Prof. Dr. Bernhard Zimmermann
FB Automatisierung und Informatik
Professor für Sprachen und Compilerbau

▲ Hochschule Harz
Hochschule für angewandte Wissenschaften

LV „Objektorientierte Softwaretechnik / Programmierparadigmen“

Praktikumsaufgabe 2

Deklaration und Implementierung der Klassen `Sekretaer` und `Chefin` zur Verwaltung der Daten von speziellen Mitarbeitern

1. Ziel

Ziel ist es, zwei Klassen zu implementieren, die aus einer gemeinsamen Basisklasse abgeleitet sind und geerbte Methoden neu implementieren.

2. Aufgabe

Es sind zwei Klassen für die Verwaltung der Daten von speziellen Mitarbeitern, nämlich Chefinnen und Sekretäre, zu realisieren und zu testen. Beide Klassen sind Unterklassen der in der Aufgabe 1 realisierten Klasse `Mitarbeiter`.

Klassendeklaration, Klassenimplementierung und das Testprogramm sollen jeweils in separaten Dateien abgelegt werden.

Legen Sie für die Parameter und die Rückgabewerte der Methoden neben dem Datentyp auch die Übergabeart (Kopie oder Referenz) und die Änderungsklasse (konstant, modifizierbar) fest. Darüber hinaus sollten Sie insbesondere für die Konstruktoren Default-Argumente definieren.

Für die Lösung dieser Aufgabe sind nur die Inhalte des Vorlesungsskripts bis einschließlich Seite 125 relevant, d.h. nur die dürfen verwendet werden.

3. Grobspezifikation der Klasse `Sekretaer`

Die Klasse `Sekretaer` speichert die Daten eines Sekretärs und stellt zusätzlich noch Methoden zur Ausgabe und zur Änderung dieser Daten zur Verfügung.

3.1. Attribute

Jeder Sekretär ist ein Mitarbeiter. Zusätzlich zu den von der Klasse `Mitarbeiter` geerbten Attributen hat ein Sekretär noch Fremdsprachenkenntnisse (Englisch, ...). Die Hauptfremdsprache ist in einem statischen Feld gespeichert. Legen Sie die Maximallänge für die Fremdsprache als statische Konstante innerhalb der Klasse an: `static const size_t FREMDSPRACHE_MAX_LEN = 17;`. Verwenden Sie für die Adressierung von Feldelementen per Index als Variablendatentypen für diese Indizes ausschließlich den vorzeichenlosen Datentyp `size_t`.

3.2. Methoden

Die Klasse `Sekretaer` übernimmt alle von der Klasse `Mitarbeiter` geerbten Methoden in ihre öffentliche Schnittstelle, implementiert die geerbte Ausgabe-Methode neu und stellt darüber hinaus noch eine weitere Methode zur Verfügung.

- `Konstruktor(persNr, gehalt, fremdsprache)`

  Der Sekretär wird mit `persNr`, `gehalt` und `fremdsprache` initialisiert. Falls `fremdsprache` mehr als die maximale Anzahl von Zeichen enthält, wird der zugehörige Attributwert entsprechend gekürzt.

- **`void`** `ausgeben(strm)`

  Ausgabe der Komponenten des Sekretärs über den Ausgabedatenstrom `strm`

- **`void`** `fremdspracheAendern(fremdsprache)`

  Ändert die Fremdsprache des Sekretärs in `fremdsprache`. Falls `fremdsprache` mehr als die maximale Anzahl von Zeichen enthält, wird die Fremdsprache entsprechend gekürzt.

- **`const char*`** `getFremdsprache()`

  Liefert einen Zeiger auf die schreibgeschützte Fremdsprache.

4. Grobspezifikation der Klasse `Chefin`

Die Klasse `Chefin` kapselt die Daten einer Chefin und stellt zusätzlich noch Methoden zur Ausgabe und zur Änderung dieser Daten zur Verfügung.

