Übungsblatt Rekursion Java

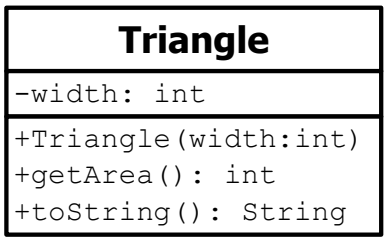
1. Triangle Numbers

Es soll ein Programm zur rekursiven Berechnung von Dreiecksflächen entworfen werden. Betrachten Sie das folgende Dreieck:

```
[]
[] []
[] [] []
```

Unter der Annahme, dass jedes Klammernpaar die Fläche 1 hat, ergibt sich eine Fläche von 6 Einheiten. Dies ist die dritte Dreieckszahl.

a) Legen Sie eine Java-Klasse gemäß folgendem Diagramm an:



Der Aufruf von:

```
Triangle t = new Triangle(3);
System.out.println(t);
System.out.println("Fläche: " + t.getArea());
```

soll obiges Dreieck mit zugehörigem Flächeninhalt ausgeben. Verwenden Sie den Debugger und Kontrollausgaben auf der Konsole, um den Ablauf des Programms genau zu analysieren.

b) Prüfen Sie bis zu welchem Limit sich die Dreieckszahlen rekursiv berechnen lassen. Mit welcher Einstellung lässt sich dieses Limit verschieben?
c) Ändern Sie ihr Programm so, dass das Dreieck andersherum angezeigt wird.

```
[] [] []
[] []
[]
```

d) Erweitern Sie die Klasse Triangle um die Methode getAreaIterativ(), die eine Schleife zur Berechnung der Fläche verwendet.

e) Wie würde sich die Flächenberechnung auf die Berechnung von Quadraten übertragen lassen? Erstellen Sie die Klasse Square ähnlich der Klasse Triangle (nur ohne toString-Methode)

2. Quersumme

Schreiben Sie ein Programm zur rekursiven Berechnung der Quersumme einer Ganzzahl.

3. Permutationen

Es soll eine Klasse zur Ausgabe aller Permutationen einer Zeichenkette programmiert werden. Eine Permutation entsteht durch das Vertauschen von Zeichen. Zum Beispiel das Wort „das" hat 6 Permutationen.

```
das
dsa
ads
asd
sda
sad
```

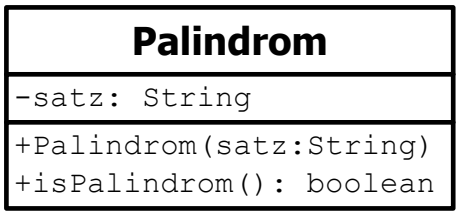
Verwenden Sie folgendes Design:

PermutationsGenerator
-word: String
+PermutationsGenerator(word:String) +getPerm(): ArrayList<String>

Rekursiver Ansatz:
(Einfacher Fall) Falls das Wort nur aus einem Buchstaben besteht, haben wir nur eine Permutation. Das Wort selbst.
(Reduktion der Komplexität) Ist das Wort länger, so schneidet man immer einen Buchstaben aus dem Wort heraus und berechnet alle Permutationen für das Restwort. Hinterher werden alle Worte wieder mit dem Buchstaben kombiniert.
Achtung: Hat ein Wort n Buchstaben, so sind die Anzahl der Permutationen n!, diese Zahl wird sehr schnell sehr groß.

4. Satzpalindrome

Es gilt eine Klasse zu entwickeln, die prüft, ob bei einem gegebenen Satz ein Palindrom vorliegt. Bei dem rekursiven Test ist Groß-/Kleinschreibung egal, Satzzeichen und Leerzeichen dürfen auch ignoriert werden.
Verwenden Sie folgendes Design:



Ein mögliches Hauptprogramm:

```
public static void main(String[] args) {
	String[] test = {
	"Die Liebe ist Sieger, stets rege ist sie bei Leid.",
	"Die Niere bot Komik: nass sank im Oktober ein Eid.",
	"Ein Examen? Ne Maxe, nie!",
	"Es eilt, immer ahnend Nebel, reger der Flegel Fred, reg' erlebend
		nen Harem mit Liese",
	"Trug Tim eine so helle Hose nie mit Gurt?"};

	for (String s : test) {
		Palindrom pali = new Palindrom(s);
		System.out.println(s + " | " + pali.isPalindrom());
	}
}
```