Christian-Albrechts-Universität zu Kiel
Sommersemester 2000
Fortgeschrittenenpraktikum (Java)

Übung 2:

Ausgabetermin: 8. Mai 2000

Abgabe: 15. Mai 2000

Aufgabe 1: [Exceptions]

Erweitern Sie das in der Vorlesung vorgestellte Beispiels des Bank und des Sparkontos, indem Sie unerwünschtes Verhalten durch *Ausnahmebehandlung* abfangen. Unerwünschtes Verhalten ist, mehr abzuheben als auf dem Konto ist. Ein weiteres unerwünschtes Verhalten ist, ein neues Konto/Sparkonto mit negativem Betrag "zu eröffnen".

Aufgabe 2: [Weiter mit den Pizzas]

Wie angekündigt geht es weiter mit der Serie der Aufgaben zu den Pizzas. Die Lösung auf Aufgabe 5 des vorangegangenen Zettels war aus verschiedenen Gründen nicht sehr befriedigend. Ein Grund zur Kritik war die fehlende Allgemeinheit. Wir hatten 6 Varianten von Pizza-Belag — Zwiebeln, Salami, Knoblauch, Oliven, Thunfisch und Lammfleisch — und unten, als "Induktionsanfang", Pizzaboden.

Wir trennen nun den "Belag" von dem Aufbau der Pizzas; die alten Definitionen werden weggeschmissen. Sei also ein Belag abstrakt als

```
abstract class Belag_D
```

gegeben, dazu als konkrete, also instantiierbare Ausformungen die oben erwähnten 6 Varianten. `new Paprika()` ergibt also eine Paprika. Mit dieser Änderung sollen Pizzas nun beispielsweise so

```
new Pizza_D (new Paprika(), (new Pizza_D (new Zwiebel(), new Bot()))
```

konstruierbar sein. Diese Aufgabe soll eine Anpassung von Aufgabe 4 bzw. 5 an die neue Struktur sein. Gefordert sind zwei Methoden

- `rem_Knoblauch()`: entferne allen Knoblauch von einer Pizza
- `subZdL()`: ersetze alle Zwiebeln durch Lamm

Dies soll wieder mittels zweier *Visitoren* geschehen, `Sub_ZdL_V` und `Rem_Knob_V`. Damit es nicht zu schwer wird, hier ein Gerüst:

```
class Rem_Knob_V {
  Pie_D forBot() {
    ....
  };
  Pie_D forTop(Object t, P_D p){
    < Rumpf >
  };
};
```

Da wir nur Knoblauch entfernen wollen, sonst nichts, ist hier insbesondere die entsprechende Fallunterscheidung im Rumpf zu überlegen: wie bestimmt man, ob das Objekt im Parameter Knoblauch ist?[1]

Aufgabe 3: [Pizza (2)]

Ungünstig an der vorangegangen Aufgabe ist, daß die beiden Methoden je nur für eine Belagvariante funktionieren — Knoblauch im einen, Zwiebeln/Lamm im anderen. Das ist das Gegenteil von "leicht erweiterbar". Die Aufgabe besteht also darin, ein *allgemeines Entfernen* und *allgemeines Substituieren* zu programmieren. Wie in dem Codefragment aus Aufgabe 2 bereits angedeutet, können Pizzas nicht nur mit Instanzen von `Belag_D` belegt werden, sondern mit beliebigen Objekten, also wollen wir gleich beliebige Objekte entfernen und ersetzen können. In den Pizzas selbst sollen also die allgemeineren Methoden `rem(Object o)` und `subst(Object n, Object o)`[2] bereitstellen, und die beiden Visitoren seien

- `class Rem_V`
- `class Sub_V`

Überlegen Sie, welche Parameter die Methoden `forBot` und `forTop` der beiden Visitoren nun haben. Eine kleine weitere Hürde könnte wieder die Fallunterscheidung sein, wie in der vorangegangenen Aufgabe im Spezialfall von Knoblauch bzw. Zwiebeln.

Allgemeine Bemerkung: Damit nicht jemand dadurch, daß er zu Beginn der Aufgabenreihe andere mögliche Lösungen, als sie mir vorschwebten, gewählt hat, die sich nicht in der vorgesehen Weise verallgemeinern lassen oder nur auf andere Weise, abgehängt wird, werde ich am Freitag meinen Lösungsvorschlag des 2ten Übungszettels zugänglich machen.

Im Übrigen: was die Allgemeinheit der Lösung betrifft, sind wir noch nicht am Ende. Im Vergleich zu Übung 2 haben wir gewissenmaßen die Pizzas und die Visitoren bezüglich ihres möglichen "Belages" *parametrisiert.* Wir haben jetzt eine listenförmige Datenstruktur, bei der wir einzelne Knoten ersetzen oder entfernen können. Damit können nachträglich neue

[1] Wie geschickt man das macht beeinflußt, wie einfach die Verallgemeinerung der nächsten Aufgabe sein wird.

[2] `n` für neu, `o` für alt.

Datentypen dem Programm hinzugefügt werden (sogut wie alles ist schließlich ein Objekt), ohne daß der Datentyp `Pizza_D` angepaßt werden muß. Wir nutzen dabei die Polymorphie der Sprache aus. Der nächste Schritt wird sein, die Pizzas/Listen so zu verallgemeinern, daß *neue Visitoren*, also Klassen, deren Methoden auf den Pizzadaten operieren wie eben das Entfernen und das Ersetzen, *hinzugefügt* werden können, ohne daß `Pizza_D` verändert werden muß! Wer das alleine hinbekommt, ist gut.