TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK
Lehrstuhl für Angewandte Informatik / Kooperative Systeme
Praktikum Grundlagen der Programmierung
Prof. Dr. A. Brüggemann-Klein, R. Palenta, J. Kranz, Dr. A. Reuss

WS 15/16
Übungsblatt 3
30.10.2015

Abgabe: 09.11.2015 (vor 5:00 Uhr)

Implementieren Sie alle Hausaufgaben in einem Package namens *blatt03*.

Hinweis: Der Styleguide wurde auf der Moodle-Seite veröffentlicht. Beachten Sie ihn bei der Bearbeitung der Aufgaben!

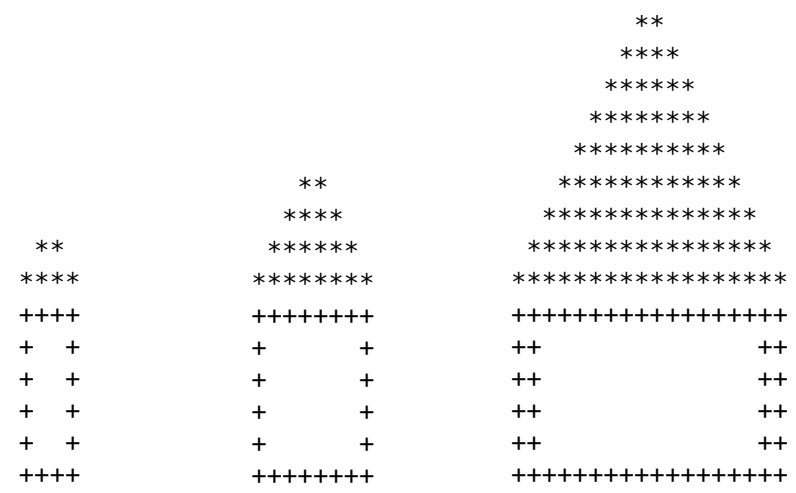
Aufgabe 3.1 (P) Terminal-Häuschen II

In dieser Aufgabe wollen wir die Ausgabe des Symbolhauses vom letzten Übungsblatt erweitern. Zum einen soll die Größe des Hauses und des Innenraumes variabel vom Nutzer bestimmt werden können. Zum anderen sollen in der Implementierung für Dach und Mauern jeweils nur zwei For-Schleifen (verschachtelt) benutzt werden, um über Zeilen und Spalten zu iterieren. Welches Zeichen für eine Position ausgegeben werden muss, soll über If-Abfragen entschieden werden. Es sollen zwei getrennte Methoden verwendet werden, um die Mauer und das Dach auszugeben. Die beiden Methoden werden dann im Hauptprogramm aufgerufen. Nutzen Sie folgendes Programmgerüst für Ihre Implementierung:

```
1  public class Haus {
2    public static void dach () {
3      /*
4       * Todo: Ausgabe des Daches
5       */
6    }
7
8    public static void mauer () {
9      /*
10      * Todo: Ausgabe der Mauer
11      */
12   }
13
14   public static void main (String[] args) {
15     /*
16      * Todo: Aufruf der Methoden zur Ausgabe
17      * von Mauer und Dach
18      */
19   }
20 }
```

Gehen Sie bei Ihrer Implementierung wie folgt vor:

- Zu Beginn soll der Nutzer nacheinander nach der Breite des Hauses h und des Innenraumes i gefragt werden. Diese Werte h und i sollen global abgespeichert werden, damit beide Methoden `dach()` und `mauer()` darauf zugreifen können. Für die Breite des Hauses h muss gelten, dass diese gerade ist, und die Breite des Innenraumes darf maximal $h - 2$ groß sein und gerade, da die Mauern rechts und links vom Innenraum gleich breit sind. Welche weiteren Bedingungen (untere Schranken) gelten für die Werte h und i? Überprüfen Sie die Eingabe des Nutzers auf diese Bedingungen und erfragen Sie so lange eine neue Eingabe, bis diese die Bedingungen erfüllt.
- Passen Sie Ihre aktuelle Implementierung so an, dass die Größe des Hauses den eingegebenen Werten entspricht. Das Dach ist halb so hoch wie das Haus breit. Die Höhe der Mauern ist konstant sechs Zeichen hoch. Beispiele:



- Ändern Sie nun Ihre Implementierung, so dass für Dach und Mauerwerk jeweils nur zwei verschachtelte For-Schleifen verwendet werden. Die äußere Schleife soll über die Zeilen, die innere Schleife über die Spalten (Position in der Zeile) iterieren.

Aufgabe 3.2 (P) Audrücke

Lösen Sie die folgende Aufgabe zunächst per Hand und überprüfen Sie dann Ihre Ergebnisse mit einem Java-Programm.

a) Werten Sie die folgenden Booleschen Ausdrücke schrittweise aus und begründen Sie das Ergebnis. Zu Beginn soll gelten: `a == 42, b == 2, c == true, d == false`.
   i) `(a < b) || !c || !d`
   ii) `(b == a) && c && !d`
   iii) `(b == b) && ((b < a) | c == !d)`
   iv) `((!c | !d) & (b < a)) && ((a != b) & (c != d))`
   v) `((b <= b) && (a <= b) && (b <= a)) || ((b <= a) && (a >= b))`

b) Werten Sie die folgenden Ausdrücke aus. Die Variable `i` soll zu Beginn den Wert 0 haben. Welchen Wert besitzt `i` jeweils nach Auswertung des Ausdrucks aufgrund der Seiteneffekte?
   i) `true | i++==7`
   ii) `true || i++==7`
   iii) `true | i++==7 || i++==7`
   iv) `true || i++==7 | i++==7`
   v) `true | i++==7 | i++==7`

