**Fragen, Antworten, Kommentare zur aktuellen Vorlesung**

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 von 2020 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 Sequenzdiagramm 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.