ulm university universität
uulm

Labor Softwareentwurf mit Multiparadigmen-Programmiersprachen
H. Falk, A. Luppold

Versuch 4
vom 2. Dezember 2013

Bearbeitung bis: 15. Dezember 2013

Objektorientierte Programmierung II

Versuch 4-1: Theorie (8 Punkte)

a) Wie unterscheiden sich Vererbung und Komposition?

b) Was versteht man im Bereich der Überladung virtueller Funktionen unter Polymorphie? Wie äußert sich diese zur Laufzeit eines Programms?

c) Wie muss eine überladene Member-methode aufgerufen werden, damit sich der dynamische und der statische Typ des zugrundeliegenden Objekts unterscheiden können? Begründen Sie Ihre Antwort.

d) Wie lässt sich bei einer abgeleiteten Klasse mit überladenen Funktionen gezielt die überladene Memberfunktion der Basisklasse aufrufen?

e) Erläutern Sie die Funktion der Schlüsselwörter `final` und `override`.

Versuch 4-2: Vererbung (35 Punkte)

a) Schreiben Sie eine Basisklasse `unimitglied`, die als Container für beliebige Mitglieder der Universität Ulm dienen soll. Ihre Basisklasse soll dabei mindestens Name, Vorname und Privatanschrift der Personen erfassen können. Über `get()`- und `set()`-Methoden soll es möglich sein, die einzelnen Werte zu verändern. Sollten die Membervariablen, in denen die einzelnen Werte der Klasse erfasst werden, als `public`, `private`, oder `protected` implementiert werden? Begründen Sie Ihre Antwort.

b) Implementieren Sie einen Konstruktor, der es ermöglich alle Werte bei der Objekterzeugung zu setzen.

c) Erstellen Sie ein Testprogramm, das mehrere Universitätsmitglieder verwalten kann. Wandeln Sie hierfür zunächst Ihre selbsterstellte Liste aus Versuch 3 so ab, dass als Daten `std::shared_ptr` auf Objekte vom Typ `unimitglied` anstelle von `int` hinterlegt werden können.
Erstellen Sie nun eine weitere Klasse `verwaltung`. Die Klasse soll es ermöglichen, neue Universitätsmitglieder anzulegen, und bestehende Mitglieder zu löschen. Für

jedes Mitglied soll ein neues Objekt vom Typ unimitglied angelegt und über einen std::shared_ptr in Ihrer Listenstruktur abgespeichert werden.

d) Sollte verwaltung von Ihrer Listen-Klasse erben, oder ist Komposition sinnvoller? Begründen Sie Ihre Entscheidung. Falls Sie Vererbung als sinnvoll erachten, erläutern Sie ob öffentliche oder private Vererbung sinnvoller ist.

e) Schreiben Sie ein Testprogramm das ein Objekt vom Typ verwaltung erzeugt und es ermöglicht, Universitätsmitglieder darin anzulegen, zu löschen, sowie die Daten zu bestehenden Mitgliedern zu löschen.

f) Erweitern Sie ihr Programm, um Personen anhand ihres Nachnamens zu finden und deren Daten ausgeben zu können.

Versuch 4-3: Mehrfachvererbung (20 Punkte)

a) Implementieren Sie die Klassen angestellte und studierende, die beide jeweils von unimitglied erben. angestellte soll als zusätzliches Merkmal mindestens eine Methode zum Setzen und Abfragen der Personalnummer besitzen. Bei studierende sollen Matrikelnummer und Studiengang erfasst werden.

b) Implementieren Sie eine Klasse hiwi, die sowohl von studierende als auch von angestellte erbt. Welche möglichen Probleme ergeben sich? Wie lassen sich diese Umgehen?

c) Erweitern Sie ihr Testprogramm (bzw. die Klasse verwaltung) aus dem vorherigen Versuch so, dass bei Studierenden zusätzlich zu den Stammdaten des Objekts unimitglied auch der Studiengang mithilfe der dazugehörigen Memberfunktion ausgegeben wird. Wie lässt sich verhindern, dass ihr Testprogramm versucht bei Objekten vom Typ angestellte (die dort nicht existierende) Memberfunktion für die Ausgabe des Studiengangs aufzurufen? (Hinweis: Sie benötigen hierzu keine zusätzlichen Membervariablen, Flags o. Ä.)

Versuch 4-4: Ausgabe (15 Punkte)

a) Überladen Sie den << Operator so, dass mit dem Befehl std::cout << verwaltung << std::endl alle Universitätsangehörige ausgegeben werden. Dabei soll bei Angestellten ebenfalls die Personalnummer, und bei Studierenden die Matrikelnummer und der Studiengang ausgegeben werden. Bei Hilfskräften sollen alle 3 Zusatzinformationen ausgegeben werden. Erstellen Sie dazu zunächst für jede Ihrer Klassen eine angepasste Funktion print_to_ostream(std::ostream &os), und rufen Sie diese dann aus dem überladenen << Operator heraus auf.

b) Wie können Sie sicherstellen, dass jede Klasse, die von unimitglied erbt, eine eigene Funktion print_to_ostream() implementieren muss? Setzen Sie dies in Ihrem Programm um.

   Können Sie auch erzwingen, dass für jede Klasse der << Operator überladen werden muss? Begründen Sie Ihre Antwort und setzen Sie dies gegebenenfalls in Ihrem Programm um.

Versuch 4-5: Casts (12 Punkte)

a) Erklären Sie die Unterschiede zwischen const_cast, dynamic_cast, static_cast, reinterpret_cast, sowie den aus C-bekannten Casts.

b) Implementieren sie eine Memberfunktion innerhalb der Klasse verwaltung, die eine std::list mit allen Studierenden ausgibt.

c) Nennen Sie den exakten Typ, den der Rückgabewert besitzen sollte. Erläutern Sie Ihre Antwort.