4.1. Attribute

Jede Chefin ist ein Mitarbeiter. Zusätzlich ist eine Chefin noch verantwortlich für einen Bereich (Vertrieb, ...), der beliebig lang sein kann, also in einem dynamischen Feld abgelegt ist. Jede Chefin hat keinen oder genau einen Sekretär. Da der Sekretär nach der Entlassung der Chefin i. a. in der Firma weiterarbeitet, darf das `Sekretaer`-Objekt nicht als festes Subobjekt in dem `Chefin`-Objekt enthalten sein. Darüber hinaus darf eine Chefin den Zustand ihres Sekretärs nicht verändern (z.B. Gehalt erhöhen/kürzen)

4.2. Methoden

Die Klasse `Chefin` übernimmt alle von der Klasse `Mitarbeiter` geerbten Methoden in ihre öffentliche Schnittstelle, implementiert die geerbte Ausgabe-Methode neu und stellt darüber hinaus noch zwei weitere Methoden zur Verfügung.

- `Konstruktoren(sekretaer, persNr, gehalt, bereich);`

  Die Chefin wird mit `sekretaer`, `persNr`, `gehalt` und `bereich` initialisiert. Dabei soll auch eine Chefin ohne Sekretär definierbar sein. `bereich` kann beliebig lang sein.

- **`void`** `ausgeben(strm)`

  Ausgabe der Komponenten der Chefin und ihres Sekretärs

- **`void`** `bereichAendern(bereich)`

  Ändert den Bereich der Chefin in `bereich`. `bereich` kann beliebig lang sein.

- **`void`** `sekretaerAendern(sekretaer)`

  Ordnet der Chefin als neuen Sekretär `sekretaer` zu.

Ein Aufruf dieser Methode ohne Parameter bewirkt, dass die Chefin anschließend keinen Sekretär hat.

- `const char* getBereich()`

  Liefert einen Zeiger auf den schreibgeschützten Bereich der Chefin.

4.3. Implementierungsanforderungen

Die Objekte der Klasse `Chefin` enthalten eine dynamische Komponente. ***Deshalb müssen der Destruktor sowie der Zuweisungsoperator und der Copy-Konstruktor in dieser Klasse überladen bzw. neu implementiert werden.*** Nach der Ausführung der letzteren beiden ist den Zielobjekten kein `Sekretaer` zugeordnet, vom Bereich der Quell-Chefin wird für die Ziel-Chefin dagegen eine Kopie angelegt.

Integrieren Sie zu Testzwecken in den Destruktor Ausgabeanweisungen, die die wichtigsten Daten des Objekts ausgeben (siehe Testprotokoll).

5. Implementierungshinweis

Keiner.

6. Testprogramm

Die Projektvorlage `Projekt-Firma` (siehe Aufgabe1) enthält u. a. ein Testprogramm `TSC.cpp`, das statische Objekte definiert, mit Werten initialisiert und deren Attribute modifiziert. Testen Sie Ihre Klassen `Sekretaer` und `Chefin` mit dem Testprogramm `TSC.cpp` .

Darüber hinaus enthält

STUD.IP – Objektorientierte Softwaretechnik / Programmierparadigmen /Praktikum/1. Praktikum

eine Datei `02_TCS_Protokoll.pdf`, die das Ablaufprotokoll des Testprogramms `TSC.cpp` zeigt.

7. Abgabe

Abzugeben sind Ausdrucke der dokumentierten Programmdateien.

Spätester Abgabetermin:

Die Lösungsunterlagen beider Klassen sollten der zuständigen LaborbetreuerIn nach der Abnahme am Ende des **1**-ten zu dieser Aufgabe stattfindenden Labortermins in Papierform übergeben werden. Ist dies nicht möglich, so müssen die Abnahme und die Übergabe der Lösungsunterlagen spätestens beim nächsten Labortermin erfolgen.

Prof. Dr. Bernhard Zimmermann