Fakultät Informatik
Professur für Software Engineering
Prof. Dr. M. Rezagholi

HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Softwarequalität, Praktikum (SS 2013)

Aufgabe Nr. 4 (Pflichtaufgabe)

Abgabetermin: 27.05.13, 11.15 Uhr

Funktionaler Test eines C++-Programms

Betrachten Sie zunächst die folgenden 3 Teilaufgaben (sie wurden vom Herrn Prof. Dr. Kaspar im Rahmen des Moduls „Programmierung 3" gestellt). Betrachten Sie anschließend eine mögliche Implementierung dieser Aufgaben in C++ (das Programm wird Ihnen zur Verfügung gestellt).

Erfüllt das Programm die Anforderungen? Spezifizieren Sie die nötigen Testfälle und Testdaten.

Aufgabe 1:

Programmieren Sie eine Klasse `Person` mit der privaten Elementvariablen `name` (der Einfachheit halber vom Typ char) und der privaten Elementvariablen `id`. Fügen sie die notwendigen `set`() und `get`() Elementfunktionen zum Zugriff auf und zur Änderung der privaten Variablen hinzu.

Leiten Sie von dieser Klasse die Klassen `Angestellter`, `Arbeiter`, `Verkaeufer` ab. `Angestellter` soll die private Variable `gehalt` und die public-Elementfunktion `bezahlung`() enthalten, die einfach `gehalt` als Rückgabewert zurückgibt, `Arbeiter` die privaten Variablen `stundenlohn` und `anzahlstunden` und die public-Elementfunktion `bezahlung`() enthalten, die `stundenlohn*anzahlstunden` zurückgibt, `Verkaeufer` die privaten Variablen `gehalt` und `provision` enthalten und die (public) Elementfunktion `bezahlung`() enthalten, die `gehalt+provision` zurückgibt. Fügen Sie notwendige `set`() und `get`() Funktionen hinzu.

Programmieren Sie einen Konstruktor der Basisklasse Person:

```
Person(char name, int id)
```

und einen Konstruktor für jede der abgeleiteten Klassen, der die Parameter an den Konstruktor der Basisklasse übergeben kann.

Aufgabe 2:

In Aufgabe 1 haben Sie eine Klasse `Person` als Basisklasse und die abgeleiteten Klassen `Angestellter`, `Arbeiter`, `Verkaeufer` realisiert. Überarbeiten Sie Aufgabe 1, indem Sie in der Klasse `Person` die Funktion `bezahlung`() zu einer virtuellen Funktion machen.

Verändern Sie die Klassenhierarchie so, dass `Verkaeufer` von `Angestellter` und nicht direkt von `Person` abgeleitet wird. (Die Klasse `Angestellter` enthält die Variable `gehalt`, die Klasse `Verkaeufer` die Variable `provision`).

Achtung: Um die virtuelle Bindung, d.h. die Entscheidung zur Laufzeit, welche Funktion `bezahlung`() verwendet wird zu demonstrieren müssen sie die Funktion `bezahlung`() über einen Pointer (eine Variable vom Typ Person *) aufrufen.

Fakultät Informatik
Professur für Software Engineering
Prof. Dr. M. Rezagholi

HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Aufgabe 3:

Programmieren Sie die Klasse `Behälter` als abstrakte Klasse mit den reinen virtuellen Elementfunktionen:

```
bool insert(Person *);
bool remove(void);
bool isFull(void);
bool isEmpty(void);
const Person * getFirst(void);
bool hasNext(void)
const Person * getNext(void);
```

Leiten Sie von dieser Klasse eine konkrete Realisierung einer Behälterklasse für Objekte vom Typ `Person` ab. Sie können die Behälterklasse auf verschiedene Art und Weise realisieren, solange ein sequenzielles Durchgehen der einzelnen Behälterelemente mit `getFirst`(), `hasNext`() und `getNext`() möglich ist. Der Pointer ist ein `const Person` *, damit Sie über diesen Pointer ein Personobjekt nicht verändern können. Infrage kommt z.B. eine Realisierung des Behälters mit einem gekapselten Feld von `Person` *-Elementen. Das Verhalten bei Einfügen oder Herausnehmen eines Elements kann z.B. dem einer Queue oder eines Stack entsprechen. Über den Konstruktor initialisieren Sie ihre Behälterklasse, so dass ihr Behälterobjekt eine feststehende Anzahl von Person Objekten aufnehmen kann, jedoch noch keine Person enthält. Fügen Sie einen parameterlosen Konstruktor hinzu der standardmäßig einen Behälter mit der Kapazität 10 Personobjekte erzeugt und einen parametrisierten Konstruktor bei dem Sie die Kapazität des Behälters angeben.

Die Funktion `insert`() soll vor dem Einfügen prüfen, ob der Behälter voll ist und „true“ zurückgeben, falls das Einfügen erfolgreich war und sonst „false“. Die Funktion `remove`() entfernt eine Person aus dem Behälter und gibt sie als Rückgabewert zurück. `isFull`() gibt „true“ zurück, wenn der Behälter voll ist, sonst „false“. `isEmpty`() gibt „true“ zurück, wenn der Behälter leer ist, sonst „false“.

Verwenden Sie für `Person` die Klassenhierarchie aus Aufgabe 2. Demonstrieren Sie die Funktionsweise ihres Behälters mit einem geeigneten Hauptprogramm. Rechnen Sie dazu die Gehaltssumme aller Personen im Behälter zusammen.

Das Vorgehen wird gewählt, so dass Behälterklassen mit verschiedenen Implementierungen mit den gleichen Schnittstellen zur Verfügung stehen, so dass eine Anwendung, die eine Ausprägung eines Behälters nutzt, diesen Behälter durch einen anderen ersetzen kann ohne den Code ändern zu müssen. Testen Sie diese Aussage, indem Sie sich einen Partner suchen, mit dem Sie die realisierte Behälterklasse tauschen. Testen Sie seine Behälterklasse (ohne etwas an ihrem Test zu ändern) und lassen Sie Ihre Behälterklasse testen.

Durch die Ableitungshierarchie von `Person` können beliebige Arten von Personen im Behälter gespeichert werden.