergreifende Funktionen (z.B. das UI) zuständig ist. Jede Funktion wird in eine eigene Klasse ausgegliedert, jede Klasse besteht aus zwei Dateien:

```
<Klassenname>.cpp
<Klassenname>.h
```

Insgesamt werden wir die folgenden Dateien haben:

```
myAdd.cpp
myAdd.h

mySub.cpp
mySub.h

Taschenrechner.cpp
Taschenrechner.h

main.cpp

Taschenrechner.ui
```

Eine Anmerkung sei gemacht, die neuen Klassen wurden myAdd und mySub genannt. Auch hier muss wieder mit der Namensgebung aufgepasst werden, da bestimmte Namen schon belegt sind. Wir hätten sie nicht add und sub, sehr wohl aber Add und Sub nennen können. Die Dateien myAdd.cpp/.h und mySub.cpp/.h sehen natürlich sehr ähnlich aus und wir werden nur die Klasse myAdd im Detail besprechen, für mySub wird der Quellcode ohne weiteren Kommentar zur Verfügung gestellt.

7.1 myAdd.h

Hierbei handelt es sich wieder um eine Headerdatei, die im Prinzip nichts Ungewöhnliches enthält.

```
//--- myAdd.h - start ---
#ifndef MYADD_H
#define MYADD_H

#include <QWidget>

class Taschenrechner;
```

```
class myAdd : public QWidget{
        Q_OBJECT

Taschenrechner *myTaschenrechner;

public:
        myAdd (Taschenrechner*);
        ~myAdd();
private slots:

};

#endif //MYADD_H
//--- myAdd.h - end ---
```

Die Klasse myAdd ist abgeleitet von der Klasse QWidget. Dies ist notwendig, damit die Klasse myAdd alle grundsätzlichen Qt Eigenschaften besitzt. Es gibt eine Klassenvariable myTaschenrechner von Typ Taschenrechner, die unsere Hauptklasse ist. Damit diese Variable erzeugt werden kann, muss der Typ Taschenrechner deklariert werden, sprich der Compiler muss wissen, dass es eine Klasse Taschenrechner gibt. Dies passiert mittels der forward declaration

```
class Taschenrechner;
```

Diese Variable muss beim Anlegen einer Instanz der Klasse myAdd belegt werden. Erzwungen wird das, da im Konstruktor von myAdd diese Variable übergeben werden muss. Notwendig ist dies, da wir aus der Klasse myAdd heraus auf Objekte (=UI Elemente) der Klasse Taschenrechner zugreifen wollen. Diese UI Elemente liegen gemäß unseres Designs (die Qt Designer ui-Datei wird aus Taschenrechner.cpp heraus dargestellt) in dieser Klasse und könnten ohne Zugriff auf diese Klasse nicht gefunden/angesprochen werden.

7.2 myAdd.cpp

Im Grundgerüst der Klasse myAdd werden erstmals nur der Konstruktor und der Destruktor implementiert. Für den Konstruktor muss überlegt werden, wie der Pointer auf die Klasse Taschenrechner übergeben werden kann. Eine Möglichkeit ist es, beim Aufruf des Konstruktors den Wert an einen Pointer TRechner zu übergeben und diesen dann an myTaschenrechner zu übergeben.

```
//--- myAdd.cpp - start ---
#include "myAdd.h"
#include "Taschenrechner.h"

myAdd::myAdd(Taschenrechner *TRechner){
        myTaschenrechner = TRechner;
}

myAdd::~myAdd(){
}
//--- myAdd.cpp - end ---
```

Es sei angemerkt, dass etwas in der Art von

```
myAdd::myAdd(Taschenrechner *myTaschenrechner){
}
```

nicht funktioniert, man muss also wirklich eine Variable übergeben und diese dann weiter reichen. Das ist sicherlich nicht sehr elegant, daher gibt es eine andere Möglichkeit Klassenvariablen zu initialisieren. Dabei handelt es sich um die so genannten Initialisierungslisten.

```
//--- myAdd.cpp - start ---
#include "myAdd.h"
#include "Taschenrechner.h"

myAdd::myAdd(Taschenrechner *TRechner):myTaschenrechner(TRechner){
}

myAdd::~myAdd(){
}
//--- myAdd.cpp - end ---
```

Man erkennt im Code wohl, wie diese Liste funktioniert, die Variable wird mit einem ":" angehangen und der Wert zur Initialisierung wird in runden Klammern übergeben. Auf diese Art und Weise können auch mehrere Variablen initialisiert werden.

