Übungsblatt 7

Abgabe: 09.12.2012

Aufgabe 1 Der TutorInnen Leid... (70%)

TutorInnen haben es nicht leicht.[1] Oft sind Methoden nicht ganz richtig implementiert und die Fehler müssen gefunden und korrigiert[2] werden. Manchmal gibt es auch keine Dokumentation. Könnt ihr helfen?

Ladet dazu das Archiv *Quiz.zip* herunter, packt es aus und nutzt es als Ausgangspunkt für eure Bearbeitung.

Methode 1 (10%). Marla Mystery hat die folgende Methode abgegeben. Leider ist sie nicht dokumentiert. Was berechnet sie und wie tut sie das?

```
3     public int method(int i)
4     {
5         int a = i;
6         int b = 1;
7         while (a - b > 1) {
8             a = (a + b) / 2;
9             b = i / a;
10        }
11        return (a + b) / 2;
12    }
```

Methode 2 (10%). Man kann in Java auch *$* und *_* zum Benennen von Variablen und Methoden verwenden. Tut man aber nicht. Karl Krypto hat dennoch die nachfolgende Methode abgegeben. Was berechnet sie und wie?[3]

```
14    public int $(int _)
15    {
16        return _ == _ + _ ? -~_ : _ * $(_ - _ / _);
17    }
```

Methode 3 (10%). Karsten Knappdaneben hat die folgende Methode zur Bestimmung des Maximums der Werte eines Arrays geschrieben. Sie funktioniert auch oft. Wann nicht? Was muss geändert werden?

```
19    public int max(int[] values)
20    {
21        int maximum = 0;
22        for (int value : values) {
23            if (value > maximum) {
24                maximum = value;
25            }
26        }
27        return maximum;
28    }
```

Methode 4 (15%). Die nachfolgende Methode von Maria Matrize soll eine übergebene Matrix (ein zweidimensionales Array mit gleicher Zeilen- und Spaltenzahl) transponieren, d.h. an der

[1] Die echten PI-1-Tutoren weisen darauf hin, dass dieser Zettel vollkommen unrealistisch ist, z.B. weil die herunter zu ladenen Dateien tatsächlich unter den angegebenen Namen auf dem Server liegen.

[2] Tutoren testen ihre Korrekturen natürlich auch.

[3] Hierzu müsst ihr möglicherweise etwas über Operatoren herausfinden, die ihr noch nicht kennt.

diagonalen Achse spiegeln. Dazu werden sich diagonal gegenüber liegende Elemente mit einem so genannten Dreieckstausch ausgetauscht. Überraschenderweise ist das Ergebnis der Methode identisch mit der Eingabe. Was ist falsch?

```
30    public int[][] transpose(int[][] matrix)
31    {
32        for (int y = 0; y < matrix.length; ++y) {
33            for (int x = 0; x < matrix[y].length; ++x) {
34                int temp = matrix[x][y];
35                matrix[x][y] = matrix[y][x];
36                matrix[y][x] = temp;
37            }
38        }
39        return matrix;
40    }
```

Methode 5 (25%). Ein Array kann man z.B. so sortieren: Das Array wird in zwei ineinander verschachtelten Schleifen durchlaufen. Die äußere Schleife läuft rückwärts über das Array und grenzt den noch zu sortierenden Teil von dem bereits sortierten Teil des Arrays ab. Die innere Schleife durchläuft den noch nicht sortierten Teil des Arrays und vertauscht immer zwei aufeinander folgende Einträge, wenn sie verkehrt herum geordnet sind. Wenn die innere Schleife zu Ende ist, ist an der oberen Grenze des noch nicht sortierten Teils des Arrays ein weiteres richtig einsortiertes Element entstanden, d.h. der unsortierte Teil des Arrays wird mit jedem Durchlauf der äußeren Schleifen um ein Element kleiner, bis das ganze Array sortiert ist. Claudius Chaos hat die nachfolgende Lösung vorgelegt, die aber noch zahlreiche Fehler enthält. Da man sie nicht einmal übersetzen kann, hat er sie auskommentiert. Könnt ihr sie korrigieren?

```
42    /*
43    public int[] sort(int array)
44    {
45    int i = array.length();
46    while (i > 1) {
47    int j = 0;
48    while (j < i) {
49    if (array[j - 1] > array[j]) {
50    int temp = array[j - 1];
51    array[j] = array[j - 1];
52    array[j - 1] = temp;
53    }
54    i = i + 1;
55    }
56    j = j - 1;
57    }
58    return array;
59    }
60    */
```

Aufgabe 2 . . . ist der Studierenden Spiel (30%)

Werner Wirrwarr hat eine in jeder Hinsicht sehr farbige Implementierung des Spiels *Pong* abgegeben. Ladet euch die Datei *Pong.zip* aus Stud.IP herunter und erklärt daran die Begriffe *Kopplung*, *Kohäsion*, *Code-Duplizierung* und *Entwurf nach Zuständigkeiten*. Wie findet ihr Werners Abgabe in Bezug auf diese Kriterien? Habt ihr noch weitere Kritik an seinem Code? Was würdet ihr anders machen?

Aufgabe 3 Bonusaufgabe: Macht es anders (5%)

Stellt Werners Abgabe bei gleicher Funktionalität so um, dass sie den Kriterien gut entworfener und geschriebener Software entspricht. Nutzt *Pong* zur Abgabe und kopiert auch eure Dateien *Quiz.java* und *QuizWorld.java* mit hinein.