Fragen, Antworten, Kommentare

Spätestens jetzt ist ein Zeitpunkt erreicht, zu dem Sie mit sich selbst klären müssen, ob Sie die Klausur mitschreiben werden. Ein persönlicher Tipp von mir ist, dass die Antwort „ich mache das von meiner Vorbereitung abhängig“ fast immer falsch ist. Sie können zu diesem Zeitpunkt beurteilen, ob Sie sich die Klausur zutrauen. Natürlich sollen Sie in der Vorbereitung weiter programmieren. Sollten Sie nicht schreiben wollen, sollten Sie den Praktikumsleitern mitteilen, dass Sie auch das Praktikum nicht bestanden haben. Starten Sie dann unbedingt im nächsten Semester die Programmierausbildung mit Vorlesung und Praktikum neu; generell ist es meist sinnvoll sich selbst eine solche zweite (und letzte) Chance zu geben. Durch die etwas anderen didaktischen Ansätze meiner Kollegen haben auch die Praktika einen sehr großen Mehrwert. Die dümmste Idee ist zu denken, dass man ja immerhin das Praktikum schon bestanden hat, auch wenn die Klausur nicht geschrieben wurde. Erfahrungen zeigen, dass eine Vorbereitung nebenbei auf eine Klausur praktisch nie klappt und das Studium spätestens im dritten Semester endet.

Falls Sie eine persönliche Einschätzung Ihrer Situation aus meiner Sicht wünschen, können Sie sich gerne bei mir melden. Generell können sie auch zu fachlichen Fragen mir in der Vorlesungszeit E-Mails schicken oder/und einen Zoom-Termin vereinbaren.

Zu Aufgabe 33:

Generell ist es im ersten Semester wichtig überhaupt eine algorithmische Lösung zu finden und diese in Code umsetzen zu können. Danach sollte man in der Lage sein über Optimierungen nachzudenken, gibt es z. B. schnellere Lösungen oder Lösungen, die weniger Variablen, d. h. Speicher verbrauchen. Hier muss man nicht auf alle Lösungsvarianten kommen, die anderen Lösungen aber verstehen. Das ist ein Thema, das in mehreren Veranstaltungen, u. a. Algorithmen und Datenstrukturen, weiter vertieft wird.

Ein Beispiel ist die geforderte Methode werteNachHinten(int w), die alle Exemplare von w nach hinten in die Messreihe schiebt.

erste Lösungsidee: Durchlaufe alle Werte, wenn w gefunden wird, lasse alle nachfolgenden Werte um eine Position nach vorne rücken (this.messwerte.set(i, this.messwerte.get(i+1)) und schreibe an die letzte Stelle w. Kurze Analyse: Korrekt, benötigt zwei Schleifen ineinander

zweite Lösungsidee: Durchlaufe alle Werte, nutze eine Hilfsliste l und einen Hilfszähler z. Wenn der Wert w ist, erhöhe z um 1, wenn nicht hänge den Wert an die Hilfsliste. Am Ende hänge z-mal den Wert w an die Hilfsliste und mache danach die Hilfsliste zu this.messwerte. Kurze Analyse: Korrekt, hat nur zwei einfache Schleifen (schneller), benötigt aber zusätzlich Speicher für eine Hilfsliste

dritte Lösungsidee: Durchlaufe die Liste mit einer Laufvariable i und merke in einer zweiten Hilfsvariable gf wieviele w schon gefunden wurden (am Anfang gf = 0). Ist der betrachtete Wert an der Position i gleich w erhöhe gf. Ist der Wert an der Position i ungleich w und gf>0, setze den Wert an der Stelle i - gf auf den Wert an der Position i. Anschaulich ist dann die Liste bis zur Position i - gf schon korrekt. Schreibe am Ende gf-Mal den Wert w an das Ende der Liste. Analyse: Korrekt, nur zwei einfache Schleifen, nur zwei Hilfsvariablen. Umsetzung

```
 public void werteNachHinten(int w) {
   if (this.messwerte == null) {
     return;
   }
   int gefundeneWerte = 0;  // gf
   for (int i = 0; i < this.messwerte.size(); i++){
     if (this.messwerte.get(i) == w) {
       gefundeneWerte++;
     } else {
       //  fuer gefundeneWerte = 0 passiert nix, deshalb kein if
       this.messwerte.set(i-gefundeneWerte, this.messwerte.get(i));
     }
   }
   for (int i = 0; i < gefundeneWerte; i++) {
     this.messwerte.set(this.messwerte.size() - 1 - i, w);
   }
 }

/*****************************
 Beispiel: 3 nach hinten
 3 3 2 4 4
 *
 i=0 gefundeneWerte=1
   *
 i=1 gefundeneWerte=2
     *
 i=2 gefundeneWerte=2 (2 an Position  i-gefundeneWerte)
 2 3 2 4 4
       *
 i=3 gefundeneWerte=2 (4 an Position  i-gefundeneWerte)
 2 4 2 4 4
         *
 i=4 gefundeneWerte=2 (4 an Position  i-gefundeneWerte)
 2 4 4 4 4
 ----
 hinten gefundeneWerte mal Wert setzen
 2 4 4 3 3
 */
```