

Fragen, Antworten, Kommentare zur aktuellen Vorlesung

Hinweis: Die Diskussion einer Beispiellösung zur Aufgabe 22 mit equals(.) für Linie finden Sie unter <https://youtu.be/2gpR9bMwSLY> (30:07).

Da die Versuche zur ersten größeren Aufgabe mit den Säulendiagrammen von recht unterschiedlicher Qualität sind, habe ich einen vollständigen, in der Entstehung bewusst nicht optimierten Entstehungsweg in einem Video <https://youtu.be/cITeJ7O3dqs> (103:03) zusammengefasst.

Im Video angemerkt, hier nochmals betont, generell ist zu klären, wer Objekte erstellen darf. Dies soll nur durch Objekte einer Klasse, typischerweise einer Verwaltungs- oder Controller-Klasse erfolgen. In obiger Aufgabe ist die Frage, wer Saeulen-Objekte erstellen darf. Dies könnte die Saeulendiagrammsteuerung sein:

```
switch(eingabe) {
  case 1:{
    this.io.ausgeben("neuer Titel: ");
    String titel = this.io leseString();
    this.io.ausgeben("Wert: ");
    int wert = this.io leseInteger();
    this.sd.hinzufuegen(new Saeule(titel, wert)); // eher kritisch
    break;
  }
}
```

Besser ist es, wenn dies im Saeulendiagramm selbst passiert. So ist sichergestellt, dass die Steuerung die Säulen gar nicht kennen muss. Je weniger Abhängigkeiten zwischen Klassen, generell desto besser ist es.

```
switch(eingabe) {
  case 1:{
    this.io.ausgeben("neuer Titel: ");
    String titel = this.io leseString();
    this.io.ausgeben("Wert: ");
    int wert = this.io leseInteger();
    this.sd.hinzufuegen(titel, wert);
    break;
  }
}
```

Mir ist nicht nur bei diesem Aufgabenblatt aufgefallen, dass bei Fehlern nicht alle Studierende den Debugger einsetzen. Die Kenntnis wie ein Debugger funktioniert ist allerdings eine elementare Grundfähigkeit in der Programmierung. Sie können genau erkennen, welche Schritte im Programm ausgeführt werden und welche Werte die Variablen haben. Beim systematischen Debuggen überlegt man vor der Ausführung des nächsten Schritts immer, was man erwartet, was passiert und überprüft dann seine Erwartungen.

Hinweis: Ja, mit der Aufgabe zum Säulendiagramm haben wir den Bereich der meist sehr einfachen Aufgaben verlassen und kommen zu den Aufgaben zu grundlegenden Programmierfähigkeiten.

Frage oder Bemerkung: Es gibt Hinweise auf die hohe Stundenbelastung durch die Praktikumsaufgaben.

Antwort: Ja die Belastung ist hoch, aber im üblichen Rahmen eines Studiums. Beachten Sie, dass es sich um eine Veranstaltung mit 10 Leistungspunkten handelt, grob 300 Arbeitsstunden für strukturiert arbeitende Studierende (zentrale Anforderung der Studierfähigkeit) im Semester, aber noch ohne Programmierwissen. Dies ergibt pro Woche im Semester ca. 14 Stunden Arbeit für Programmierung 1. Das sind 3 Stunden Vorlesung, 2 Stunden Praktikum (30+90 Minuten) und 7 Stunden zur Vor- und Nachbereitung und Bearbeitung von Praktikumsaufgaben (damit im Umfang von mindestens 5 Stunden). Sollten Sie bei einer Aufgabe sehr lange brauchen, sollten Sie sie schieben und mit anderen Personen (Prof, Mitarbeiter, andere Studis) später reden. Sollten Sie mehrfach deutlich länger brauchen, wissen Sie, dass Sie den aktuell gestellten Anforderungen momentan nicht entsprechen. Hier müssen Sie dann überlegen, ob Sie mehr Zeit investieren wollen und können oder z. B. Programmierung mit ihrem erreichten Vorwissen im nächsten Semester nochmal versuchen. (Meine Erfahrung zeigt, dass ein dritter Versuch sinnlos ist.)

Berechnung: Semester hat 26 Wochen, wir ziehen 3 Wochen Urlaub ab. Die Vorlesungszeit hat maximal 15, meist 14 bis 12 Veranstaltungswochen. Sind im Maximalfall $15 \cdot 14 = 210$ Stunden. Die restliche Zeit dient der Prüfungsvorbereitung oder/und Durchführung (bei Hausarbeiten) und der Nachbereitung. Sie umfasst dann $11 \cdot 8$ Stunden. Wird die Nachbereitung eigentlich geprüft? Indirekt ja, da nicht alle wichtigen, meist technischen Themen wie z. B. Installation und Nutzung von Git, CodeTogether und Docker, in Veranstaltungen erklärt werden. Das Wissen ist aber sehr hilfreich und teilweise notwendig. Generell sollten Sie an eigenen Programmierprojekten arbeiten, um Ihre Fähigkeiten zu trainieren und zu verbessern.