1 Übungsaufgabe 1: Anwendung von AspectC++

In dieser Übungsaufgabe sollt ihr losgelöst vom BST-Übungsstoff erste Erfahrungen mit der Programmiersprache AspectC++ sammeln.

1.1 Logging-Bibliothek

Zuerst sollt ihr ein einfaces Logging-Framework implementieren, welches mit Hilfe von Aspekten in eine bestehende C++-Anwendung eingebunden werden kann.

1.1.1 Aufrufstatistikmodul

Hier sollt ihr einen Aspekt entwickeln, der die Aufrufe einer bestimmten Member-Funktion für eine Menge von Klassen zählt. Dabei soll der Zähler die Anzahl der Aufrufe pro Objektinstanz erfassen.
Die genaue Signatur der Funktion und die Menge der Klassen sollen vom Anwender des Aspekts spezifizierbar sein. Um das zu erreichen, bietet sich die Verwendung von rein virtuellen *Pointcuts* an. Der Anwender kann dann den abstrakten Aspekt mit entsprechenden *Pointcut*-Ausdrücken instanziieren.

1.1.2 Aufrufstatistik für Funktionen mit bestimmtem Rückgabewert

Schreibt nun ein weiteres Aufrufstatistikmodul, welches global erfasst, wie oft eine Menge von Funktionen mit einer Menge von bestimmten Rückgabetypen aufgerufen wird. (Beispiel: Alle Funktionen, deren Name auf `foo` endet und die entweder einen `int` oder ein Objekt der Klasse `foobar` zurückgeben.)

1.1.3 Logger

Als zentrale Instanz für die Ausgabe der aufgezeichneten Daten soll es eine Klasse `Logger` geben. Diese soll ein Interface bereitstellen, um Objekte, für die Daten aufgezeichnet werden, zu registrieren und wieder abzumelden. Objekte, für die das Logging aktiviert ist, sollen dabei über ihre komplette Lebenszeit am Logger registriert sein.
Darüberhinaus soll es eine Funktion geben, welche die aufgezeichneten Daten für alle zum Zeitpunkt des Aufrufs registrieren Objekte ausgibt.

1.1.4 Ein wiederverwendbare Singleton-Implementierung

Erstellt eine wiederverwendbare Singleton-Implementierung, die auf eine vom Anwender bestimmbare Menge von Klassen angewendet werden kann. Dies kann wieder über rein virtuelle *Pointcuts* realisiert werden.
Für die Klasse `Logger` soll programmweit nur eine einzige Instanz angelegt sein. Dies soll durch die Verwendung eurer Singleton-Implementierung bewerkstelligt werden.

1.2 Nullzeigerprüfungen

Programmiert einen Aspekt, der für eine einstellbare Menge von Funktionen prüft, ob der Rückgabewert `NULL` ist. Ist dies der Fall, so gebt aus, wo dies passiert ist und werft eine Ausnahme.

1.3 Testprogramm

Schreibt ein C++ Testprogramm, welches den kompletten Funktionsumfang eurer Aspekte demonstriert. Für die Implementierung des Testprogramms und auch der Aspekte dürft ihr auf die komplette C++-Laufzeitbibliothek zurückgreifen. Die Einschränkungen, die für die Gastebene gelten, kommen hier nicht zum tragen.

Aufgaben:

→ Implemetierung der verschiedenen beschriebenen Aspekte

→ Erstellen eines aussagekräftigen Testprogrammes, das die Funktionsfähigkeit eurer Aspekte zeigt.

Hinweise:

- Die Implementierung von Aspekten in `*.ah`-Dateien benötigt ebenfalls Include-Guards.
- *Extension-Slices* sollten am besten in einer eigenen `*.ah`-Datei implementiert werden. Zur Verwendung in Aspekten dann einfach mit `#include` in die entsprechende Datei einfügen.
- *Join-Points* für kontrollflussveränderndes *Advice* bezieht sich immer auf den Aufruf oder die Ausführung von Funktionen. Man kann also nur am Beginn, Ende oder beim Aufruf von Funktionen Aspektcode einweben. So müssen sämtliche Stellen, die man mittels Aspekten beeinflussen möchte, entsprechende Eigenschaften haben.
- **Toolchain:** Unter `/proj/i4bst/bin` findet ihr den Aspekt-Compiler. In `/proj/i4bst/tools/aspectc++` könnt ihr euch auch einige AspectC++-Beispiele anschauen. Am einfachsten lässt sich der `ag++` in euer Makefile integrieren. Dazu müsst ihr nur die Übersetzeraufrufe des `g++` durch `ag++` ersetzen. Auch Abhängigkeitsregeln lassen sich mit `ag++` analog zu `g++` mit Hilfe der `-M` und `-MM` Kommandozeilenparameter erzeugen.
- Mit der Option `-p` kann man den Pfad angeben, unter dem der `ag++` die Aspekte sucht, die er in die zu übersetzende Datei einwebt. Gibt man diese Option nicht an, so wird das aktuelle Arbeitsverzeichnis verwendet.
- Dokumentation über AspectC++ findet ihr auf `http://aspectc.org/Documentation.5.0.html`. Nützlich sind vor allem das *AspectC++ Language Quick Reference Sheet*, eine kurz gefasste Übersicht über alle Sprachmerkmale und die *AspectC++ Language Reference*, welche alle Sprachmerkmale detailliert beschreibt.

1.4 Abgabe: am 13.06.2012