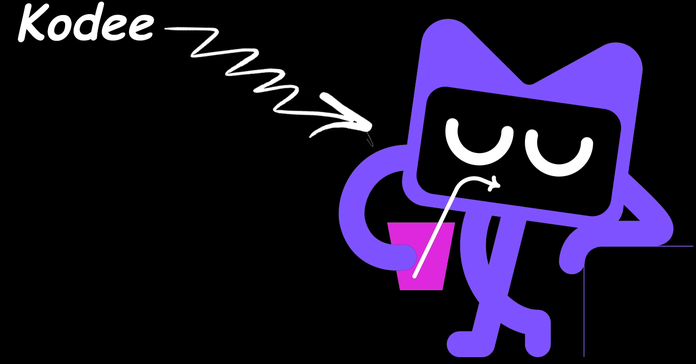
TDD for Kotlin (and beginners)

A brief crash course

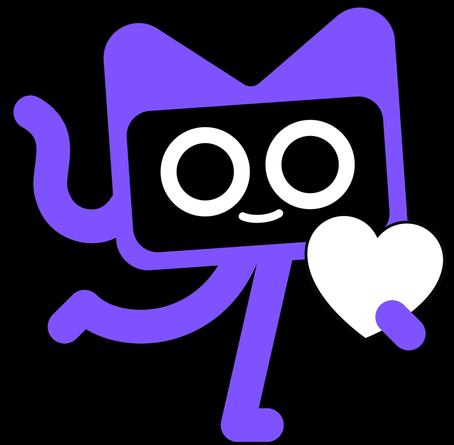
whoami

- Android engineer by day @~~LichtBlick SE~~
- KMP believer by night
- Rust dilettante in my free time
- organizer for KUG & Rust & XTC Berlin & GDG Android Berlin



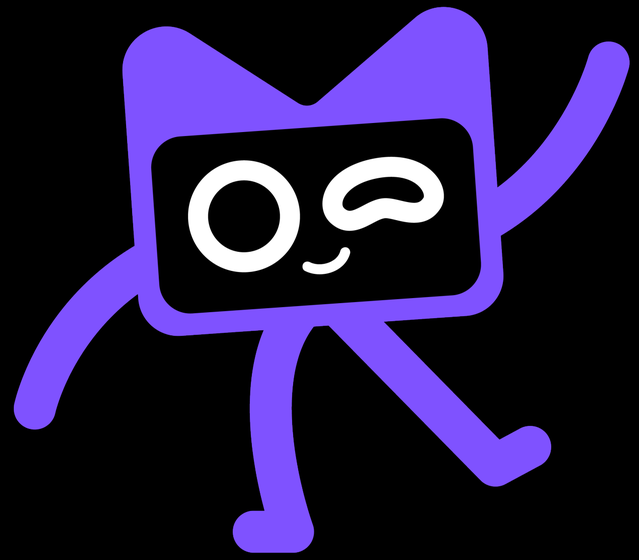
whoami

- Find me on Twitter/GitHub: @BitPogo
- Find me on all the Slacks/LinkedIn: Matthias Geisler
- Join: Kotlin Slack; Software Crafters Slack



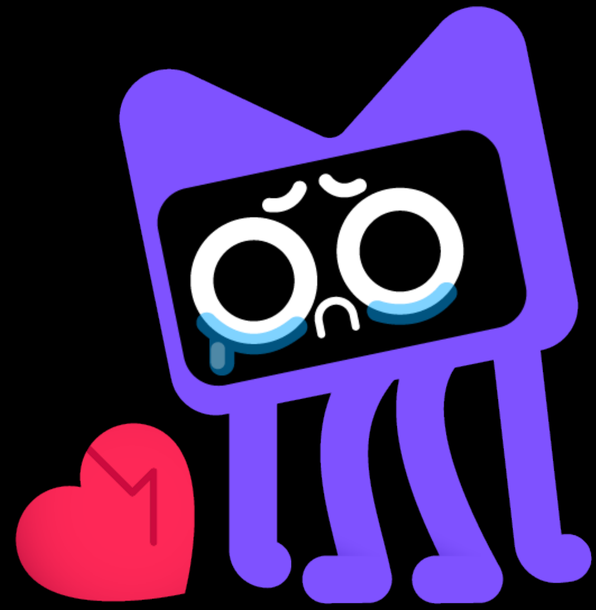
Let’s get to know each other...