7.2.1 Erzeugen von Instanzen der neuen Klassen

Die neuen Klassen wurden jetzt definiert. Sie haben bisher noch keine Funktionen implementiert. Das wird später kommen. Die Frage ist jetzt noch, wie man diese Klassen wirklich nutzt. Nur vom Erstellen der Header und cpp Files werden die Klassen noch nicht verwendet. Man muss im bisherigen Coding eine Instanz der neuen Klassen anlegen, dies geschieht über den "new" Operator innerhalb des Konstruktors der Klasse Taschenrechner.

```
myAdd *addWidget = new myAdd(this);
mySub *subWidget = new mySub(this);
```

Hier wird ein Pointer addWidget/subWidget vom Typ myAdd/mySub angelegt (myAdd *addWidget). Der Pointer wird durch die Funktion myAdd() initialisiert. Dazu muss bei Klassen der Operator "new" verwendet werden im Gegensatz zu intrinsischen Datentypen, z.B. int a = 1, bei denen kein "new" verwendet wird. Die Funktion myAdd() ist der Konstruktor der Klasse myAdd. So, wie diese Klasse definiert ist, muss immer ein Parameter übergeben werden. Dieser Parameter ist der Pointer auf die Instanz der Klasse Taschenrechner. Diesen Pointer kann man im Konstruktor mit der Anweisung "this" übergeben.

Wichtig: Damit diese Funktionen in Taschenrechner.cpp bekannt sind, darf man nicht vergessen, die Header-Dateien der Klassen mittels #include einzubinden.

Bevor jetzt der Quellcode mittels make übersetzt werden kann, muss berücksichtigt werden, dass neue Dateien erzeugt wurden. Diese sind im Makefile nicht angegeben, so dass es beim Übersetzen zu einer Fehlermeldung kommen wird. Einfach mal ausprobieren. Damit das Makefile angepasst wird, nochmals

```
qmake -project
qmake
```

ausführen. Jetzt sollte das Programm mittels make übersetzt werden können. Es ist allerdings noch keine neue Funktionalität enthalten. Weiterhin wird eine Warnung ausgegeben, dass addWidget und subWidget nicht benutzt werden. Diese Warnung kann ignoriert werden, da wirklich mit den Objekten noch nichts passiert.

8 Finale Implementierung

Nun müssen die Funktionen zum Addieren und Subtrahieren noch in die anderen Dateien verschoben werden. Als erstes können aus der Klasse Taschenrechner die Deklaration und Implementierung der Methode addAB() entfernt werden, da diese nicht mehr gebraucht werden.

Aus Taschenrechner.h muss nur die Deklaration der Funktion addAB() entfernt werden, ansonsten ändert sich nichts.

```
//--- Taschenrechner.h - start ---
#ifndef TASCHENRECHNER_H
#define TASCHENRECHNER_H

#include "ui_Taschenrechner.h"

class Taschenrechner : public QMainWindow, public Ui::MainWindow{
        Q_OBJECT

public:
        Taschenrechner (QMainWindow *parent = 0);
        ~Taschenrechner();
private slots:
        void showAdd();
        void showSub();
        void slotClose();
};

#endif //TASCHENRECHNER_H
//--- Taschenrechner.h - end ---
```

