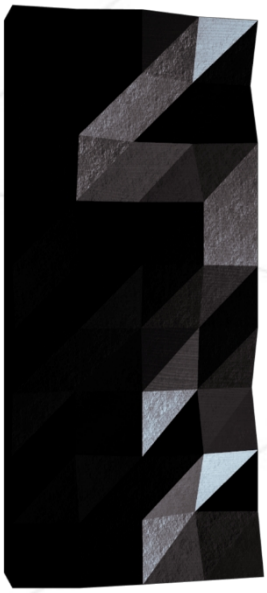

C++ - Modul 05

Wiederholungen und Ausnahmen

Zusammenfassung:
Dieses Dokument enthält die Übungen des Moduls 05 aus den C++ Modulen.

Version: 10.1

Inhalt

Kapitel I

Einleitung

C++ ist eine allgemeine Programmiersprache, die von Bjarne Stroustrup als Weiterentwicklung der Programmiersprache C oder "C with Classes" (Quelle: Wikipedia) entwickelt wurde.

Das Ziel dieser Module ist es, Sie in die **objektorientierte Programmierung** einzuführen. Dies wird der Ausgangspunkt für Ihre C++-Reise sein. Viele Sprachen werden empfohlen, um OOP zu lernen. Wir haben uns für C++ entschieden, da es von Ihrem alten Freund C abgeleitet ist. Da es sich um eine komplexe Sprache handelt, und um die Dinge einfach zu halten, wird Ihr Code dem Standard C++98 entsprechen.

Wir sind uns bewusst, dass modernes C++ in vielerlei Hinsicht anders ist. Wenn Sie also ein kompetenter C++-Entwickler werden wollen, liegt es an Ihnen, nach den 42 Common Core weiterzugehen!

Kapitel II

Allgemeine Vorschriften

Kompilieren

- Kompilieren Sie Ihren Code mit C++ und den Flags -Wall -Wextra -Werror
- Ihr Code sollte sich trotzdem kompilieren lassen, wenn Sie das Flag -std=c++98 hinzufügen

Formatierungs- und Benennungskonventionen

- Die Übungsverzeichnisse werden folgendermaßen benannt: ex00, ex01, ... , exn
- Benennen Sie Ihre Dateien, Klassen, Funktionen, Mitgliedsfunktionen und Attribute wie in den Richtlinien gefordert.
- Schreiben Sie Klassennamen im Format **UpperCamelCase**. Dateien, die Klassencode enthalten, werden immer nach dem Klassennamen benannt. Zum Beispiel: Klassenname.hpp/Klassenname.h, Klassenname.cpp, oder Klassenname.tpp. Wenn Sie also eine Header-Datei haben, die die Definition einer Klasse "BrickWall" enthält, die für eine Ziegelmauer steht, lautet ihr Name BrickWall.hpp.
- Wenn nicht anders angegeben, müssen alle Ausgabemeldungen mit einem Zeilenumbruch abgeschlossen und auf der Standardausgabe ausgegeben werden.
- *Auf Wiedersehen Norminette!* In den C++-Modulen ist kein Kodierungsstil vorgeschrieben. Sie können Ihrem Lieblingsstil folgen. Aber denken Sie daran, dass ein Code, den Ihre Mitbewerber nicht verstehen, auch nicht benotet werden kann. Geben Sie Ihr Bestes, um einen sauberen und lesbaren Code zu schreiben.

Erlaubt/Verboten

Sie programmieren nicht mehr in C. Zeit für C++! Deshalb:

- Sie dürfen fast alles aus der Standardbibliothek verwenden. Anstatt sich also an das zu halten, was Sie bereits kennen, wäre es klug, so viel wie möglich die C++-ähnlichen Versionen der C-Funktionen zu verwenden, an die Sie gewöhnt sind.
- Sie können jedoch keine andere externe Bibliothek verwenden. Das bedeutet, dass C++11 (und abgeleitete Formen) und Boost-Bibliotheken verboten sind. Die

folgenden Funktionen sind ebenfalls verboten: *printf(), *alloc() und free(). Wenn Sie sie verwenden, wird Ihre Note 0 sein und das war's.

- Beachten Sie, dass, sofern nicht ausdrücklich anders angegeben, der using-Namensraum <ns_name> und
  Freundschaftswörter sind verboten. Andernfalls wird Ihre Note -42 sein.
- **Du darfst die STL nur in den Modulen 08 und 09 verwenden.** Das bedeutet: keine **Container** (Vektor/Liste/Map/usw.) und keine **Algorithmen** (alles, was den <algorithm>-Header erfordert) bis dahin. Andernfalls wird Ihre Note -42 sein.