1. Why are you here?
2. What you expect from the Workshop?
3. Which technology stack you prefer?



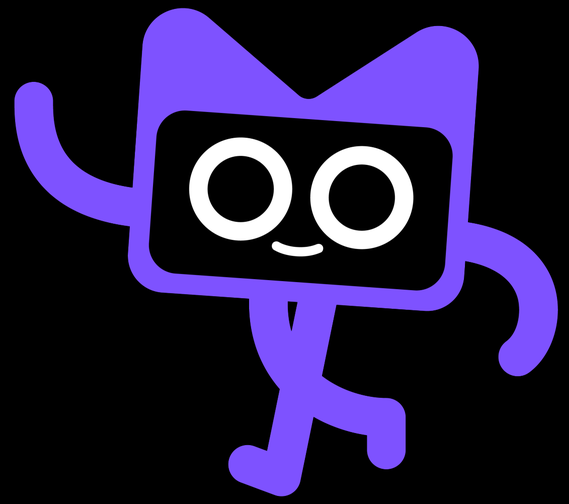
What we will not achieve today...

- A crystal clear picture how to do it
- You will be convinced 100% to do it
- A understanding of the methodologies in depth
- TDD Mastery
- ...



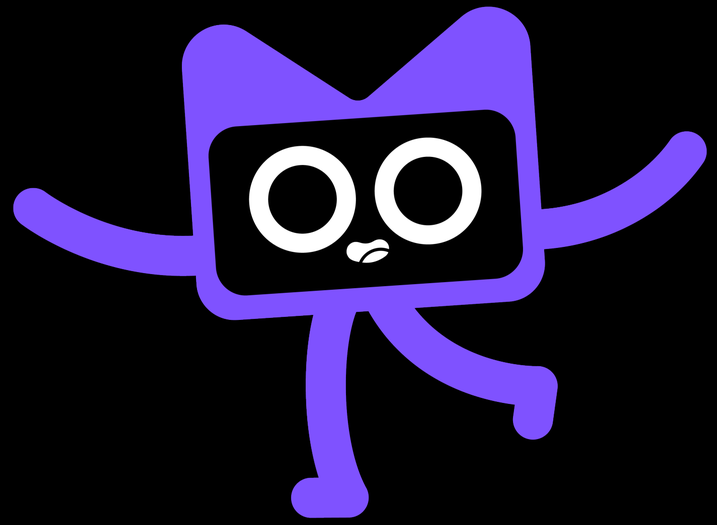
…but we'll manage

- Give you a glimpse what TDD is and how it works
- You know where to go on from here next
- You know whom to ask 😉
- Give you a (different) perspective on TDD
- …



What we are up today?!