In Taschenrechner.cpp müssen zwei Änderungen vorgenommen werden. Die Datei sieht nun wie folgt aus:

```
//--- Taschenrechner.cpp - start ---
#include "Taschenrechner.h"
#include "myAdd.h"
#include "mySub.h"

Taschenrechner::Taschenrechner(QMainWindow *parent) : QMainWindow(parent){
        setupUi(this);

        myAdd *addWidget = new myAdd(this);
        mySub *subWidget = new mySub(this);

        InputA -> setText("0");
        InputB -> setText("0");
        InputA_2 -> setText("0");
        InputB_2 -> setText("0");



        connect(Calculate, SIGNAL(clicked()), addWidget, SLOT(addAB()));
        connect(Calculate_2, SIGNAL(clicked()), subWidget, SLOT(subAB()));
        connect(actionQuit, SIGNAL(triggered()), qApp, SLOT(quit()));
```

```
        connect(actionAdd, SIGNAL(triggered()), this, SLOT(showAdd()));
        connect(actionSub, SIGNAL(triggered()), this, SLOT(showSub()));
}

Taschenrechner::~Taschenrechner(){
}

void Taschenrechner::showAdd(){
        stackedWidget -> setCurrentIndex(0);
}

void Taschenrechner::showSub(){
        stackedWidget -> setCurrentIndex(1);
}
//--- Taschenrechner.cpp - end ---
```

Als erstes wurde die Implementierung der Funktion addAB() entfernt. Als nächsten Schritt müssen die connect Anweisungen angepasst werden. Diese sehen nun so aus:

```
connect(Calculate, SIGNAL (clicked()), addWidget, SLOT(addAB()));
connect(Calculate_2, SIGNAL (clicked()), subWidget, SLOT(subAB()));
```

Wenn der QPushButton ”Calculate” das Signal clicked() erhält, ist der Empfänger nicht mehr ”this” sondern das neu angelegte Objekt addWidget. Dies geschieht, da nun gerade in addWidget die Funktion zum Addieren enthalten ist. Der Slot heißt wieder addAB(), ist aber ein Slot der Klasse myAdd. Für die Subtraktion ist der connect String im wesentlichen identisch, der QPushButton hier hat einen anderen Namen, da es technisch ein anderer Knopf ist. Empfänger ist hier die Funktion subAB() der Instanz subWidget der Klasse mySub.

Die Anpassungen in myAdd.h und myAdd.cpp sind im wesentlichen wieder identisch wie die in mySub.h und mySub.cpp. myAdd.h hat nun folgendes Aussehen:

```
//--- myAdd.h - start ---
#ifndef MYADD_H
#define MYADD_H

#include <QWidget>

class Taschenrechner;

class myAdd : public QWidget{
        Q_OBJECT

Taschenrechner *myTaschenrechner;

public:
        myAdd (Taschenrechner*);
        ~myAdd();
private slots:
        void addAB();
};

#endif //MYADD_H
//--- myAdd.h - end ---
```

Einzig die Deklaration des Slots addAB() ist eingefügt worden. In myAdd.cpp muss diese Funktion als Nächstes implementiert werden:

```
//--- myAdd.cpp - start ---
#include "myAdd.h"
```

```
#include "Taschenrechner.h"


myAdd::myAdd(Taschenrechner *TRechner):myTaschenrechner(TRechner){
}


myAdd::~myAdd(){
}

void myAdd::addAB(){
        double a, b;
        a = (myTaschenrechner -> InputA -> text()).toDouble();
        b = (myTaschenrechner -> InputB -> text()).toDouble();
        myTaschenrechner -> ResultC -> setText(QString("%1").arg(a+b,0,'f',4));
}
//--- myAdd.cpp - end ---
```

Die Implementierung ist im Prinzip identisch mit der Implementierung, als dieser Slot noch zur Klasse Taschenrechner gehörte. Das einzige Problem, welches sich ergibt, ist, dass die ganzen UI Objekte zur Klasse Taschenrechner gehören und hier nicht bekannt sind. Die Instanz der Klasse Taschenrechner, die hier benutzt wird, wurde beim Erzeugen von addWidget angegeben und in myTaschenrechner gespeichert. Wenn man nun auf Objekt im UI zugreifen will, kann man dies über

```
myTaschenrechner -> UI Objekt -> Methode des UI Objekts
```

erreichen. Der Trick ist also nicht das UI Objekt direkt anzusprechen, sondern den "Umweg" über eine Instanz der Klassen Taschenrechner zu gehen, diese Instanz heißt hier myTaschenrechner.

