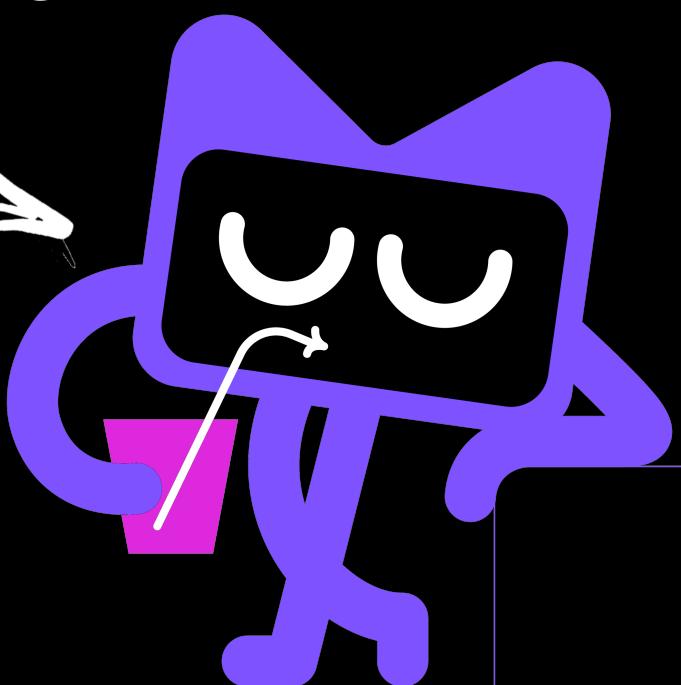


TDD for Kotlin (and beginners)

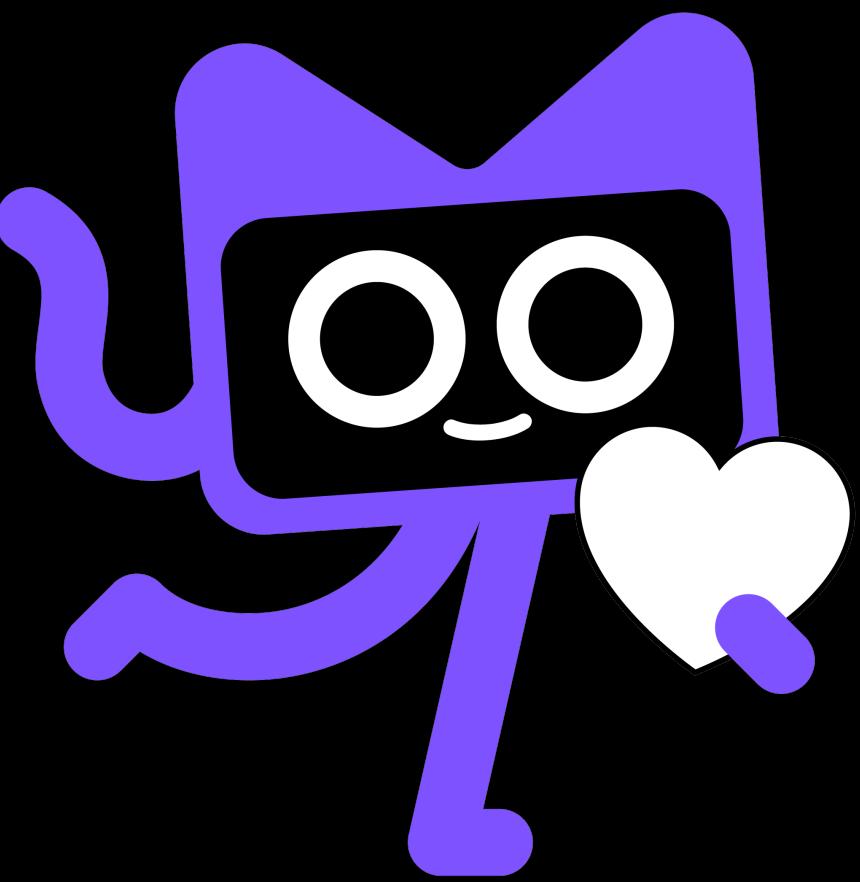
A brief crash course

whoami

- Senior Software Engineer by day @ Airalo
- KMP believer
- Koin Champion (and nobody knows it)
- organizer for KUG & XTC Berlin & XP Unconference & more by night
Kodee