1. Prologe ✅
2. A brief intro into TDD ⬅️
3. Let’s make our hands „dirty"
4. Give Feedback

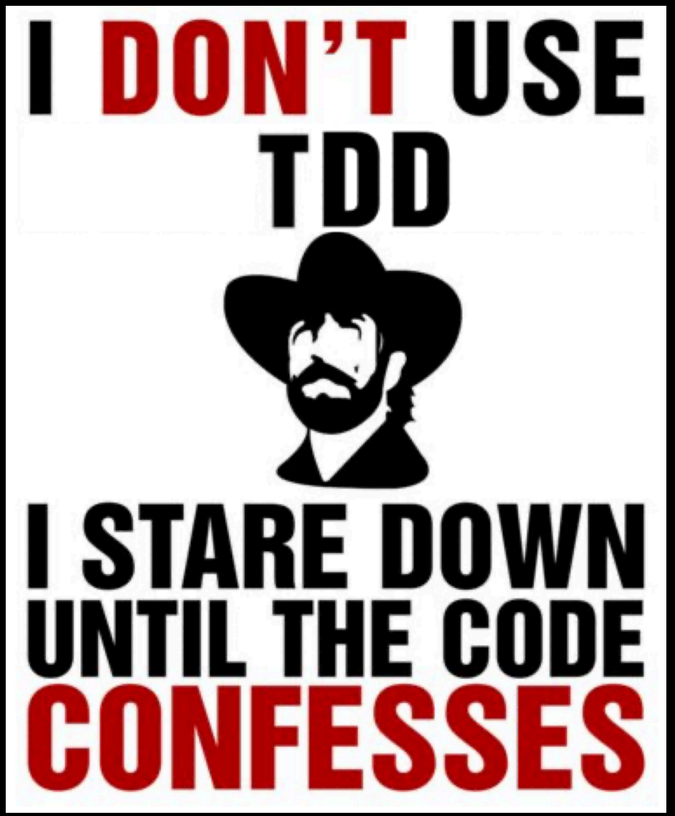


Why we are doing TDD?

THIS PAGE
INTENTIONALLY
LEFT BLANK

Why we are doing TDD?

- FEAR
- Laziness
- Feedback
- Protect us from ourself
- Fun
- Think upfront
- Ethical practice
- …



How we get there?

...conceptually...

Your daily mantra should be:

Write a failing test...

... make the test pass...

... refactor your code!

Writing a failing test...

- You are not allowed to touch any production code before it
- Compiler errors count as a failing test

Why?

This way we make clear what we assume happens (behaviour), what are dependencies...

Make the test past...

- You are allowed to write only enough code to make the test pass

Why?

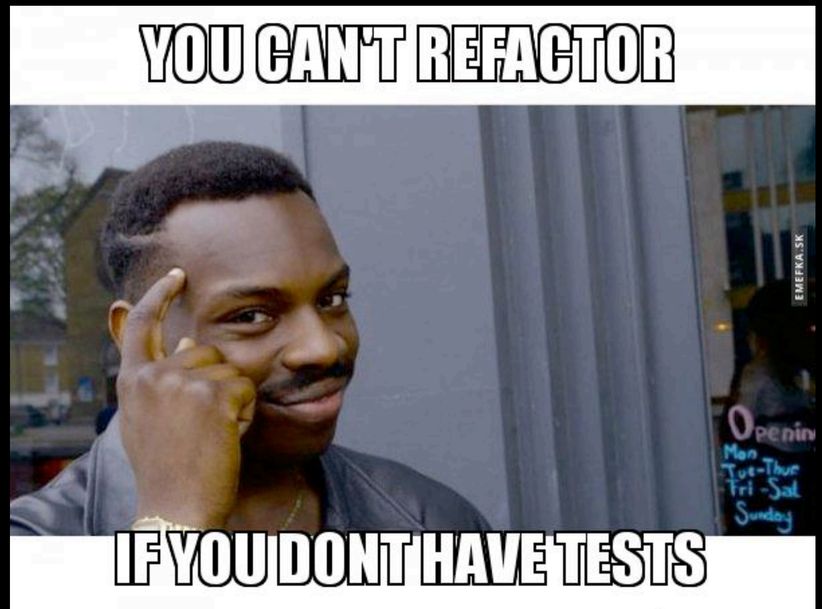
This way we make sure we do not cheat ourself and introduce behaviour what we had not in mind and can produce side effects.

Refactor...