Der Taschenrechner sollte sich nun übersetzen lassen und das Addieren und Subtrahieren unterstützen.

9 Autoren

Edits	User
1	Albmont[1]
1	Basit˜dewikibooks[2]
1	Dexbot[3]
10	Dirk Huenniger[4]
6	Erik Streb[5]
8	EwieEmil[6]
1	Gboelter[7]
4	Joebar[8]
1	Juetho[9]
1	Klaus Eifert[10]
1	LivingShadow[11]
1	MazeChaZer[12]
1	Mmorath[13]
1	Mr N[14]
1	Mr. Anderson[15]
1	Nameless23[16]
1	OliverKurz[17]
2	Pb˜dewikibooks[18]
1	Quiethoo[19]
2	Tones29[20]
3	Uncopy[21]

1 https://de.wikibooks.org/wiki/Benutzer:Albmont
2 https://de.wikibooks.org/w/index.php%3ftitle=Benutzer:Basit˜dewikibooks&action=edit&redlink=1
3 https://de.wikibooks.org/wiki/Benutzer:Dexbot
4 https://de.wikibooks.org/wiki/Benutzer:Dirk_Huenniger
5 https://de.wikibooks.org/wiki/Benutzer:Erik_Streb
6 https://de.wikibooks.org/w/index.php%3ftitle=Benutzer:EwieEmil&action=edit&redlink=1
7 https://de.wikibooks.org/w/index.php%3ftitle=Benutzer:Gboelter&action=edit&redlink=1
8 https://de.wikibooks.org/w/index.php%3ftitle=Benutzer:Joebar&action=edit&redlink=1
9 https://de.wikibooks.org/wiki/Benutzer:Juetho
10 https://de.wikibooks.org/wiki/Benutzer:Klaus_Eifert
11 https://de.wikibooks.org/wiki/Benutzer:LivingShadow
12 https://de.wikibooks.org/wiki/Benutzer:MazeChaZer
13 https://de.wikibooks.org/w/index.php%3ftitle=Benutzer:Mmorath&action=edit&redlink=1
14 https://de.wikibooks.org/wiki/Benutzer:Mr_N
15 https://de.wikibooks.org/w/index.php%3ftitle=Benutzer:Mr._Anderson&action=edit&redlink=1
16 https://de.wikibooks.org/w/index.php%3ftitle=Benutzer:Nameless23&action=edit&redlink=1
17 https://de.wikibooks.org/w/index.php%3ftitle=Benutzer:OliverKurz&action=edit&redlink=1
18 https://de.wikibooks.org/wiki/Benutzer:Pb˜dewikibooks
19 https://de.wikibooks.org/wiki/Benutzer:Quiethoo
20 https://de.wikibooks.org/w/index.php%3ftitle=Benutzer:Tones29&action=edit&redlink=1
21 https://de.wikibooks.org/wiki/Benutzer:Uncopy