whoami



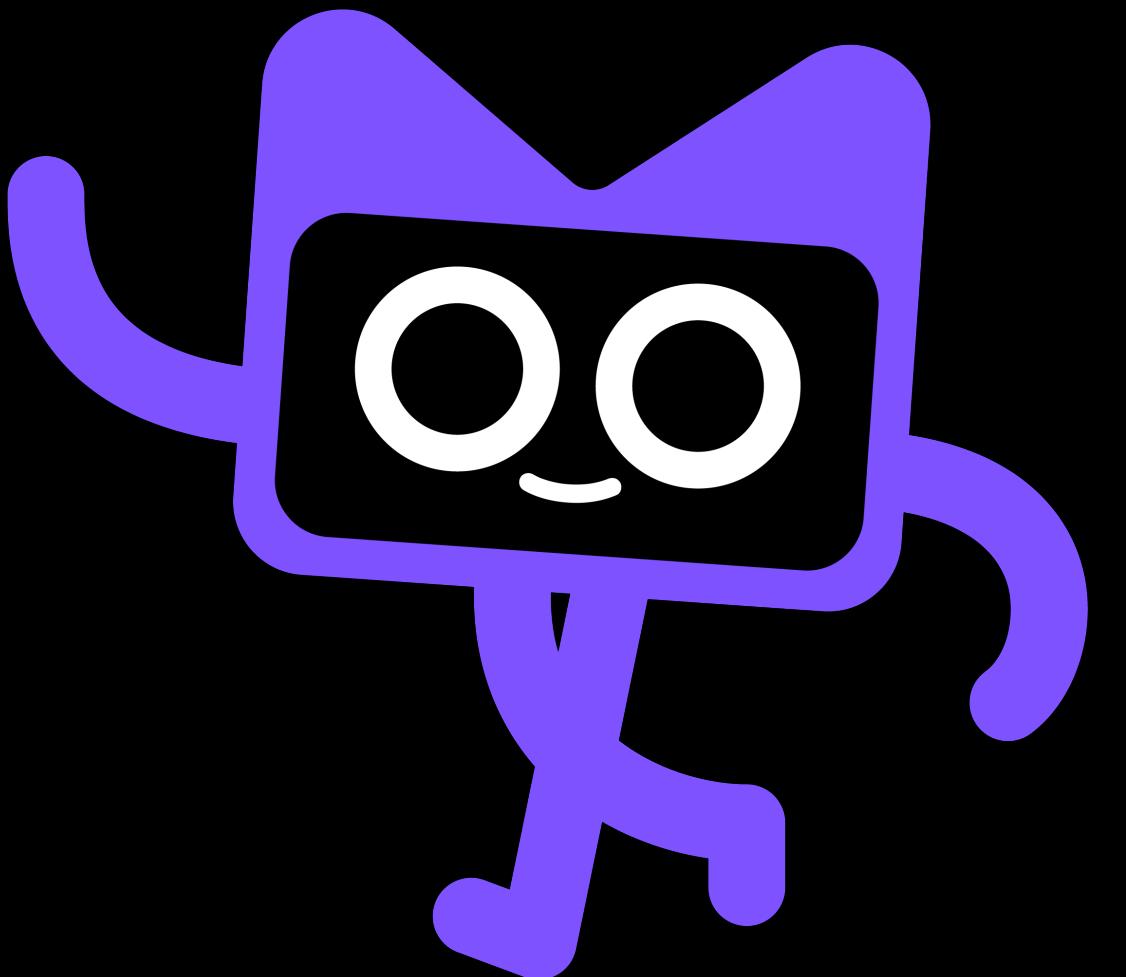
What we will not achieve today...

- 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
- ...