Einige Designanforderungen

- Speicherlecks treten auch in C++ auf. Wenn Sie Speicher zuweisen (mit der Funktion new
  Schlüsselwort), müssen Sie **Speicherlecks** vermeiden.
- Von Modul 02 bis Modul 09 muss der Unterricht in der **orthodoxen kanonischen Form** gestaltet werden**, es sei denn, es ist ausdrücklich etwas anderes angegeben**.
- Jede Funktionsimplementierung in einer Header-Datei (mit Ausnahme von Funktionsschablonen) bedeutet 0 für die Übung.
- Sie sollten in der Lage sein, jeden Ihrer Header unabhängig von anderen zu verwenden. Daher müssen sie alle Abhängigkeiten einschließen, die sie benötigen. Allerdings müssen Sie das Problem der doppelten Einbindung vermeiden, indem Sie **Include-Guards** hinzufügen. Andernfalls wird Ihre Note 0 sein.

Lies mich

- Sie können bei Bedarf zusätzliche Dateien hinzufügen (z. B. um Ihren Code aufzuteilen). Da diese Aufgaben nicht von einem Programm überprüft werden, können Sie dies gerne tun, solange Sie die vorgeschriebenen Dateien einreichen.
- Manchmal sehen die Richtlinien einer Übung kurz aus, aber die Beispiele können Anforderungen aufzeigen, die nicht ausdrücklich in den Anweisungen stehen.
- Lesen Sie jedes Modul vollständig durch, bevor Sie beginnen! Wirklich, tun Sie es.
- Bei Odin, bei Thor! Benutze deinen Verstand!!!

Sie werden eine Menge Klassen implementieren müssen. Das kann mühsam sein, es sei denn, Sie können in Ihrem Lieblings-Texteditor Skripte schreiben.

Sie haben einen gewissen Spielraum bei der Durchführung der Übungen. Halten Sie sich jedoch an die vorgeschriebenen Regeln und seien Sie nicht faul. Sie würden eine Menge nützlicher Informationen verpassen! Zögern Sie nicht, sich über theoretische Konzepte zu informieren.

Kapitel III

Übung 00: Mama, wenn ich groß bin, will ich Bürokratin werden!

	Übung : 00
Mami, wenn ich groß bin, will ich ein Bürokrat werden!	
Turn-in-Verzeichnis : *ex00/*	
Einzureichende Dateien: `Makefile, main.cpp, Bureaucrat.{h, hpp}, Bureaucrat.cpp`	
Verbotene Funktionen : `Keine`	

`Bitte beachten Sie, dass Ausnahmeklassen nicht in orthodoxer kanonischer Form gestaltet werden müssen. Aber jede andere Klasse muss das.`

Lassen Sie uns einen künstlichen Alptraum aus Büros, Fluren, Formularen und Warteschlangen schaffen.
Klingt lustig? Nein? Schade.

Beginnen wir mit dem kleinsten Rädchen in dieser riesigen bürokratischen Maschine:

dem **Bürokraten**. Ein **Bürokrat** muss haben:

- Ein konstanter Name.
- Und eine Note, die von **1** (höchstmögliche Note) bis **150** (niedrigstmögliche Note) reicht.

Jeder Versuch, einen Bureaucrat mit einem ungültigen Grad zu instanziieren, muss eine Exception auslösen:
entweder eine Bureaucrat::GradeTooHighException oder eine Bureaucrat::GradeTooLowException.

Sie werden Getter für diese beiden Attribute bereitstellen: getName() und getGrade(). Implementieren Sie auch zwei Mitgliedsfunktionen, um den Grad des Bürokraten zu erhöhen oder zu verringern. Wenn der Grad außerhalb des Bereichs liegt, werden beide Funktionen die gleichen Ausnahmen wie der Konstruktor auslösen.

Zur Erinnerung. Da die Besoldungsgruppe 1 die höchste und die Besoldungsgruppe 150 die niedrigste Besoldungsgruppe ist, sollte der Beamte bei einer Erhöhung der Besoldungsgruppe 3 die Besoldungsgruppe 2 erhalten.

