# Objektgymnastik: Jeff Bay’s Object Calisthenics

Jeff Bay’s “Object Calisthenics” [1] sind neun Regeln in einem Training die Softwareentwickler dazu herausfordern, besseren Code zu schreiben. In gleichnamigen Essay beschreibt Jeff Bay die Regeln der Object Calithenics. Jeff Bay sieht vor die Regeln in einer kurzen, zwei Stündigen oder halbtägigen Übung anzuwenden. Das Übungsprojekt mit etwa 1k Zeilen Quellcode dient hierbei ausschließlich dem Übungszwecken. Die strikte Regelbefolgung des Entwicklers stellt bei der Übung die eigentliche Herausforderung dar. Durch das strikte Anwenden der Regeln das damit verbundene Verständnis der Entwurfsmuster und Prinzipien hinter den Regeln verbessert der Entwickler seine Programmierfähigkeiten. Lauf Jeff Bay ist dies hilfreich, wenn er in realen Projekten Anwendungen entwickelt. Das durch die Übung hinzugewonnenen Wissen kann nun praktisch angewendet werden, natürlich aber ohne die strikte Regelbefolgung der Object Calisthenics. Nach Jeff Bay wird die Übung “give new programmers an opportunity to learn best practices while writing their own code.” [1, p.70]. (Programmierern die Möglichkeit geben eigenen Code zu schreiben und dabei, beste Vorgehensweisen zu lernen).

Bevor Jeff Bay die Regeln beschreibt, geht er auf die Problemeentstehung

In seinem Essay beschreibt Jeff Bay die neun Regeln, welche er für die Übung aus gängigen Prinzipien und Praktiken heraus erstellt hat. Er hebt

According to Jeff Bay the exercise “will give new programmers an opportunity to learn best practices while writing their own code.” [1, p.70]. He furthermore states that it is easy to create procedural code, and that programmers coming from procedural programming are furthermore stuck in old habits [1, p.80]. The disadvantages of procedural code are obvious: missing bundling of data and behavior, the difficulties in maintainability, bad understandability, lack of modularity, missing structure and missing overview, and the often discusses problem of reusability [1]. By going through a process of rethinking during a small exercise, the developer’s perspective on existing code and the way he will write code in the future might change radically. When the developer’s knowledge of object orientation is increased after he conducted the exercise, the resulting code might have a higher cohesion, looser coupling, less or better no duplication, a better encapsulation, better testability, and an increased readability.

Die Idee hinter den Object Calisthenics ist, Durch die strikte Regelbefolgung während der Übung wird der Entwickler dazu herausgefordert

Jeff Bay’s "Object Calisthenics" [1] are nine rules that train the software developer to write better object oriented code. In his paper, he created concrete rules out of general software principals and patterns. These rules shall be applied in a short exercise, usually about two to four hours with around 1k lines of code. With these concrete rules the trainee doing the exercise can improve his software development skills, which is helping him when applying general software principals and patterns to real world software projects.

According to Jeff Bay the exercise “will give new programmers an opportunity to learn best practices while writing their own code.” [1, p.70]. He furthermore states that it is easy to create procedural code, and that programmers coming from procedural programming are furthermore stuck in old habits [1, p.80]. The disadvantages of procedural code are obvious: missing bundling of data and behavior, the difficulties in maintainability, bad understandability, lack of modularity, missing structure and missing overview, and the often discusses problem of reusability [1]. By going through a process of rethinking during a small exercise, the developer’s perspective on existing code and the way he will write code in the future might change radically. When the developer’s knowledge of object orientation is increased after he conducted the exercise, the resulting code might have a higher cohesion, looser coupling, less or better no duplication, a better encapsulation, better testability, and an increased readability.

What follows are the nine rule of the Object Calisthenics:

1. Use only one level of indentation per method
2. Don’t use the else keyword
3. Wrap all primitives and strings
4. Use only one dot per line
5. Don’t abbreviate
6. Keep all entities small
7. Don’t use any classes with more than two instance variables
8. Use first-class collections
9. Don’t use any getters/setters/properties

There is a pattern or a principal behind every rule. By applying the rules the developer is forced to think about these patterns and he has to implement them accordingly. “???Meine empfehlung”: Try it! Conduct the exercise for example with a simple TicTacToe or Minesweeper game – a big cup of coffee might help!

When completing the training challenge the developer has to observe his own code and check if his own coding style satisfies the nine rules of the Object Calisthenics. Tool support could shorten the time of the training and furthermore guarantee that the developer sticks to the given rules.

In a student research paper, I am currently evaluating the development of tool support for the Object Calisthenics. In the course of the paper I already implemented a prototype. The tool created during the research of the paper is realized in form of an Eclipse plugin. It successfully validates all rules of the Object Calisthenics properly and indicates corresponding violations. The prototype is hosted “on github???’. A blog post about the implementation of rule violation will follow when the research paper is “abgeschlossen” successfully.