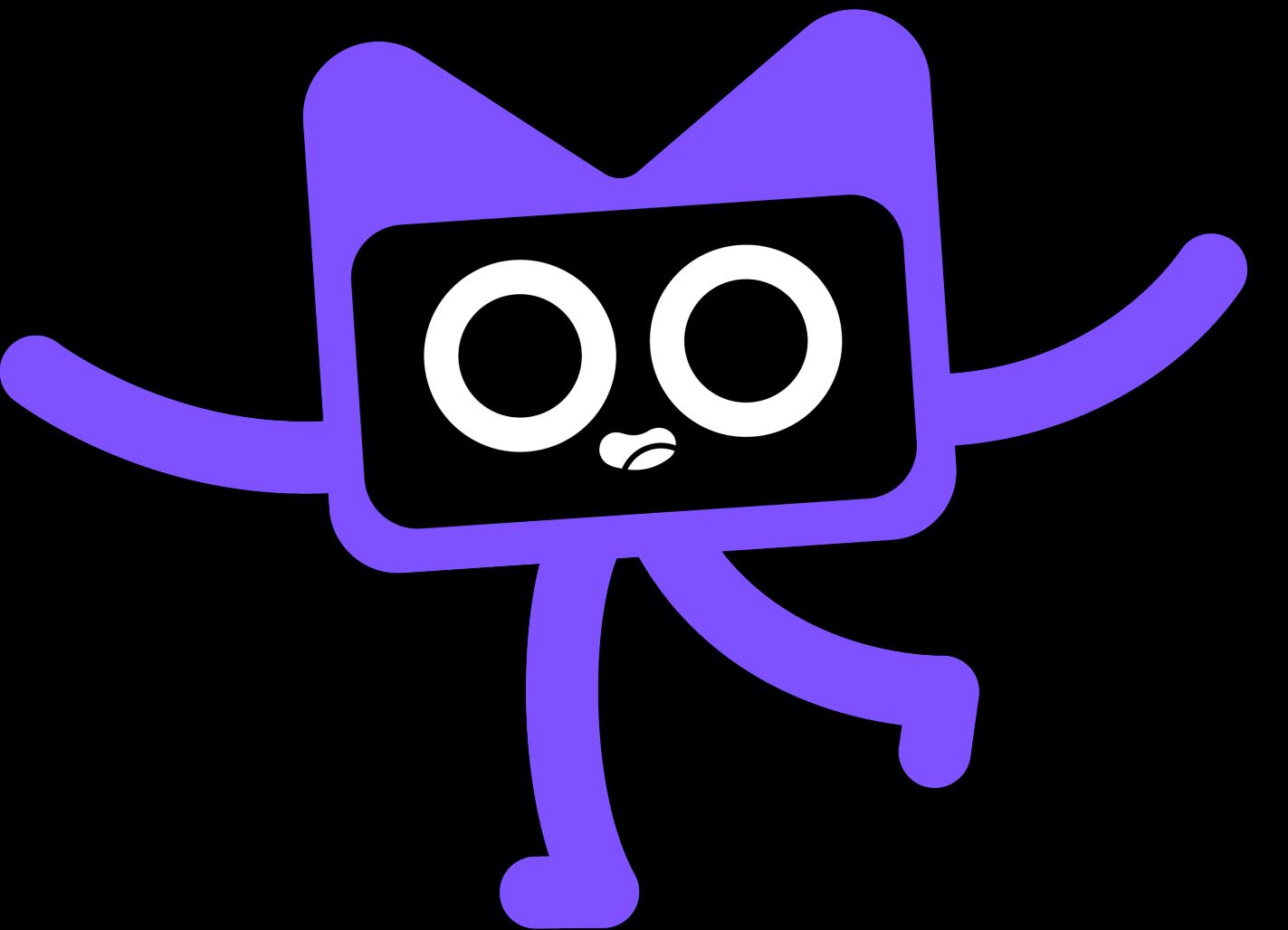
...but we'll manage

- 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. Prologue ✓
2. A brief intro into TDD ←
3. Let's make our hands „dirty”
4. Give Feedback