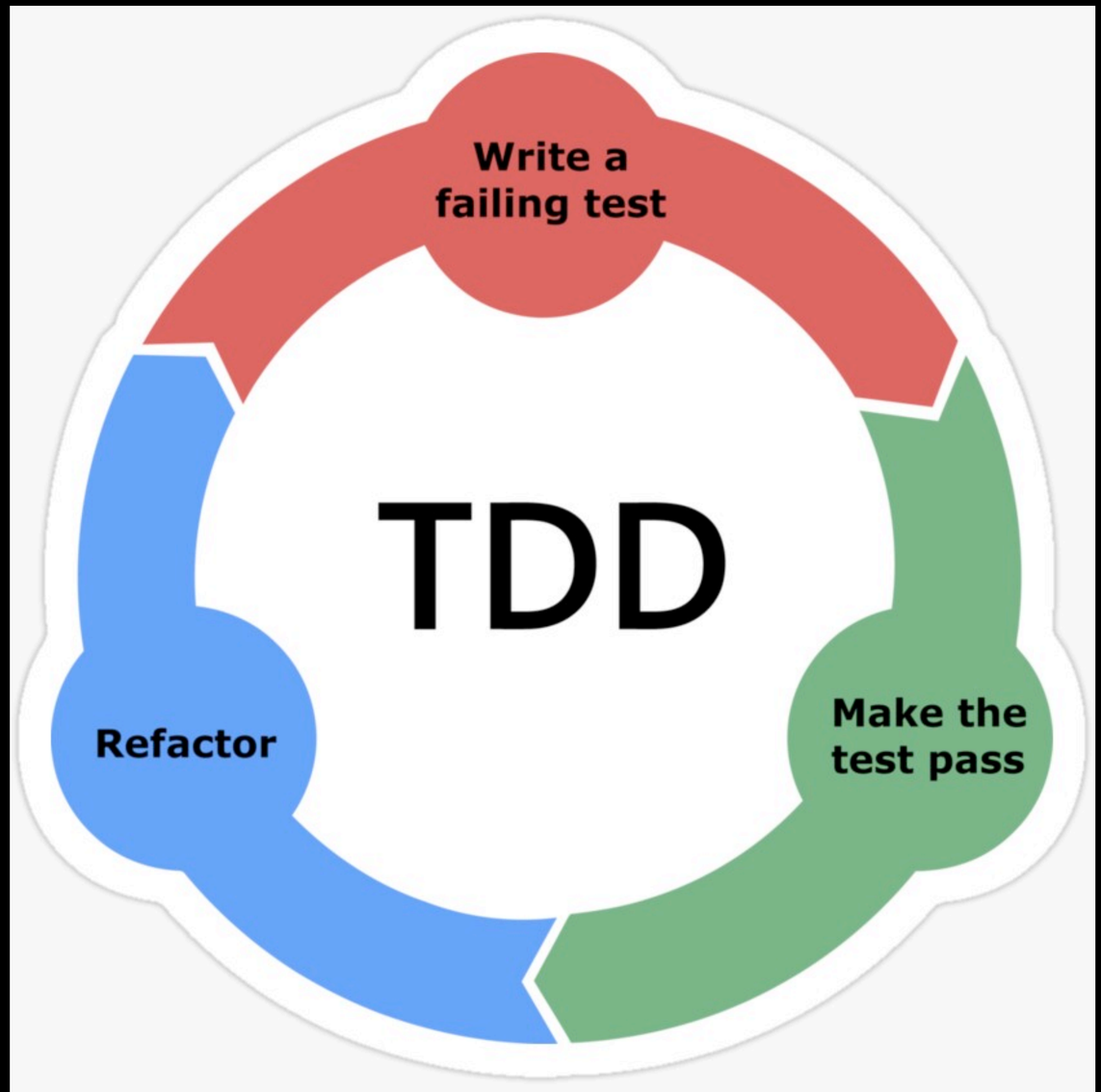
- Restructure your code to make easier to read/understand
- Remove duplications
- ...

Why?

Clean up your mess, before your mess cleans you up.

