Thema, Ziele: automatisierte Unit Tests

Die Vorlage für alle folgenden Aufgaben ist im Ordner *Vorlage* dieses Praktikums zu finden.

Aufgabe 0: Qt Projekt inkl. googletest für Till Unit Test einrichten

Für den automatisierten Test müssen Sie zuerst ein neues Qt Projekt mit dem Namen *RabattTest* erstellen. Anders als gewohnt reicht hier das Projekt Template *Qt Console Application*. Anschliessend soll die googleTest Library zum Projekt hinzugefügt werden.

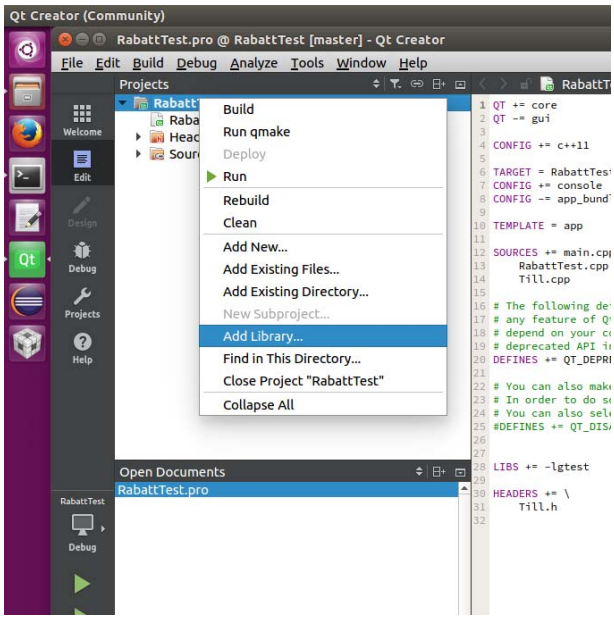

Abbildung 1 Add Library Kontextmenü in QtCreator

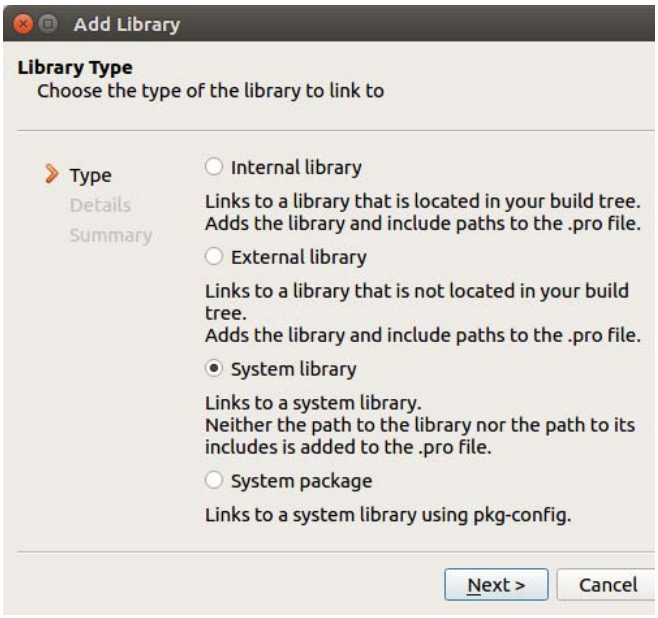

Abbildung 2 System Library auswählen

Die folgende Anleitung zeigt, wie die GoogleTest Library zum Projekt hinzugefügt wird:

Machen Sie zuerst ein Rechtsklick auf Ihrem Projekt (siehe Abbildung 1). Nach Abbildung 2 drücken Sie auf *Next*. Wählen Sie die *gtest* Library aus. Sie finden Sie im GoogleTest Ordner im Pfad */usr/local/lib/libgtest.a*.

Der Pfad zu den Headerfiles ist für System Libraries bereits hinterlegt. Hier müssen keinen zusätzlichen Pfad angeben.