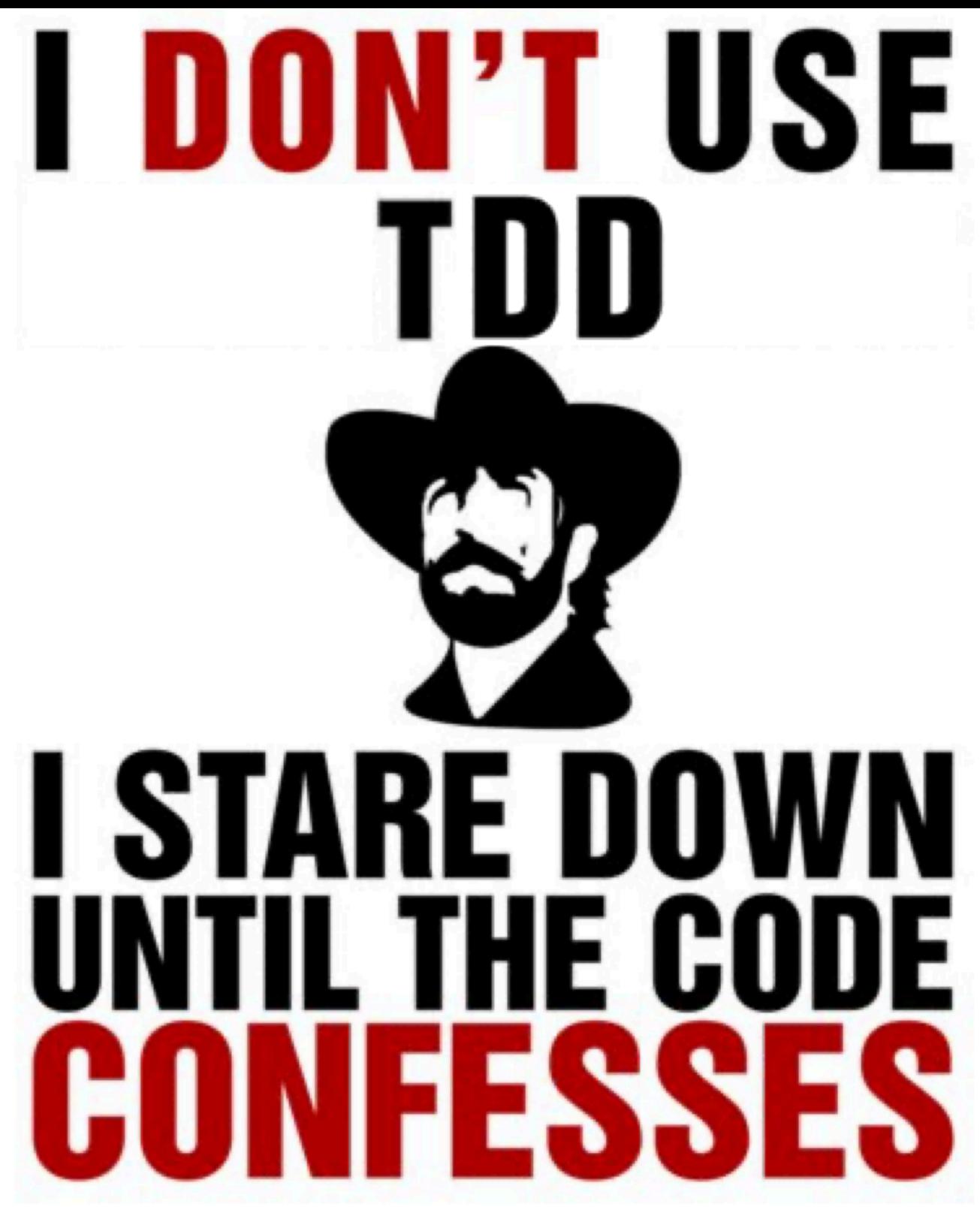


Why we are doing TDD?

THIS PAGE
INTENTIONALLY
LEFT BLANK

Why we are doing TDD?

- FEAR
- Laziness (is not a goal, it is a mindset 🤙)
- Feedback
- Protect us from ourself
- Fun
- Think upfront
- Ethical practice
- ...



„Tests are the Programmer's Stone, transmuting fear into boredom.“

How we get there?

...conceptually...

Your daily mantra should be:

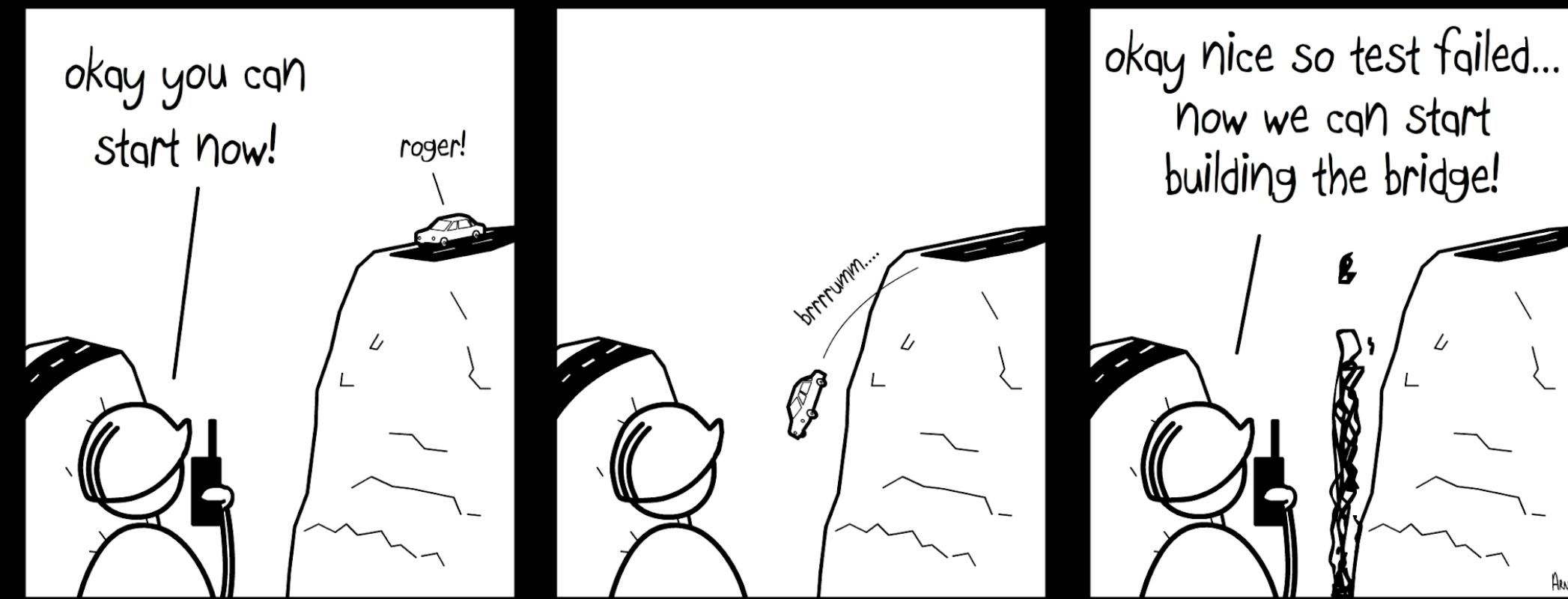
Write a failing test...

... 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 pass...



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

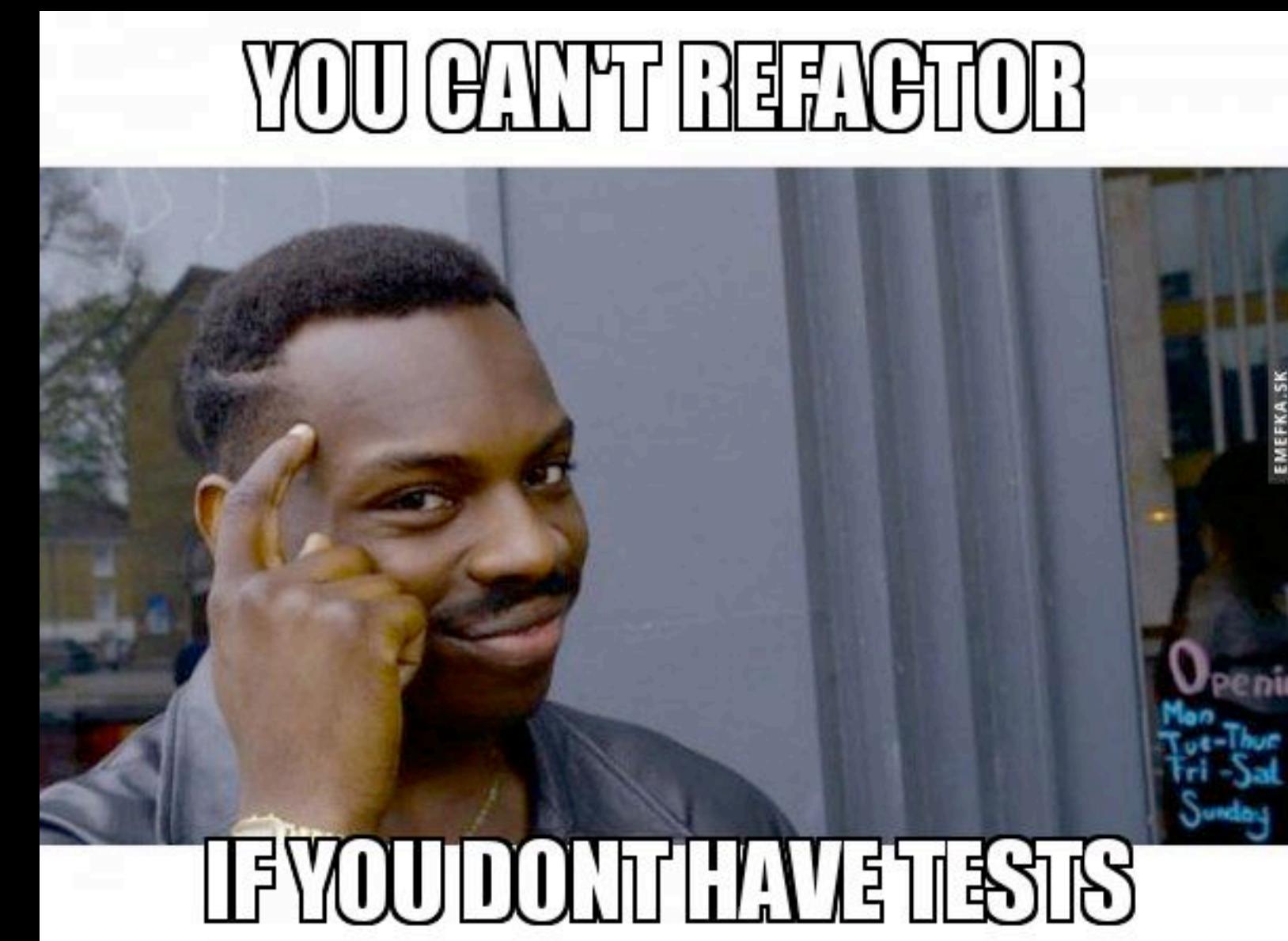
Why?

