Lehrstuhl für Angewandte Informatik I
Prof. Dr. Bernhard Westfechtel
Sandra Greiner

WS 2017/18
07. Dezember 2017

Konzepte der Programmierung

Übung 8 – Abgabe: 21.12.2017

In diesem Übungsblatt werden die Prinzipien der Rekursion vertieft. Hilfestellung dazu finden Sie im e-Learning und auf dem letzten Übungsblatt. Außerdem soll es mit Arrays und Objekten *weihnachtlich* werden.

Aufgabe 8.1 (Lebkuchenhaus, 2D-Arrays)
Implementieren Sie eine Methode, die ein **Lebkuchenhaus** mittels eines 2D-Arrays (Datentyp: `char`) mit gleich langen Zeilen zeichnet. Die Höhe des *Wohnbereichs* (und des Dachs) wird übergeben und ist immer durch zwei teilbar, denn die Breite des Wohnraums ist immer das Doppelte der Höhe.

Das Haus setzt sich zusammen aus zwei Rechtecken, die zusammen ein Quadrat bilden: Das erste Rechteck ist das Dach und genauso hoch wie der Wohnbereich (diese Höhe wird übergeben). Um den Zuckerguss zu simulieren, wird die linke Seite des Daches mit '2'en gezeichnet, die rechte Seite mit '6'en.
Der Wohnbereich, ist das zweite Quadrat. Der Boden besteht aus '0'en, die linke und die rechte Wand aus '8'en. Innerhalb des Wohnbereichs befinden sich Schneeflocken (`'*'`)
Das fertige Haus schaut für die Höhe 6 (Dach und Wohnbereich sind je 6 Einheiten hoch, der Wohnbereich 12 Einheiten breit), z.B. so aus:

```
1           26
2          2  6
3         2    6
4        2      6
5       2        6
6      2          6
7      8**********8
8      8**********8
9      8**********8
10     8**********8
11     8**********8
12     000000000000
```

Implementieren Sie auch eine drucke-Methode(), die das Array übergeben bekommt und auf der Konsole ausgibt. (**4 Punkte**)

Folgendes ist [optional]

- Füllen Sie das Dach noch mit Schneekörnchen (`.`)
- Speichern Sie das Haus statt in gleich langen Zeilen in unterschiedlich langen Zeilen. Es sollen also nur Felder mit Inhalt existieren und nicht mit Leerzeichen.

Aufgabe 8.2 (Typkonversion – Methoden und Rekursion)
Die *Quersumme* $q(n)$ einer natürlichen Zahl n ist definiert als die Summe ihrer Ziffern. Die *wiederholte Quersumme* hingegen berechnet sich aus der Quersumme solange, bis diese einstellig

ist. Berechnen Sie in dieser Aufgabe mittels Schleifen die *wiederholte Quersumme* einer Ganzzahl. Für die Eingabe der Zahl 195 würde die Berechnung also wie folgt ablaufen:

$$195 \rightarrow\ 1+9+5=15\ \rightarrow 1+5=\ 6$$

Folglich gilt: $q(195) = 6$.

(a) Implementieren Sie zunächst eine Klasse `Quersumme` mit einer **Klassenmethode** `berechneQuersummeSchleife()` zur Berechnung der wiederholten Quersumme (Datentyp `byte`) einer ganzen Zahl n (Datentyp `long`). Die Summe soll mit Hilfe von (einer) Schleife(n) berechnet werden.

Hinweis: Da die übergebende Zahl sehr groß sein können, das Ergebnis sich jedoch im Wertebereich $[0; 9]$ bewegt, sollen bei der Implementierung passende Datentypen, nämlich `long` bzw. `byte`, verwendet werden. Achten Sie dabei stets auf die nötigen Typkonvertierungen.

(b) Implementieren Sie eine zweite Methode zur Berechnung der wiederholten Quersumme, sodass sie mittels **rekursiver** Aufrufe berechnet wird. Die einfache Quersumme (Zwischensumme) können Sie dabei weiterhin in einer Schleife berechnen.

(c) Welche Art der Rekursion verwenden Sie? Klassifizieren Sie die Rekursion hinsichtlich der Kriterien: direkt/indirekt, linear und schlicht (s. Rekursion, Folie 25ff.). Diese Aufgabe darf im Quellcode beantwortet werden.

(2+2+2 = 6 Punkte)

Aufgabe 8.3 (Arrays und Rekursion)

(a) Erstellen Sie eine neue Klasse `ArrayInvertierer` mit einer **rekursiven** Methode, die den Inhalt eines ihr übergebenen 1D-Arrays gefüllt mit Strings invertiert. Auf diese Weise kann z.B. ein deutscher Satz im Array gespeichert werden und dann verkehrt herum ausgegeben werden.