Dem Projekt kann die Klasse *Till* hinzugefügt werden. Dies kann über *Add Existing Files…* über das Kontextmenü in Abbildung 1 gemacht werden.

Das Test main.cpp File könnte wie folgt aussehen:

```
///
/// \file   main.cpp
/// \date   30.11.2016
/// \author Michael Trummer
///

#include "VoltageDividerTest.h"
#include "EDecadeTest.h"
#include "VoltageDividerWidgetTest.h"

#include <gtest/gtest.h>

int main(int argc, char *argv[])
{
  QApplication app(argc, argv); //wird bei Verwendung von QWidgets benötigt

  ::testing::InitGoogleTest(&argc, argv);
  return RUN_ALL_TESTS();
}
```

Aufgabe 1: Till Unit Test schreiben

Erstellen Sie ein weiteres Cpp File mit dem Namen *TillTest.cpp* und fügen Sie es zum Projekt hinzu.
Der Inhalt der Datei kann wie folgt beginnen:

```
#include <gtest/gtest.h>
#include "Till.h"
TEST(TillTest, rabatt_iec11_1)
{
  EXPECT_DEATH(Till::rabatt(10001, 0, 0), "");
}

// … weitere TESTs…
```

Schreiben Sie den Unit Test für die *Till* Klasse. Da die *Till* Klasse nur die Rabatt Funktion enthält, beschränkt sich dieser Unit Test auf das Testen der `rabatt` Funktion.
Implementieren Sie alle Testfälle, die im letzten Praktikum für die `rabatt` Funktion mittels Äquivalenzklassen und Grenzwertanalyse erstellt haben.

Aufgabe 2: Till Unit Test Code Coverage

Nachdem das Test-Programm von Aufgabe 1 ausgeführt werden kann, kann die Coverage mittels folgenden Schritten angezeigt werden:

- Mit Qt kann das Coverage Tool gcov in Zusammenhang mit lcov genutzt werden. Um dieses Tools einfach zu nutzen ist ein Script mit dem Namen *generateCoverageView.sh* im Übungsverzeichnis abgelegt. Dieses kann verwendet werden, um direkt eine HTML-Seite mit den Coverage-Informationen zu erhalten.
- Im .pro-File müssen die entsprechenden Flags gesetzt werden und die gcov Library muss dazu gelinkt werden. Das sieht wie folgt aus:

  #GCOV Settings

  QMAKE_CXXFLAGS += -fprofile-arcs -ftest-coverage

  QMAKE_LDFLAGS += -fprofile-arcs -ftest-coverage

  LIBS += -lgcov

- Abschliessend kann das Script wie folgt aufgerufen werden

  ./generateCoverageView.sh RabattTest

Im Firefox wird Ihnen die Coverage von beteiligten File angezeigt. Überprüfen Sie die Coverage von der Klasse *Till*. Die Coverage wird nicht 100 % betragen. Ergänzen Sie die Tests für die *Till* Klasse, damit Sie 100 % Coverage erreichen.

Aufgabe 3: Author Mock für Unit Test in Library Projekt

Bei der nächsten Aufgabe soll die Klasse *Book* (siehe Vorlageordner) mittels Unit Test getestet werden. Damit *Book* ohne Abhängigkeiten zu der Klasse *Author* getestet werden kann, muss *Author* mittels Interface *IAuthor* zu Book getrennt werden.

Erstellen Sie das Interface und passen Sie *Author* so an, dass die Klasse *Author* von *IAuthor* ableitet. Übergeben Sie dem Ctor von *Book* neu eine Referenz auf ein *IAuthor* Object anstatt einem *Author* Object.

Bringen Sie das Projekt entsprechend wieder zum Laufen, damit der gleiche Konsolenoutput wie vorher ausgegeben wird.