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

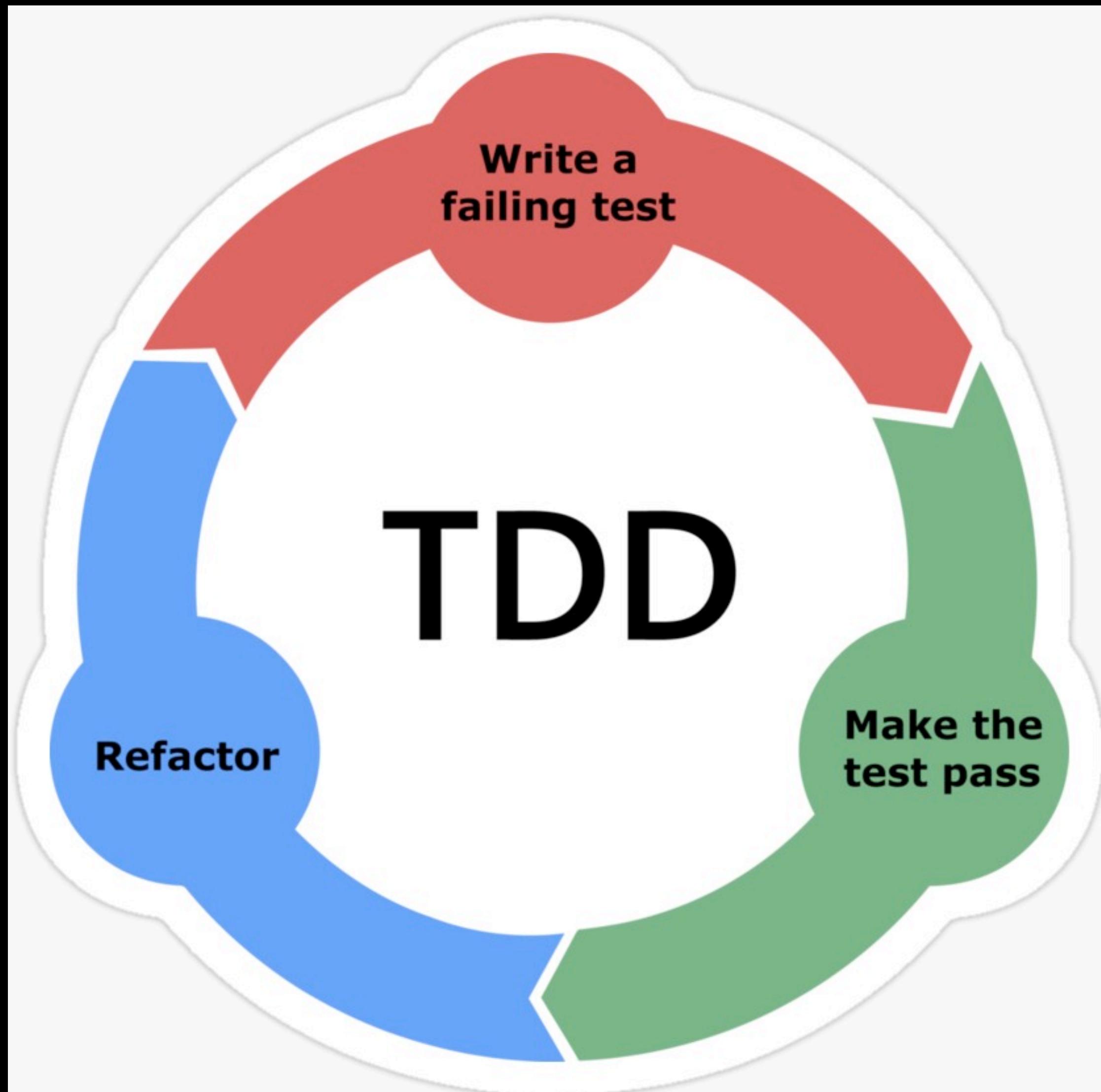
Refactor...

