Übungen zu Softwareentwicklung 1 **WS 2000/2001**
Übung 7

Name: _____ Tutor: _____

Matrikelnummer: _____ Punkte: _____

Gruppe: _____ **Abzugeben bis: 13.12.00, 12:00**

Übungsleiter: _____ Bearbeitungsdauer: _____

Aufgabe 1 (6 Punkte): Tirolerisch

Schreiben Sie ein Java-Programm, das einen Text einliest und wieder ausgibt, wobei folgende Zeichenersetzungen vorgenommen werden:

Zeichenfolge	ersetzen durch
`"st"` am Wortende(danach folgt ein Leer- oder Satzzeichen)	`"sch"`
`"st"` in der Wortmitte oder am Wortanfang	`"scht"`
`"St"` am Wortanfang	`"Scht"`

Beispiel: "*Was ist los?*" wird zu "*Was isch los?*", "*lustig*" wird zu "*luschtig*"

Hinweise:

- Achten Sie auf die richtige Verwendung von Objekten der Klassen `String` bzw. `StringBuffer` (siehe Java Docu bzw. Übung 6 Aufgabe 2)
- Es sind hier keine Klassen notwendig (Programm nur in `main` , so wie vor Übung 6)

Aufgabe 2 (6 Punkte): Weihnachtslebkuchen

Helfen Sie dem Weihnachtsmann Lebkuchen für die lieben Kinder zu backen, indem Sie eine Lebkuchenklasse mit den Attributen Form (`int`), Glasur (`boolean`) und Verzierung (`String`) definieren. Legen Sie fest, wie ein Standardlebkuchen aussehen soll ("*no arg*"-*Konstruktor*). Legen Sie außerdem einige mögliche Formen (z.B. Herz, Baum, Stern, ...) fest.
Im Testteil (`main`) soll der Benutzer bei einer Abfrage eine der gegebenen Formen auswählen (Auswahl → `int`). Über eine zweite Abfrage soll bestimmt werden, ob der Lebkuchen eine Zuckerglasur bekommt oder nicht (j/n → `boolean`). In einer dritten Abfrage kann die Verzierung des Lebkuchen (z.B. Nüsse, Streusel, ...) angegeben werden (Texteingabe → `String`).
Dann erzeugen Sie einen solchen Lebkuchen (*mit einem entsprechenden Konstruktor!*)
In einer letzten Abfrage geben Sie noch an, wieviele Lebkuchen von der spezifizierten Sorte gemacht werden sollen. Danach erzeugen Sie die geforderte Anzahl von Kopien des Originals (*wieder mit einem entsprechenden Konstruktor!*) und geben Sie die Merkmale jeder Kopie aus (*durch Zugriff auf die Kopie selbst!*).
Der Bildschirmdialog könnte folgendermaßen aussehen:

```
Choose a shape:
[0] tree
[1] star
[2] heart
1
With icing (y/n)? y
What topping? nut
How many? 3
  1 star-shaped gingerbread with icing and nut on top ready
  2 star-shaped gingerbread with icing and nut on top ready
  3 star-shaped gingerbread with icing and nut on top ready
Produce another sort? n
```

Aufgabe 3 (12 Punkte): Hausverwaltung

Sie - als HausverwalterIn - wollen genau wissen, wer wo in den von Ihnen verwalteten Häusern wohnt. Dazu brauchen Sie ein kleines Programm, dass Ihnen die gewünschten Informationen schnell zugänglich macht, und Ihnen auch erlaubt, Änderungen einfach durchzuführen.
Implementieren Sie ein Hausverwaltungsprogramm, das folgende Funktionalität bieten:

a) Es sollen beliebig viele Häuser verwaltet werden können. Zur Laufzeit sollen Häuser dazugenommen oder wieder abgegeben werden können (d.h. man braucht Methoden zum Hinzufügen, Entfernen, ev. Suchen von Häusern (s.VO)).
b) Von jedem Haus möchten Sie wissen, wo es sich befindet (eindeutige Adresse). Jedes Haus hat eine fixe Anzahl von Stockwerken und in jedem Stockwerk eine fixe Anzahl von Wohnungen.
c) Man soll auf die Bewohner (Personen) jeder Wohnung zugreifen können. Es können jederzeit Personen in eine Wohnung ein- oder aus einer Wohnung ausziehen (selbe Situation wie bei den Häusern (Hinzufügen/Enfernen/(Suchen)).
d) Von den Personen will man den Namen und das Alter wissen (diese beiden Angaben sollen eine Person innerhalb einer Wohnung eindeutig identifizieren).
e) Schreiben Sie ein Testprogramm, das mind. 2 Häuser, die Sie verwalten, erzeugt. Diese sollen unterschiedlich viele (mind. 2) Stockwerke besitzen. Bevölkern Sie die erzeugten Wohnungen mit unterschiedlich vielen Personen (es dürfen auch manche (max.½) Wohnungen leer bleiben).
f) Sehen Sie Methoden vor, die es Ihnen erlauben, die Personendaten der Bewohner auf dem Bildschirm auszugeben. Im Testprogramm soll ein Aufruf der Form
   `myHouseAdmin.writeResidents();`
   genügen. Dadurch soll das Hausverwaltungs-Objekt, alle seine Haus-Objekte anweisen, ihre Bewohner auszugeben. Die einzelnen Haus-Objekte leiten das ihrerseits an ihre Wohnungs-Objekte weiter. Diese können dann endlich wirklich auf die Personen zugreifen.

Die Ausgabe sollte nach Häusern, Stockwerken, Wohnungen gegliedert sein, und könnte zum Beispiel so aussehen:

```
House: Freistaedterstr.315
  Floor 0
    Apt# 0:
    Apt# 1: Joan(40)
  Floor 1
    Apt# 0: Jack(56)
    Apt# 1: Andi(29), Rudi(27)
  Floor 2
    Apt# 0:
    Apt# 1: Johnny(8), Frank(10), Sam(12), Toni(36), Lisa(39)

House: Altenbergerstr.69
  Floor 0
    Apt# 0: Jimmy(14), Sarah(43), Kevin(45)
  Floor 1
    Apt# 0: Hank(33)
    Apt# 1: Tom(23), Mike(23), Steve(24)
    Apt# 2: Paul(77), Susan(76)
```

Hinweise:

- Als Klassen bieten sich Hausverwaltung (a), Haus (b), Wohnung (c), Person (d) an.
- Um untergeordnete Objekte zu verwalten, können Sie Arrays (ev. mehr-dimensional) oder lineare Listen verwenden, je nachdem welche Form der Datenstrukturierung Ihnen am geeignetsten erscheint. Begründen Sie Ihre Entscheidung in der Lösungsidee.
- Beachten Sie auch die Datenkapselung (get-/set-Methoden)!
- Bei jeder Klasse, die die Ein-/Ausgabe-Funktionen braucht, müssen Sie nach "`class` *`ClassName`*" "`extends Basic`" ergänzen.