1. Öffnen Sie das Werkzeug „objectiF“ mit der Vorlage UML with C++.

2. MVC – Model-View-Controller-Paradigma

Mit dem erstmaligen MVC-Konzept werden Interaktion und Funktion klar voneinander getrennt (seit Ende der 70-ziger im Smalltalk-Kontext bekannt; Fowler: Patterns für Enterprise Application-Architekturen“ mitp Verlag; Krasner, Pope 1988[1]).

Dieses Konzept ist inzwischen weit verbreitet und durchaus auch verändert worden; nicht aber in seinem Grundsatz, der in folgendem Konzept besteht:

Model:

- Die Komponente Model realisiert die fachliche Anforderung (die Funktionalität).
- Die Daten werden gekapselt, d.h. es stehen Zugriffsoperationen zum Hinzufügen, zum Verändern und zum Abfragen der Daten zur Verfügung.

View:

- Die Ansicht („View“) bietet dem Anwender die Sicht auf die Daten, d.h. die Ansicht übernimmt die Präsentation der Daten nach außen.
- Die Ansicht nutzt dabei die zur Verfügung stehenden Zugriffsoperationen des Models.
- Zu einem Model kann es mehrere, auch verschiedene Ansichten geben.

Controller:

- Jeder Ansicht ist ein Controller zugeordnet.
- Der Controller empfängt die Nutzereingaben, wertet diese Eingaben aus und reagiert darauf.
- Die Reaktion kann sein:
  - eine Information an die entsprechende(n) Ansicht(en), die Visualisierung zu ändern (z.B. ausgewählten Text zu markieren).
    oder
  - der Aufruf einer Funktion des Models.

Ablauf:
Durch die Interaktion des Anwenders verändern sich ggf. die Werte der Daten.

- ➔ Das Model informiert die bei ihm angemeldeten Ansichten (Views), d.h. die „Kundenliste“
- ➔ Die informierten Ansichten reagieren mit der Veränderung der Visualisierung (Change-update-Mechanismus)

[1] Cookbook for Using the Model-View-Controller User Interface Paradigme in Smalltalk-80, Journal of Object-Oriented Programmierung, SOIGs Publication, Vol. 1(3), 26-49

3. Aufgaben

a) Skizzieren Sie das MVC-Konzept.

b) Erstellen Sie ein Klassendiagramm, das drei Klassen (Model, View, Controller) und ihre Beziehungen zueinander so anordnet, dass die beschriebene Struktur abgebildet wird.

c) Erstellen Sie ein Sequenzdiagramm, das folgenden Nachrichtenaustausch zeigt (zunächst mit einer Ansicht):

- ➔ Ein Objekt vom Typ „View“ meldet sich bei einem Objekt vom Typ „Model“ an.
- ➔ Ein Anwender hat mit Hilfe der Benutzerschnittstelle ein Ereignis ausgelöst.
- ➔ Ein Objekt vom Typ „Controller“ bearbeitet das Ereignis (Aufruf einer Funktion des Objektes vom Typ „Model“).
- ➔ Dieses Model-Objekt teilt dem View-Objekt mit, dass es seinen Zustand geändert hat.
  (Es „probagiert“ diese Zustandsänderung mit Hilfe einer privaten Methode.)
- ➔ Das View-Objekt wird darüber informiert und holt sich daraufhin die geänderten Daten.
- ➔ Diese geänderten Daten werden zur Ansicht gebracht.

d) Erstellen Sie ein Sequenzdiagramm mit zwei View-Objekten.

e) Diskutieren Sie die Vor- und Nachteile des MVC-Konzeptes.