Why?

Clean up your mess, before your mess cleans you up. (Like duplication)



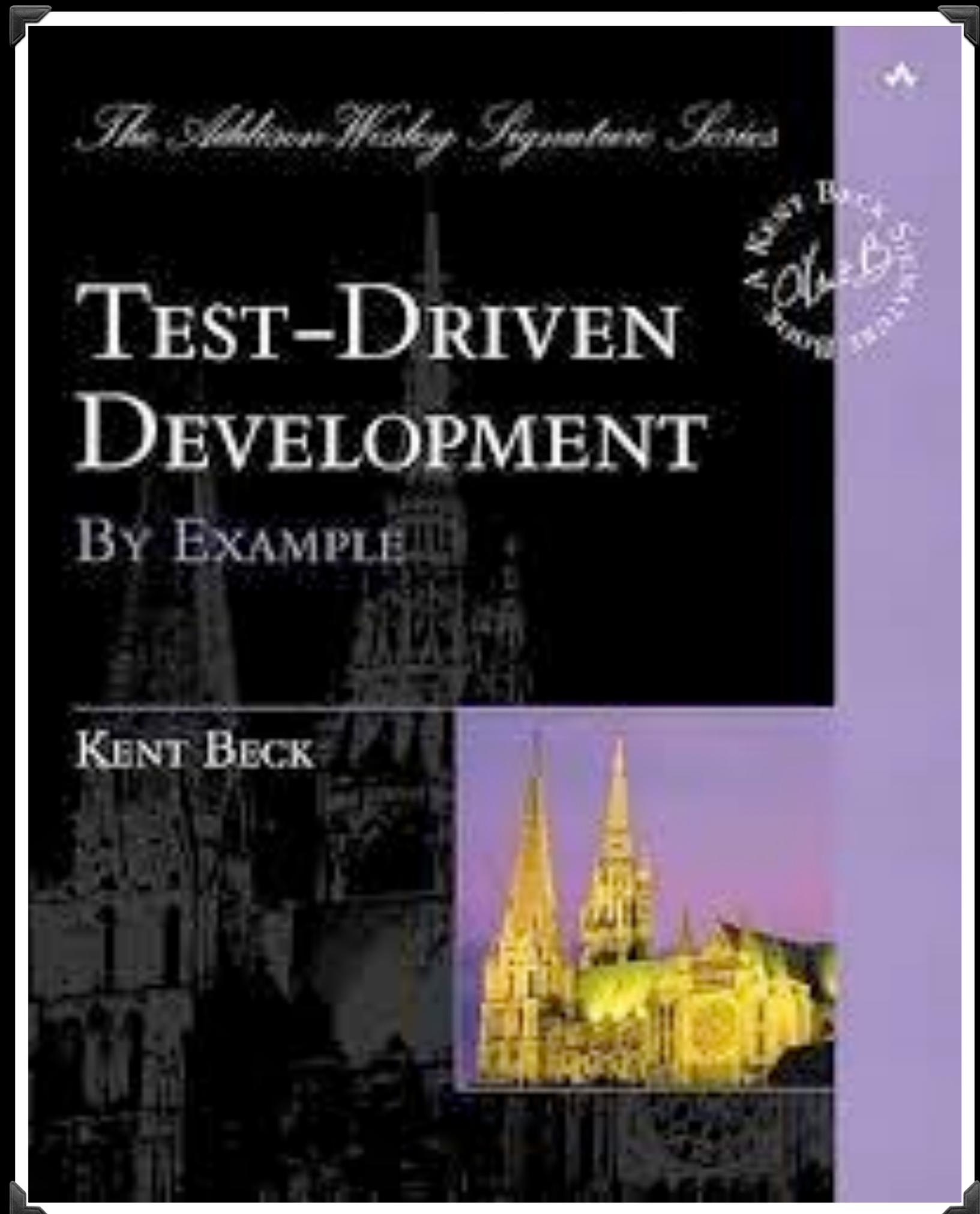
The cycle of life



Is that all?