Die ausgelösten Ausnahmen müssen mit try- und catch-Blöcken abfangbar sein:

```
Versuchen Sie
{
  /* einige Sachen mit Bürokraten machen */
}
catch (std::exception & e)
{
  /* Behandlung der Ausnahme */
}
```

Sie werden eine Überladung des Einfügeoperators (") implementieren, um etwas wie (ohne die spitzen Klammern) zu drucken:

<Name>, Bürokratengrad <Grad>.

Reichen Sie wie üblich einige Tests ein, um zu prüfen, ob alles wie erwartet funktioniert.

Kapitel IV

Übung 01: Aufstellen, Maden!

	Übung : 01
Aufstellen, Maden!	
Turn-in-Verzeichnis : *ex01/*	
Abzugebende Dateien: `Dateien der vorherigen Übung + Form.{h, hpp}, Form.cpp`	
Verbotene Funktionen : `Keine`	

Jetzt, wo Sie Bürokraten haben, sollten wir ihnen etwas zu tun geben. Was könnte es Besseres geben als das Ausfüllen eines Stapels von Formularen?

Dann lassen Sie uns eine Klasse **Form** erstellen. Sie hat:

- Ein konstanter Name.
- Ein boolescher Wert, der angibt, ob die Datei vorzeichenbehaftet ist (bei der Konstruktion ist sie das nicht).
- Zur Unterzeichnung ist eine konstante Note erforderlich.
- Und eine konstante Note, die für die Ausführung erforderlich ist.

Alle diese Attribute sind **privat**, nicht geschützt.

Die Noten des **Formulars** folgen den gleichen Regeln wie die des Bürokraten. Daher werden die folgenden Ausnahmen ausgelöst, wenn ein Formulargrad außerhalb der Grenzen liegt: Form::GradeTooHighException und Form::GradeTooLowException.

Wie zuvor, schreiben Sie Getter für alle Attribute und eine Überladung des Einfügeoperators ("), der alle Informationen des Formulars ausgibt.

Fügen Sie auch eine beSigned()-Mitgliedsfunktion zum Formular hinzu, die einen Bureaucrat als Parameter annimmt. Sie ändert den Status des Formulars in signiert, wenn der Grad des Bürokraten hoch genug ist (höher oder gleich dem erforderlichen Grad). Denken Sie daran, dass Grad 1 höher ist als Grad 2.
Wenn die Note zu niedrig ist, wird eine Form::GradeTooLowException ausgelöst.

Zum Schluss fügen Sie eine signForm()-Mitgliedsfunktion zum Bureaucrat hinzu. Wenn das Formular unterschrieben wurde, wird es so etwas wie ausgeben:

<Bürokrat> unterzeichnet <form>

Andernfalls wird etwas gedruckt wie:

<Bürokrat> konnte <Formular> nicht unterschreiben, weil <Grund>.

Führen Sie einige Tests durch und geben Sie sie ab, um sicherzustellen, dass alles wie erwartet funktioniert.

Kapitel V

Übung 02: Nein, Sie brauchen Formular 28B, nicht 28C...

	Übung : 02
Nein, Sie brauchen Formular 28B, nicht 28C...	
Turn-in-Verzeichnis : *ex02/*	
Einzureichende Dateien: `Makefile, main.cpp, Bureaucrat.[{h, hpp},cpp], Bureaucrat.cpp + AForm.[{h, hpp},cpp], ShrubberyCreationForm.[{h, hpp},cpp], + RobotomyRequestForm.[{h, hpp},cpp], PresidentialPardonForm.[{h, hpp},cpp]`	
Verbotene Funktionen : `Keine`	

Da Sie nun über die grundlegenden Formulare verfügen, ist es an der Zeit, ein paar weitere zu erstellen, die tatsächlich etwas bewirken.

In allen Fällen muss die Basisklasse Form eine abstrakte Klasse sein und sollte daher in AForm umbenannt werden. Denken Sie daran, dass die Attribute des Formulars privat bleiben müssen und dass sie sich in der Basisklasse befinden.

Fügen Sie die folgenden konkreten Klassen hinzu:

- **ShrubberyCreationForm**: Erforderliche Noten: Zeichen 145, Ausführung 137
  Erstellt eine Datei <target>_shrubbery im Arbeitsverzeichnis und schreibt ASCII-Bäume hinein.
- **RobotomyRequestForm**: Erforderliche Noten: sign 72, exec 45
  Macht einige Bohrgeräusche. Informiert dann, dass <Ziel> in 50 % der Fälle erfolgreich robotomiert wurde. Andernfalls wird mitgeteilt, dass die Robotomie fehlgeschlagen ist.
- **PresidentialPardonForm**: Erforderliche Noten: Zeichen 25,
  Ausführung 5 Informiert, dass <Ziel> von Zaphod Beeblebrox begnadigt wurde.

