CoderDojo

Test-driven development

By driving development with automated tests and then eliminating duplication, any developer can write reliable, bug-free code no matter what its level of complexity. (Kent Beck, 2002)

CoderDojo

What do we want to achieve?

- improve the lives of the users of our software
- reduce the onboarding time of new team members
- changes in our code should be easy and without risks
- reduce the maintenance efforts
- want to get feedback as early as possible
- easy to understand documentation of the software

CoderDojo

How do we achieve it?

- write new code only if an automated test has failed
- eliminate duplication
- run the tests as often as possible

What are the prequisites?

- we (the developers) have to write tests
- version control system (Git, Subversion, ...)
- the development environment has to support refactoring and running the tests fast
- our design must consist of many highly cohesive, loosely coupled components

Programming phases

1. Red:

   Write a little test that doesn't work, and perhaps doesn't even compile at first.

2. Green:

   Make the test work **quickly**, committing whatever sins necessary in the process.

3. Refactor:

   Eliminate all of the duplication created in merely getting the test to work.

The goal of the game is to write the smallest test method that represents real progress toward your end goal.

CoderDojo

Demo

We want to implement multi-currency money that should support:

- creation of Euros (EUR) and Swiss Franc (SFR)
- operations like „+“ and „-“ (e.g. 5EUR – 2.5EUR)
- Exchanging the currency (e.g. EUR → SFR)

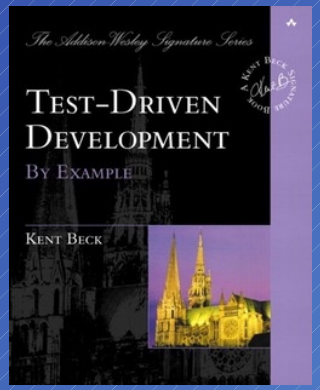
References

Test Driven Development: By Example

by Kent Beck

Released November 2002

ISBN: 9780321146533



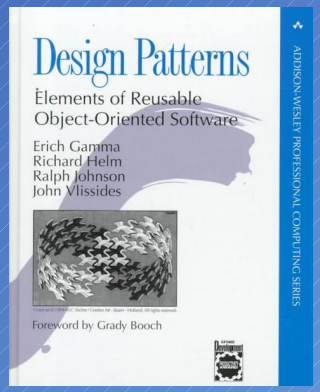
Design Patterns

Elements of Reusable Object-Oriented Software

by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides

Publisher: Prentice Hall

ISBN: 978-0-201-63361-0



Communities

https://www.softwerkskammer.org