Learning programming is hard — teaching it well is even more challenging. At university, the focus is often on functional correctness and neglects the topic of clean and maintainable code, despite the dire need for developers with this skill set within the software industry. We present a feedback-driven teaching concept for college students in their second to third year that we have applied and refined successfully over a period of more than six years and for which received the faculty's teaching award. Evaluating the learning process within a semester of student submissions (n=18) with static code analysis tools shows satisfying progress. Identifying the correction of the in-semester programming assignments as the bottleneck for scaling the number of students in the course, we propose using a knowledge base of code examples to decrease the time to feedback and increase feedback quality. From our experience in assessing student code, we have compiled such a knowledge base with the typical issues of Java learners' code in the format of before/after comparisons. By simply referencing the problem to the student, the quality of feedback can be improved, since such comparisons let the student understand the problem and the rationale behind the solution. Further speed-up is achieved by using a curated list of
static code analysis checks to help the corrector in identifying violations in the code swiftly. We see this work as a foundational step towards online courses with hundreds of students learning how to write clean code.

**Kongress- / Buchtitel:**
Proceedings of the 1st Workshop on Innovative Software Engineering Education

**Jahr:**
2018

**Monat:**
Mar

**Seiten:**
24--27

**Serien-ISSN:**
1613-0073

**WWW:**

**Occurences:**
- Hochschulbibliographie > 2018 > Fakultäten > Informatik > Informatik 11 - Lehrstuhl für Connected Mobility (Prof. Ott)
- Einrichtungen > Fakultäten > Fakultät für Informatik > Lehrstühle der Informatik > Informatik 11 - Lehrstuhl für Connected Mobility (Prof. Ott)

**Entries:**