Aufgabenblatt 4 Vererbung, virtuelle Funktionen und Polymorphie

Aufgabe 1

Erzeugen Sie die Klasse Rechteck. Sie soll charakterisiert sein durch die zwei privaten Gleitkommawerte laenge und breite. Als Schnittstelle nach außen sollen die Elementfunktionen getBreite(), setBreite(), getLaenge() und setLaenge() dienen. Die zwei Elementfunktionen umfang() und rFlaeche() sollen Umfang und Fläche des Rechtecks ausgeben. Leiten Sie von der Basisklasse Rechteck eine Klasse Quader ab. Programmieren Sie die Funktionen qFlache() und volumen(), die die Oberfläche, bzw. das Volumen des Quaders ausgeben. Hinweis: Sie brauchen eine zusätzliche (private) Variable hoehe in der Klasse Quader und die Schnittstellenfunktionen zum setzen und Abfragen dieser Variablen. Schreiben Sie ein Hauptprogramm, um die Funktionalität zu testen. Ist Vererbung hier die richtige Konstruktion? Haben Sie einen besseren Vorschlag?

Aufgabe 2

Führen Sie ein Experiment durch. Untersuchen Sie anhand des unten angegebenen Beispiels die Zuweisung von Objekten, die in einer Ableitungsbeziehung zu einander stehen und die Zuweisung von Zeigern, die auf Objekte zeigen, die in einer Ableitungsbeziehung zueinander stehen. Verwenden Sie dazu die unten angegebenen Klassen in einem Hauptprogramm.

```
struct Base {
    int a;
};

struct Derived: public Base {
    int b;
};
```

Experimentieren Sie mit den folgenden Zuweisungen

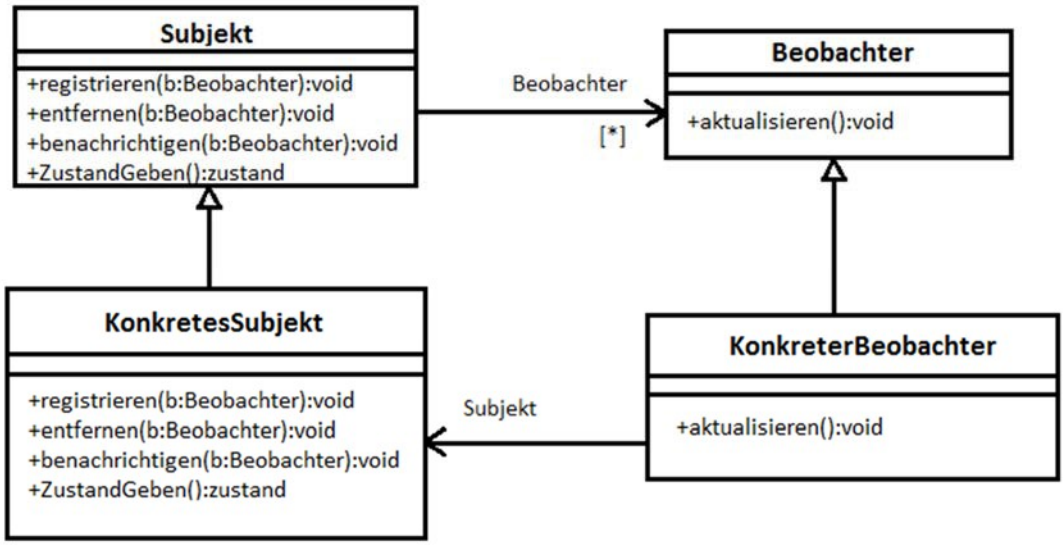
1. Typ Base an ein anderes Objekt vom Typ Derived
2. Typ Derived an ein anderes Objekt vom Typ Base
3. Base * an Derived *
4. Derived * an Base *

Welche Wertzuweisungen sind möglich, welche nicht. Warum? Protokollieren Sie Ihre Versuche und Ihre Beobachtungen (Ausgaben, Fehlermeldungen) und geben Sie eine Erklärung für Ihre Beobachtungen.

Aufgabe 3

In dieser Aufgabe geht es darum, das Beobachter Muster (Observer Pattern) zu implementieren bzw. anzuwenden. Im ersten Schritt informieren Sie sich über das Entwurfsmuster. Startpunkt kann eine Suchmaschine sein, in die Sie die Stichworte „Entwurfsmuster Beobachter“ bzw. „Designpattern Observer“ eingeben

a) Implementieren sie das Muster, wie es in diesem Klassendiagramm beschrieben ist (Quelle: Wikipedia).



Implementieren Sie die Schnittstellen Subjekt und Beobachter als abstrakte Klassen mit rein virtuellen Funktionen. Implementieren Sie KonkretesSubjekt und KonkreterBeobachter mit geeigneten Textausgaben in den Funktionen. Sie dürfen sich an einem Beispiel das Sie recherchiert haben, orientieren. Geben Sie Ihre Quellen an. Achtung: das Beispiel in Wikipedia hat bei meinem Test nicht ohne Bearbeitung funktioniert. Demonstrieren Sie Ihren Code durch eine geeignete main()

b) Implementieren Sie als Anwendungsfall die Beobachtung von Aktienkursen. Eine Aktie ist ein konkretes Subjekt. Beobachter registrieren sich für eine Aktie. Ändert sich der Preis einer Aktie werden alle registrierten Beobachter informiert. Versuchen Sie Ihren Code aus a) zu verwenden. Wie sieht die Wiederverwendung aus? Demonstrieren Sie Ihren Code durch eine geeignete main()

c) Erweitern Sie b) um eine zweite (zusätzliche) Art der Benachrichtigung. Hier wird der Beobachter nur informiert, wenn die Aktie unter einen definierten Schwellwert sinkt. Können Sie diese Funktionalität implementieren, ohne die Schnittstellen zu ändern? Wenn nein: welche Änderung haben Sie an den Schnittstellen durchgeführt und warum? Demonstrieren Sie Ihren Code durch eine geeignete main()

Die Programme sind in Felix abzugeben. Dokumentierten Sie die C++-Texte, z.B mit Kommentaren. Der Abgabetermin ist im Felix Kurs hinterlegt.