Abbildungsverzeichnis

- GFDL: Gnu Free Documentation License. http://www.gnu.org/licenses/fdl.html
- cc-by-sa-3.0: Creative Commons Attribution ShareAlike 3.0 License. http://creativecommons.org/licenses/by-sa/3.0/
- cc-by-sa-2.5: Creative Commons Attribution ShareAlike 2.5 License. http://creativecommons.org/licenses/by-sa/2.5/
- cc-by-sa-2.0: Creative Commons Attribution ShareAlike 2.0 License. http://creativecommons.org/licenses/by-sa/2.0/
- cc-by-sa-1.0: Creative Commons Attribution ShareAlike 1.0 License. http://creativecommons.org/licenses/by-sa/1.0/
- cc-by-2.0: Creative Commons Attribution 2.0 License. http://creativecommons.org/licenses/by/2.0/
- cc-by-2.0: Creative Commons Attribution 2.0 License. http://creativecommons.org/licenses/by/2.0/deed.en
- cc-by-2.5: Creative Commons Attribution 2.5 License. http://creativecommons.org/licenses/by/2.5/deed.en
- cc-by-3.0: Creative Commons Attribution 3.0 License. http://creativecommons.org/licenses/by/3.0/deed.en
- GPL: GNU General Public License. http://www.gnu.org/licenses/gpl-2.0.txt
- LGPL: GNU Lesser General Public License. http://www.gnu.org/licenses/lgpl.html
- PD: This image is in the public domain.
- ATTR: The copyright holder of this file allows anyone to use it for any purpose, provided that the copyright holder is properly attributed. Redistribution, derivative work, commercial use, and all other use is permitted.
- EURO: This is the common (reverse) face of a euro coin. The copyright on the design of the common face of the euro coins belongs to the European Commission. Authorised is reproduction in a format without relief (drawings, paintings, films) provided they are not detrimental to the image of the euro.
- LFK: Lizenz Freie Kunst. http://artlibre.org/licence/lal/de
- CFR: Copyright free use.

- EPL: Eclipse Public License. `http://www.eclipse.org/org/documents/epl-v10.php`

Copies of the GPL, the LGPL as well as a GFDL are included in chapter Licenses[22]. Please note that images in the public domain do not require attribution. You may click on the image numbers in the following table to open the webpage of the images in your webbrower.

22 Kapitel 10 auf Seite 37

1	Dirk Huenniger, Joebar	

10 Licenses

10.1 GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS 0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion. 1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work. 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary. 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures. 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee. 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

* a) The work must carry prominent notices stating that you modified it, and giving a relevant date. * b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices". * c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it. * d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate. 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

* a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange. * b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge. * c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b. * d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements. * e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying. 7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

* a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or * b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or * c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or * d) Limiting the use for publicity purposes of names of licensors or authors of the material; or * e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or * f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way. 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10. 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so. 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it. 11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law. 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from

conveying the Program. 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such. 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version. 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

10.2 GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free"in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference. 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Versionöf the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A SSecondary Sectionïs a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The Ïnvariant Sectionsäre certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Textsäre certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent"copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparentïs called Öpaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page"means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page"means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher"means any person or entity that distributes copies of the Document to the public.

A section Ëntitled XYZ"means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as Äcknowledgements", "Dedications", Ëndorsements", or "History".) To "Preserve the Titleöf such a section when you modify the Document means that it remains a section Ëntitled XYZäccording to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License. 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies. 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document. 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

* A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. * B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. * C. State on the Title page the name of the publisher of the Modified Version, as the publisher. * D. Preserve all the copyright notices of the Document. * E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. * F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. * G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. * H. Include an unaltered copy of this License. * I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "Historyïn the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. * J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History§ection. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. * K. For any section Entitled Äcknowledgementsör "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. * L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. * M. Delete any section Entitled Ëndorsements". Such a section may not be included in the Modified Version. * N. Do not retitle any existing section to be Entitled Ëndorsementsör to conflict in title with any Invariant Section. * O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled Ëndorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "Historyïn the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled Äcknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled Ëndorsements". 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document. 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ägg regateïf the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate. 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled Äcknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title. 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it. 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ör any later versionäpplies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document. 11. RELICENSING

"Massive Multiauthor Collaboration Site"(or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration"(or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA"means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

Ïncorporate"means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is ëligible for relicensingïf it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts."line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

10.3 GNU Lesser General Public License

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below. 0. Additional Definitions.

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work. 1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL. 2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

* a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or * b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

* a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

* a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the Combined Work with a copy of the GNU GPL and this license document. * c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document. * d) Do one of the following: o 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source. o 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user’s computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version. * e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

* a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License. * b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy’s public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.