C++ FOR (Donnerstagvormittag)

---

1. Pragmatische Leitgedanken der Software-Entwicklung
2. Beispiel für "Open Close"-Prinzip
3. Übung

Kürzere Pausen werden jeweils nach Bedarf eingelegt.

Die Besprechung der Musterlösung(en) erfolgt im direkten Anschluss an die Mittagspause.

Pragmatische Leitgedanken der Software-Entwicklung

- "Open-Close"

- "Don't Repeat Yourself"

"Open Close"-Prinzip

Gemäß diesem Prinzip sollte jede Software-Architektur einen gesunden Ausgleich zwischen zwei gegeneinander stehenden Zielen gewährleisten:

- Software sollte *offen für Veränderungen* sein, beispielsweise
  - Anpassung an künftig geänderten Bedarf
  - Absehbar anstehende Erweiterungen
  - Verwendung in ähnlich gelagerten Fällen
- Software sollte aber auch *robust* sein in dem Sinne, dass
  - Änderungen nicht versehentlich oder in einer ansonsten unbeabsichtigten Weise erfolgen;
  - zumindest sollten unbeabsichtigte Änderungen leicht zu identifizieren sein,
  - ebenso solche, die zielgerichtet im Rahmen der Offenheit erfolgten aber aus irgend einem Grund unvollständig blieben.

Mechanismen zur Strukturierung

Klassen

Klassen fassen "Daten" und "Verarbeitung" zusammen und sind somit ein Mechanismus zur Kapselung, der den Blick auf die abstrakten Operationen lenkt, weg von Datenstrukturen und Algorithmen.

Unterprogramme

Unterprogramme teilen Verarbeitungsschritte auf, vom komplexen Gesamtablauf bis hinunter zu kleinen, einfach überschau- und testbaren Einheiten.

Bibliotheken

Bilbliotheken sind Sammlungen wiederverwendbarer Komponenten, die für sich betrachtet aber kein wesentliches Eigenleben führen sondern erst - quasi "von außen" - zum Leben erweckt werden.

Mechanismen zur Strukturierung (2)

Frameworks

Frameworks folgen den "Hollywood-Principle": *Don't call us, we call you*.

Sie stellen eine oft sehr komplexe Gesamtfunktionlalität zur Verfügung, an denen die spezifischen Anpassungen an vorausgeplanten Erweiterungspunkten eingehängt werden.

Geplante Erweiterbarkeit

Ja, aber auch: "Keep It Small and Simple!"

"Don't Repeat Yourself"

Eine wichtige Erkenntnis kann bei der erfolgreichen Arbeitsteilung zwischen "Mensch und Maschine" sollte folgende sein:

Menschen

- besitzen oft ein hohes Maß an Kreativität,
- sind aber in aler Regel schlecht darin, Dinge präzise zu wiederholen,
  - sei es in immer wieder ein- und derselben Weise,
  - sei es mit systematischen Variationen.

Computer

- besitzen kaum echte Kreativität,*
- sind aber extrem gut darin, Dinge präzise zu wiederholen:
  - Insbesondere ermüden sie nicht, auch bei sich ständig wiederholten und
  - dabei allenfalls leicht variierenden Tätigkeiten.

---

*: Es ist dabei weniger entscheidend, dass per Computer vielleicht gelegentlich "überraschende Ergebnisse" erzielt werden können - etwa in der Art, dass ein Computer ein Musikstück komponieren könnte, dass es vielleicht sogar in die Hitparade schafft. Interessanter ist die Frage nach einem Programm, mit dessen Hilfe ein Computer im Dialog mit einem menschlichen Partner von diesem nicht innerhalb weniger Minuten als Maschine erkannt würde, wie es im Fall der Turing-Tests letzten Endes doch immer wieder schnell der Fall ist.

Mechanismen zur Wiederverwendung

Klassen

Universell gehaltene Klassen bilden in der Regel die kleinsten, wiederverwendbaren Bausteine in einem (konsequent) Objektorientierten Entwurf.

Unterprogramme

In einem eher klassichen (prozeduralen) Entwurf dominieren Unterprogramme.

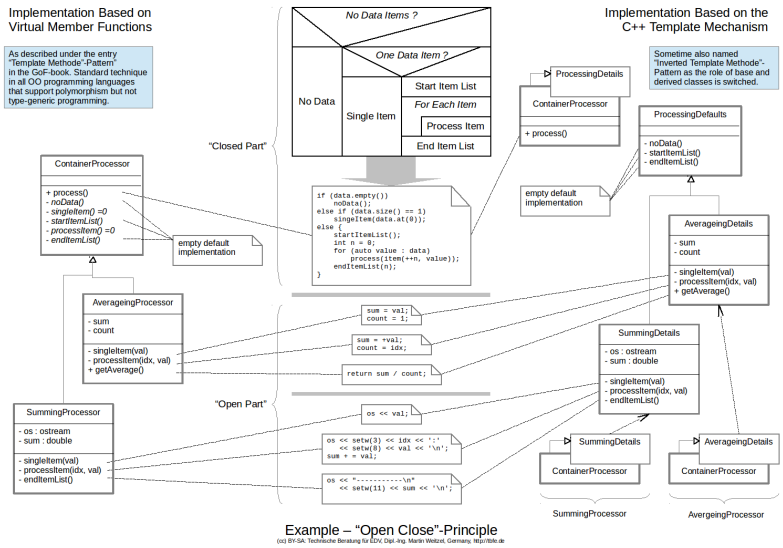
Datenstrukturen

Mit Hilfe der C++ templates lassen sich sehr gut wiederverwendbare Bausteine für Datenstrukturen realisieren.*

*: OK ... technisch gesehen sind die STL-Container Klassen ...

Beispiel für "Open Close"-Prinzip

- Basierend auf virtuellen Member-Funktionen
- Basierend auf C++-Templates



Basierend auf virtuellen Member-Funktionen

Siehe Info-Grafik zusammen mit den Ausführungen des Dozenten.

Basierend auf C++-Templates

Siehe Info-Grafik zusammen mit den Ausführungen des Dozenten.

Übung

Ziel der Aufgabe:

Das GoF Template Methode Pattern soll - als Beispiel für das "Open-Close"-Prinzip - in zwei Techniken umgesetzt werden.

1. "Klassisch" mit virtuellen Member-Funktionen
2. "Modern" mit C++-Templates*

Weitere Details werden vom Dozenten anhand des Aufgabenblatts sowie der vorbereiteten Eclipse-Projekte erläutert.

* ... die wiederum nur eine zufällige namentliche Übereinstimmung mit dem Pattern haben.