Aufgabe 4: Book Unit Test in Library Projekt

Erstellen Sie einen Author Mock mit dem Namen *AuthorMock* im Test Projektpfad. Die Klasse *AuthorMock* soll von *IAuthor* ableiten.
Erstellen Sie anschliessend Anweisungsüberdeckungs-Testfälle für die Klasse *Book*. Verwenden Sie ein Objekt von *AuthorMock* anstatt von Author.

Weitere Informationen: Verwenden von gcov in Eclipse

In Eclipse ist gcov bereits mittels Plugin eingebunden. Es kann über das in der untenstehenden Abbildung markierte Icon konfiguriert werden (siehe Abbildung 3).

Abbildung 3 Icon um Profile zu konfigurieren

Danach *Profile Configurations* auswählen.

In der linken Spalte kann das gewünschte Projekt ausgewählt werden, und im Tab *Profile* kann *gcov* ausgewählt werden (siehe Abbildung 4).

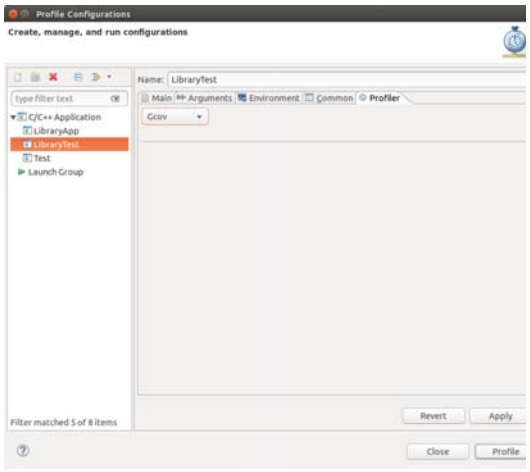

Abbildung 4 Gcov als Profile auswählen

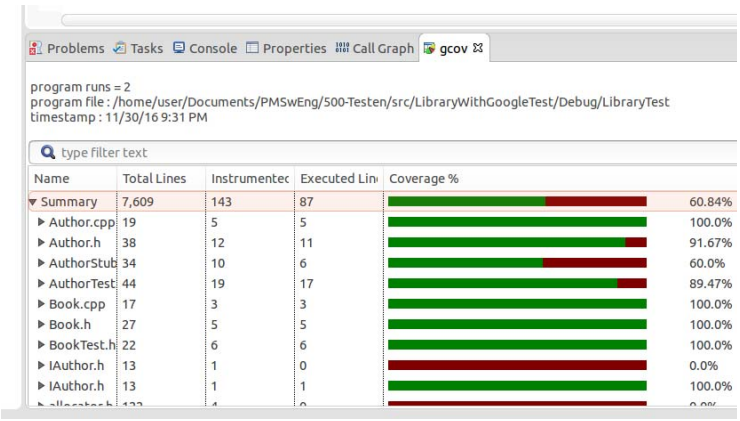

Abbildung 5 Gcov coverage output

Nach dem Bestätigen wird das Programm inklusive *gcov* ausgeführt. Es erscheint ein neuer Tab im unter Bereich, welcher das Resultat von *gcov* präsentiert (siehe Abbildung 5).

Werden die Files über den *Project Explorer* im linken Bereich von Eclipse ausgewählt, werden die Codezeilen entsprechend der Anweisungsüberdeckung eingefärbt (siehe Abbildung 6)

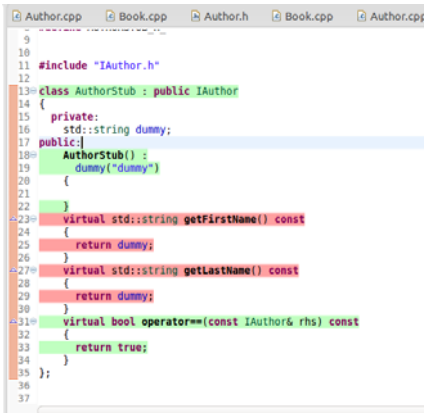

Abbildung 6 Eingefärbte Codezeile durch gcov