Aufgabe 2

Methoden in Programmen funktionieren nicht immer für alle möglichen Eingaben - klassische Beispiele sind die Quadratwurzel einer negativen Zahl oder die Division durch Null. Zulässige (bzw. in negierter Form unzulässige) Eingabewerte sollten dann spezifiziert werden, was mit Hilfe sog. assertions geschehen kann. Assertions prüfen zur Laufzeit den Wertebereich der Parameter einer Methode, bevor der Methodenkörper ausgeführt wird. Wenn ein oder mehrere Argumente zur Laufzeit unzulässig sind, wird eine Ausnahme geworfen.

```
3  public class Assertion {
4    public static double[][] magic(double[][] A, double[][] B, int m) {
5      double[][] C = new double[m][m];
6      for (int i = 0; i < m; i++)
7        for (int j = 0; j < m; j++)
8          for (int k = 0; k < m; k++)
9            C[i][j] += A[i][k] * B[k][3];
10     return C;
11   }
12 }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/Assertion.java`

Betrachten Sie das folgende Java-Programm:

(a) Beschreiben Sie kurz, was dieses Programm tut.

(b) Implementieren Sie drei nützliche Assertions, die zusammen verhindern, dass das Programm abstürzt.

(c) Wann und warum kann es sinnvoll sein, keine expliziten Assertions anzugeben? Erläutern Sie, warum das potentiell gefährlich ist.

(d) Assertions können wie oben sowohl für Vorbedingungen am Anfang einer Methode als auch für Nachbedingungen am Ende einer Methode verwendet werden. Solche Spezifikationen bilden dann sog. Kontrakte. Kontrakte werden insbesondere auch statisch verwendet, d. h. nicht zur Laufzeit. Skizzieren Sie ein sinnvolles Anwendungsgebiet und beschreiben Sie kurz den Vorteil der Verwendung von Kontrakten in diesem Anwendungsgebiet.