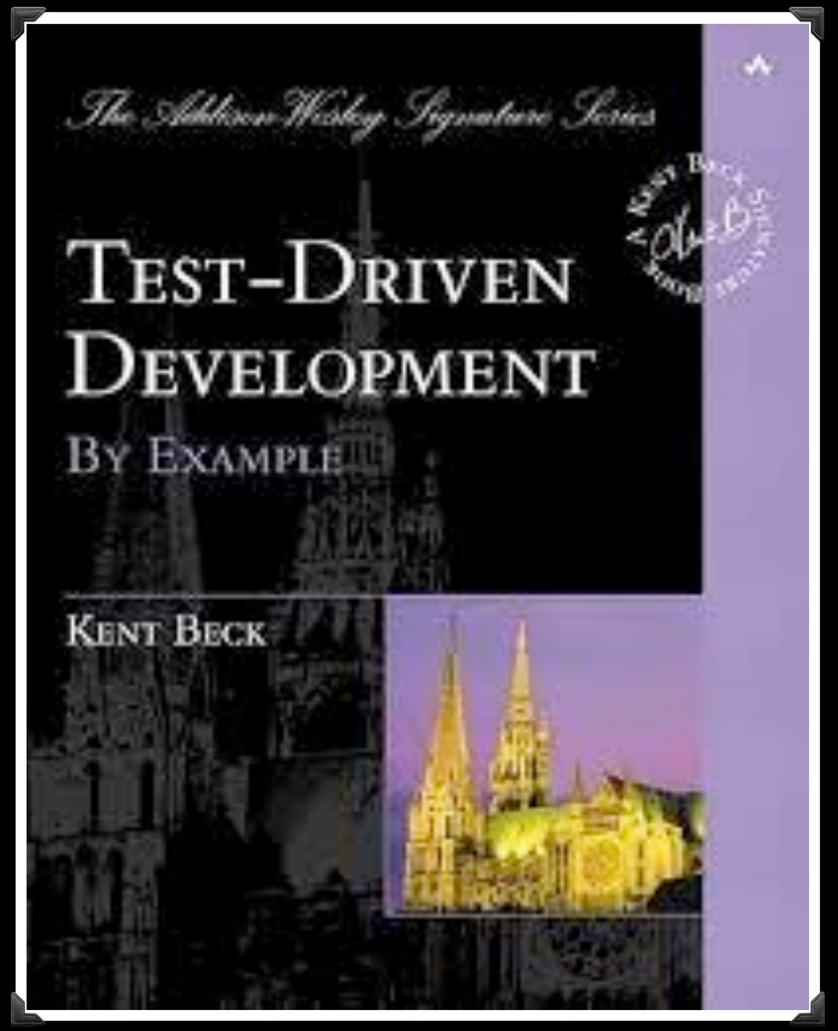


The cycle of life



Is that all?