Wichtig: Nicht das übergebene Array soll verändert werden, sondern ein Neues erstellt werden und von der Methode zurückgegeben werden.
Tipp: Sie dürfen dafür gerne auch rekursive Hilfsmethoden benutzen.

(b) Implementieren Sie dann eine main-Methode in der Sie zwei verschieden lange String-Arrays erzeugen und die Wortreihenfolge umkehren lassen. Das Ergebnis soll auf der Konsole ausgegeben werden und könnte z.B. wie folgt aussehen.

```
Den Satz
    Ich find dich einfach klasse!
wuerde Meister Yoda so aussprechen:
    klasse! einfach dich find Ich
Den Satz
    Das war super einfach/schwer
wuerde Meister Yoda so aussprechen:
    einfach/schwer super war Das
```

(3+2 = 5 Punkte)

[optional] Wenn das ursprüngliche String-Array selbst verändert werden, soll braucht die rekursive Methode keinen Rückgabe. Versuchen Sie, diese Aufgabe ohne das Nutzen einer Hilfsmethode zu lösen.

Aufgabe 8.4
In der Zeit vor Weihnachten werden wieder viele Plätzchen und Lebkuchen gebacken. Die Leute naschen fleißig die kreativsten und traditionellen Sorten. Wir bauen uns einen Backautomaten, der uns die Backarbeit abnimmt.

(a) Schreiben Sie als Erstes einen Aufzählungstyp `Plaetzchentyp`, der alle gewünschten Platzchensorten auflistet. Er soll mindestens Lebkuchen, Zimtsterne, Spitzbuben und Makronen enthalten.

(b) Schreiben Sie nun eine Klasse `PlaetzchenDose`. Sie hat Platz für genau eine Plätzchensorte, die sie als Objektfeld speichert. Die maximale Menge an Plätzchen, die darin Platz hat, wird ebenfalls in einem Objektfeld gespeichert. Außerdem speichert sie die Anzahl der gerade darin vorhandenen Plätzchen. Schreiben Sie darin weiterhin

- einen **Konstruktor**, der die maximale Füllmenge und die Plätzchensorte übergeben bekommt. Initial sind keine Plätzchen vorhanden. Schreiben Sie zusätzlich zwei öffentliche Methoden, mit denen man die Sorte der enthaltenen Plätzchen und den aktuellen Inhalt der Dose erfragen kann.
  Handelt es sich dabei um Objekt- oder Klassenmethoden? Fügen Sie eine kurze Begründung als Kommentar im Quellcode hinzu.
- eine Methode `fuelle()`, die eine Anzahl übergeben bekommt und damit den aktuellen Bestand erhöht. Achten Sie darauf, dass die Dose nicht überfüllt wird. Dann soll eine Warnung ausgegeben und die Anzahl der überschüssigen Plätzchen zurückgegeben werden.
- eine Methode `istLeer()`, die angibt, ob die Dose schon leer gegessen ist.

(c) Implementieren Sie nun eine Klasse `BackAutomat`. Diese erleichtert uns die Arbeit und füllt uns die Dosen mit frisch gebackenen Plätzchen. Implementieren Sie dazu folgende Bestandteile:

- Ganzzahlige Felder `mehl`, `zucker`, `butter` und `ei`, die zu Anfang maximal befüllt sind. Der Automat hat Platz für maximal
  - 3000 Einheiten (Gramm) Mehl
  - 50 Eier
  - 3000 Einheiten Zucker und
  - 2000 Einheiten Butter
- `backe()`, die eine Plätzchensorte und die gewünschte Anzahl übergeben bekommt. Sie gibt eine Plätzchendose zurück, die mit der gewünschten Anzahl voll gefüllt ist und somit genau Platz für diese Anzahl an Plätzchen hat. Die benötigten Speisemittel werden dabei reduziert durch Aufruf der folgenden Methode:
- `reduziere()`: Sie bekommt eine Plätzchensorte und die Anzahl der Plätzchen übergeben und reduziert den Bestand des Automaten entsprechend.

  Für je 20 Plätzchen werden folgende Ressourcen verbraucht
  - Lebkuchen: 200g Mehl, 50g Butter und 100g Zucker,
  - Zimtsterne: 2Eier, 50g Mehl, 100g Zucker, 50g Butter
  - Spitzbuben: 100g Mehl, 25g Butter, 50g Zucker

  - Makronen: 3 Eier, 150g Zucker

  Bei der Berechnung des Gesamtverbrauchs für eine gewisse Anzahl an Plätzchen wird der Rest nicht berücksichtigt (ganzzahlige Divison).

- eine Methode `fuelle()`, die den Automaten wieder komplett befüllt.

to be continued

(1+4+2 = 7 Punkte)