Aufgabe 3.3 (P) Summe inkrementell

In dieser Aufgabe soll die Summe zweier Zahlen berechnet werden. Dabei dürfen Integer-Variablen jedoch nur um eins inkrementiert oder dekrementiert (`x++, x−−`) werden. Gehen Sie dabei wie folgt vor:

- Erstellen Sie eine Methode `public static void read()`, die den Nutzer nacheinander nach zwei Summanden fragt und diese in globalen statischen Integer-Variablen `a` und `b` speichert.
- Verwenden Sie eine Methode `swap()`, um sicherzustellen, dass die im nächsten Schritt beschriebene Schleife nur `min`$(|a|, |b|)$ oft durchlaufen wird. Den Betrag einer Integer-Variablen können Sie mit der Methode `Math.abs(int x)` bestimmen.
- Im folgenden gehen wir davon aus, dass $|a| \leq |b|$. Inkrementieren bzw. dekrementieren Sie den Wert der Variablen `b` $|a|$-mal, so dass anschließend die Variable `b` den Wert $a + b$ hat.

- Geben Sie die Summe auf der Konsole aus.

Aufgabe 3.4 (P) Kontrollflussdiagramm

Zeichnen Sie den Kontrollflussgraphen für das folgende `Java`-Fragment:

```
int a, b, c;
a = 42;
b = 4;
while (a > 10) {
   b = 2*a;
   c = b / 2;
   if (c < a) {
      b=b-a;
   }
   a = a - 4;
}
System.out.println(b);
```

Sie dürfen dabei `System.out.println` durch `write` ersetzen.

Hinweis: Zum Zeichnen können Sie z.B. das Programm *OpenOffice Draw* verwenden. Speichern Sie Ihren Kontrollflussgraphen im *pdf*-Format ab. Das Programm *OpenOffice Draw* finden Sie unter:
http://de.openoffice.org/downloads

Aufgabe 3.5 [3 Punkte] (H) Multiplikationstabelle

Schreiben Sie ein Programm namens `Multtable.java`, das die Multiplikationstabelle für alle Zahlen von 1 bis x ausgibt ($x > 0$). Dabei soll x vom Benutzer eingegeben werden. Für $x = 10$ ist die Ausgabe folgende:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Hinweis: Nutzen Sie folgende Methodenaufrufe, um die Ausgabe zu formatieren:

a) `System.out.print('\t');` (erzeugt einen Tabulator),
b) `System.out.print(i);` (gibt die Zahl i aus) und
c) `System.out.println();` (erzeugt einen Zeilenvorschub).

Hinweis: Für das Einlesen von Benutzereingaben gilt generell, dass

- eingegebene Zahlen auf gültige Werte überprüft werden sollen und
- eingegebene Zeichenketten daraufhin überprüft werden sollen, ob sie dem erforderlichen Format entsprechen.

Macht der Benutzer eine ungültige Eingabe, so soll ein entsprechender Hinweis ausgegeben und die Eingabe erneut eingelesen werden, bis der Benutzer einen gültigen Wert eingibt.

Aufgabe 3.6 [5 Punkte] **(H) Sternkreuz**

Die folgenden Abbildungen zeigen aus dem Symbol '*' bestehende Quadrate mit Diagonalen.

```
*******          ********
**   **          **    **
* * * *          * *  * *
*  *  *          *  **  *
* * * *          *  **  *
**   **          * *  * *
*******          **    **
                 ********
```

Schreiben Sie ein Programm `Sternkreuz.java`, welches diese Muster mit beliebiger Kantenlänge erzeugt und in einer Variablen vom Typ `String` speichert. Legen Sie dazu statische Variablen `kantenlaenge` vom Typ `int` und `sternkreuz` vom Typ `String` an und schreiben Sie die folgenden (parameterlosen) Methoden:

- `input()`: Fragt die Kantenlänge vom Benutzer ab (auf der Konsole) und speichert sie in der Variablen `kantenlaenge`. Die Kantenlänge soll mindestens 2 sein.
- `draw()`: Zeichnet ein Sternkreuz mit Kantenlänge `kantenlaenge` in die Variable `sternkreuz`. (Zeilenumbrüche werden als `\n` dargestellt.)
- `output()`: Gibt das Sternkreuz auf der Konsole aus.

Das Hauptprogramm soll **mittels dieser Methoden** so lange vom Benutzer eine Kantenlänge erfragen und das dazugehörige Sternkreuz ausgeben, bis das Sternkreuz mit der Kantenlänge 42 oder das mit Kantenlänge 2 ausgegeben wird.

Aufgabe 3.7 [5 Punkte] **(H) Kontrollflussdiagramm**

Zeichnen Sie den Kontrollflussgraphen für das folgende `Java`-Fragment:

```
int i, loop, erg;
erg = 1;
loop = 4;
while (erg != 42) {
   i = 1;
   if (erg > 42){
      erg = erg - 16;
   }
   while (i < loop) {
      i = i + 1;
      erg = erg + 19;
   }
   System.out.println(erg);
}
```

Sie dürfen dabei `System.out.println` durch `write` ersetzen.

Hinweis: Zum Zeichnen können Sie z.B. das Programm *OpenOffice Draw* verwenden. Speichern Sie Ihren Kontrollflussgraphen im *pdf*-Format ab. Das Programm *OpenOffice Draw* finden Sie unter:
http://de.openoffice.org/downloads