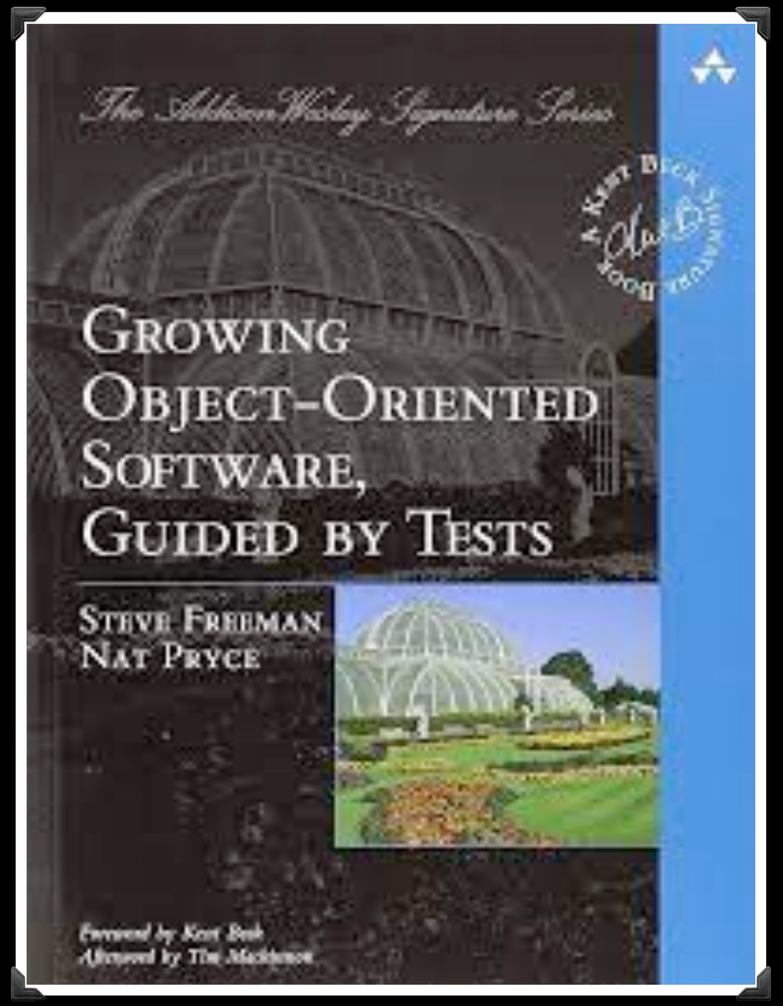
Chicago Style/Classic Style



„TDD is a good for geeks who form emotional attachments to code“

- Mini “Integration Tests“
- Fundamentals
- Fake it until you make it
- …

London Style/Mockist Style



All you want is green

- Acceptance Tests!
- Isolation is Queen
- Mocks (only on interfaces!)
- …

...and?

Both approaches are not mutual exclusive... but you will end up preferring one of them and even mix them up!

Use the approach which gives you/your team the highest confidence

What else?

- You need to learn to listen to the tests, since they constantly telling you something
- You need to be patient with yourself… learning TDD needs time… and practice!
- You need to be disciplined since changing you coding behaviour is a hard!

Little helpers

To practice:

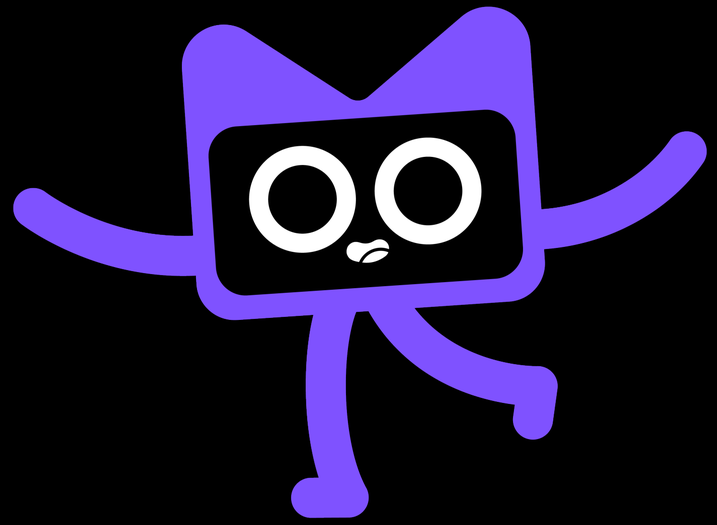
- https://cyber-dojo.org

To read even more:

- Design Patter Book
- Mocks aren't Stubs
- Refactoring Book

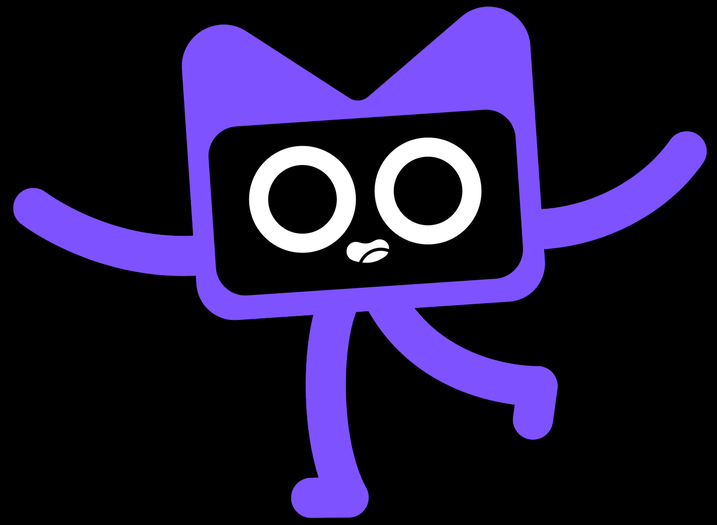
What we are up today?!

1. Prologe ✅
2. A brief intro into TDD ✅
3. Let’s make our hands „dirty"⬅️
4. Give Feedback



What we are up today?!

1. Prologe ✅
2. A brief intro into TDD ✅
3. Let’s make our hands „dirty" ✅
4. Give Feedback⬅️



Please give Feedback

http://tinyurl.com/2nbx5xzk

Done?!

Thank you!!!!!

Questions?