Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Christian Böhm
Annahita Oswald, Bianca Wackersreuther

München, 08.01.2010

Einführung in die Programmierung

WS 2009/10

Übungsblatt 10: Mehr zu Objektorientierung, Arbeiten mit Strings

Besprechung: 18./20./21./22.01.2010

Ende der Abgabefrist: Montag, 18.01.2009 10:00 Uhr.

Hinweise zur Abgabe:
Geben Sie bitte Ihre gesammelten Lösungen zu diesem Übungsblatt in einer Datei `loesung10.zip` unter `http://www.pst.ifi.lmu.de/uniworx/` ab.
Bitte beachten Sie, dass die Aufgabe 10-3 nicht in die Bonusregelung eingeht. Bereiten Sie diese aber bitte trotzdem vor, damit Sie der Übung optimal folgen können.

Aufgabe 10-1 *Objektreferenz und* `null` **10 Punkte**

Die Datei `Text.java` hat folgenden Inhalt:

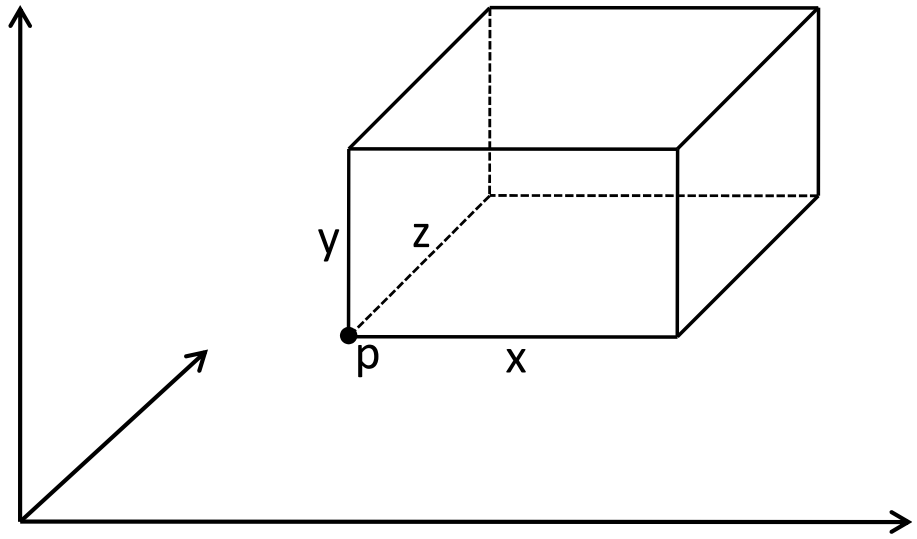
```
public class Text {
    public static void main(String[] args) {
        String text;
        // *1*

        text = null;
        // *2*

        text = "";
        // *3*

        text = "ABC";
        // *4*
    }
}
```

Erklären Sie jeweils kurz (2-3 Sätze), welchen *Wert* der Ausdruck `text.length()` an den mit Kommentaren markierten Stellen `*1*` bis `*4*` jeweils hat. Beschreiben Sie dabei auch, *warum* gerade dieser Wert zustande kommt.

Schreiben Sie Ihre Antwort in eine Datei `text.txt` und geben Sie diese ab.

Aufgabe 10-2 *Objektorientierte Modellierung* **10 Punkte**

Bei Quadern handelt es sich um Objekte, deren Kanten achsenparallel in einem dreidimensionalen Koordinatensystem liegen. Quader werden spezifiziert (siehe Bild) durch einen der Eckpunkte `p` und die Länge der an `p` angrenzenden Kanten `x` (parallel zur x-Achse), `y` (parallel zur y-Achse) und `z` (parallel zur z-Achse).



Hinweis: Achten Sie im Folgenden in der gesamten Aufgabe auf sinnvolle Datenkapselung.

(a) Implementieren Sie eine Klasse `Punkt3D` zur Verwaltung dreidimensionaler Punkte in einem reellen Koordinatensystem als Erweiterung der Klasse `Punkt2D`, die Sie auf der Vorlesungs-Homepage finden:

Definieren Sie neben einem geeigneten Konstruktor und geeigneten Attributen die folgende Methode:
`void verschiebe(double x, double y, double z)`,
die den Punkt um `x` entlang der x-Achse, um `y` entlang der y-Achse und um `z` entlang der z-Achse verschiebt.

(b) Implementieren Sie eine Klasse `Quader`, die die Klasse `Punkt3D` aus der vorherigen Teilaufgabe sinnvoll verwendet. Die Klasse soll einen geeigneten Konstruktor und geeignete Attribute bereitstellen. Implementieren Sie für die Klasse Quader zusätzlich folgende Methode:
`void verschiebe(double x, double y, double z)`,
die den Quader um `x` entlang der x-Achse, um `y` entlang der y-Achse und um `z` entlang der z-Achse verschiebt.

Aufgabe 10-3 *`String` vs. `StringBuilder`/`StringBuffer`* **0 Punkte**

Als angehende Akademiker sollten Sie stets bereit sein, die Äußerungen Ihres Dozenten zu hinterfragen. Kann es wirklich sein, dass die Konkatenation von Strings problematisch ist? Das ist doch eine Operation, die ständig vorkommt und sogar mit einem eigenen Operator unterstützt wird!

Zur Überprüfung der Effizienz von `String`-Konkatenationen gegenüber der Verwendung von `StringBuilder` und `StringBuffer` schreiben Sie in einer Klasse `StringEffizienz` drei Methoden

- `public static String buildString(int n)`
- `public static String buildStringBuilder(int n)`
- `public static String buildStringBuffer(int n)`

die jeweils einen `String` zurückgeben, der die Zahl 10^n für $n \geq 0$ darstellt, also einen `String` bestehend aus dem Zeichen `'1'` n-mal gefolgt von dem Zeichen `'0'`. Die n-fache Wiederholung soll in jeder der drei Methoden gleichartig als **`for`**-Schleife implementiert sein, nur wird der `String` in den verschiedenen Methoden unterschiedlich gebildet:

- `buildString` verwendet fortgesetzte `String`-Konkatenation
- `buildStringBuilder` verwendet einen `StringBuilder`, um die Zeichenkette aufzubauen
- `buildStringBuffer` verwendet einen `StringBuffer`, um die Zeichenkette aufzubauen

Implementieren Sie nun in der `main`-Methode der Klasse einen Effizienz-Test zur Untersuchung des Laufzeitverhaltens der Methoden. Dazu können Sie sich natürlich an der Klasse `Laufzeit` (Übungsblatt 8, Aufgabe 2) orientieren. Pro Durchlauf der Klasse `StringEffizienz` soll jeweils nur eine der drei Methoden getestet werden, d.h. in der `main`-Methode darf jeweils nur eine Methode aufgerufen werden. Sie können dies z.B. durch Verwendung unterschiedlicher Kommandozeilenargumente für die einzelnen Methoden realisieren:

Beispiel:
Der Aufruf
`java StringEffizienz -string`
führt in der `main`-Methode zu einem Aufruf der Methode `buildString`,
der Aufruf
`java StringEffizienz -builder`
führt in der `main`-Methode zu einem Aufruf der Methode `buildStringBuilder`.

Was beobachten Sie?