Sie alle benötigen nur einen Parameter in ihrem Konstruktor: das Ziel des Formulars. Zum Beispiel "home", wenn Sie zu Hause ein Gebüsch pflanzen wollen.

Fügen Sie nun die Funktion execute(Bureaucrat const & executor) const zum Basisformular hinzu und implementieren Sie eine Funktion, um die Aktion des Formulars in den konkreten Klassen auszuführen. Sie müssen überprüfen, ob das Formular unterschrieben ist und ob der Grad des Bürokraten, der versucht, das Formular auszuführen, hoch genug ist. Andernfalls werfen Sie eine entsprechende Exception.

Ob Sie die Anforderungen in jeder konkreten Klasse oder in der Basisklasse prüfen wollen (und dann eine andere Funktion zur Ausführung des Formulars aufrufen), bleibt Ihnen überlassen. Allerdings ist der eine Weg schöner als der andere.

Zum Schluss fügen Sie die Funktion executeForm(AForm const & form) in den Bureau- crat ein. Sie muss versuchen, das Formular auszuführen. Wenn sie erfolgreich ist, drucken Sie etwas wie:

<Bürokrat> ausgeführt <form>

Wenn nicht, wird eine explizite Fehlermeldung ausgegeben.

Führen Sie einige Tests durch und geben Sie sie ab, um sicherzustellen, dass alles wie erwartet funktioniert.

Kapitel VI

Übung 03: Immerhin besser als Kaffee kochen

	Übung : 03
Wenigstens ist das besser als Kaffee kochen	
Turn-in-Verzeichnis : *ex03/*	
Abzugebende Dateien: `Dateien aus vorherigen Übungen + Intern.{h, hpp}, Intern.cpp`	
Verbotene Funktionen : `Keine`	

Da das Ausfüllen von Formularen schon lästig genug ist, wäre es grausam, unsere Bürokraten zu bitten, dies den ganzen Tag lang zu tun. Zum Glück gibt es Praktikanten. In dieser Übung müssen Sie die Klasse **Intern** implementieren. Der Praktikant hat keinen Namen, keine Note, keine besonderen Eigenschaften. Das einzige, was die Bürokraten interessiert, ist, dass er seine Arbeit macht.

Der Praktikant hat jedoch eine wichtige Fähigkeit: `die` Funktion makeForm(). Sie nimmt zwei Zeichenketten entgegen. Der erste ist der Name eines Formulars und der zweite ist das Ziel des Formulars. Sie gibt einen Zeiger auf ein **Formularobjekt** zurück (dessen Name der als Parameter übergebene ist), dessen Ziel mit dem zweiten Parameter initialisiert wird.

Es wird etwas gedruckt wie:

Praktikant erstellt <form>

Wenn der als Parameter übergebene Formularname nicht existiert, wird eine explizite Fehlermeldung ausgegeben.

Sie müssen unleserliche und hässliche Lösungen wie die Verwendung eines if/elseif/else-Waldes vermeiden. Diese Art von Lösungen wird bei der Bewertung nicht akzeptiert. Sie sind nicht mehr in Piscine (Schwimmbad). Wie üblich müssen Sie testen, ob alles wie erwartet funktioniert.

Der folgende Code erstellt beispielsweise ein **RobotomyRequestForm** mit dem Ziel "Ben- der":

```
{
    Intern someRandomIntern;
    Form*   rrf;

    rrf = someRandomIntern.makeForm("robotomy request", "Bender");
}
```

Kapitel VII

Einreichung und Peer-Evaluierung

Reichen Sie Ihre Arbeit wie gewohnt in Ihrem Git-Repository ein. Nur die Arbeit in Ihrem Repository wird während der Verteidigung bewertet. Zögern Sie nicht, die Namen Ihrer Ordner und Dateien zu überprüfen, um sicherzustellen, dass sie korrekt sind.

```
16D85ACC441674FBA2DF65190663F9373230CEAB1E4A0818611C0E39F5B26E4D774F1
74620A16827E1B16612137E59ECD492E468A92DCB17BF16988114B98587594D12810
E67D173222A
```