Chicago Style/Classic Style

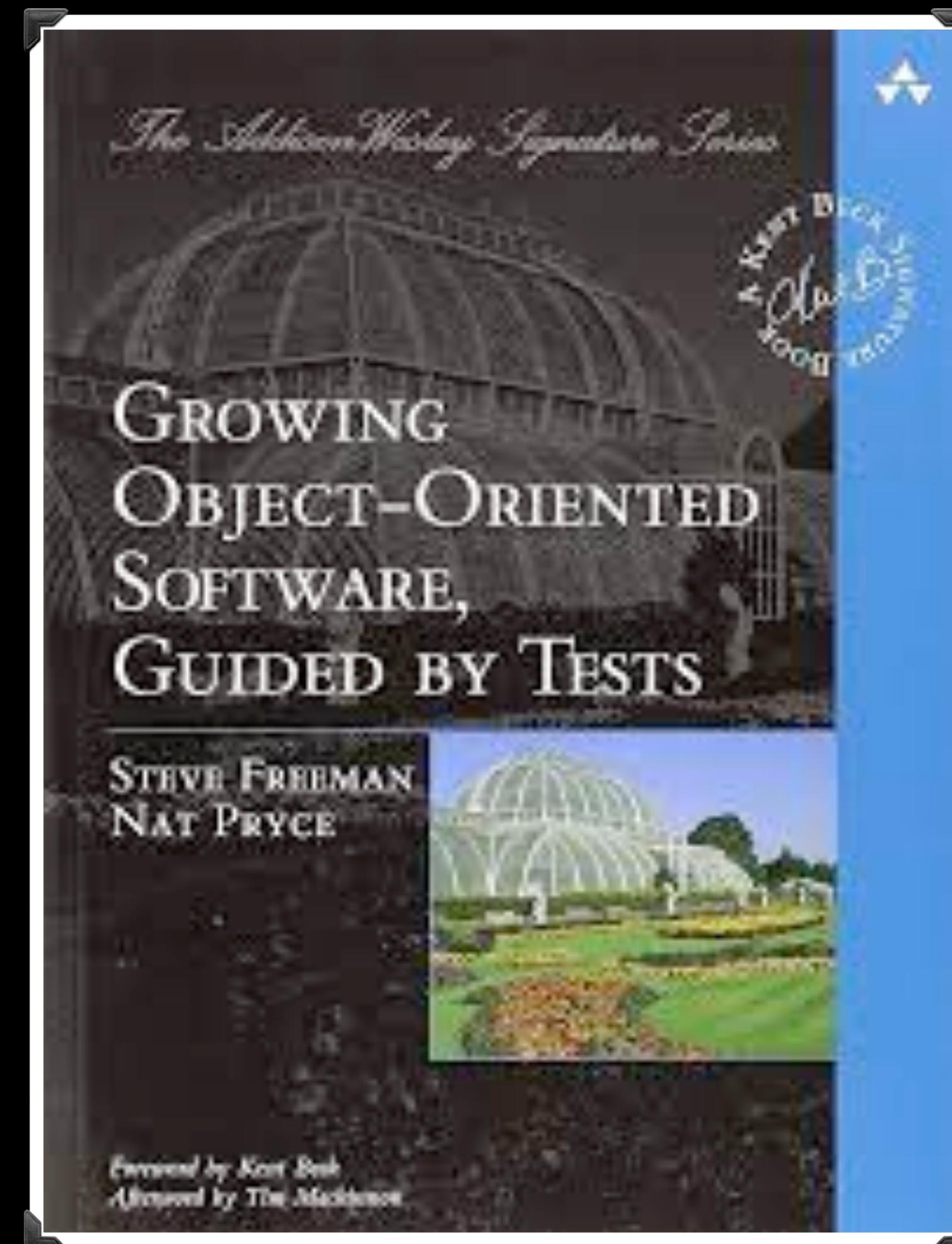


„TDD is a good for geeks who form emotional attachments to code“

- 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 what you own!)
- ...



...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:

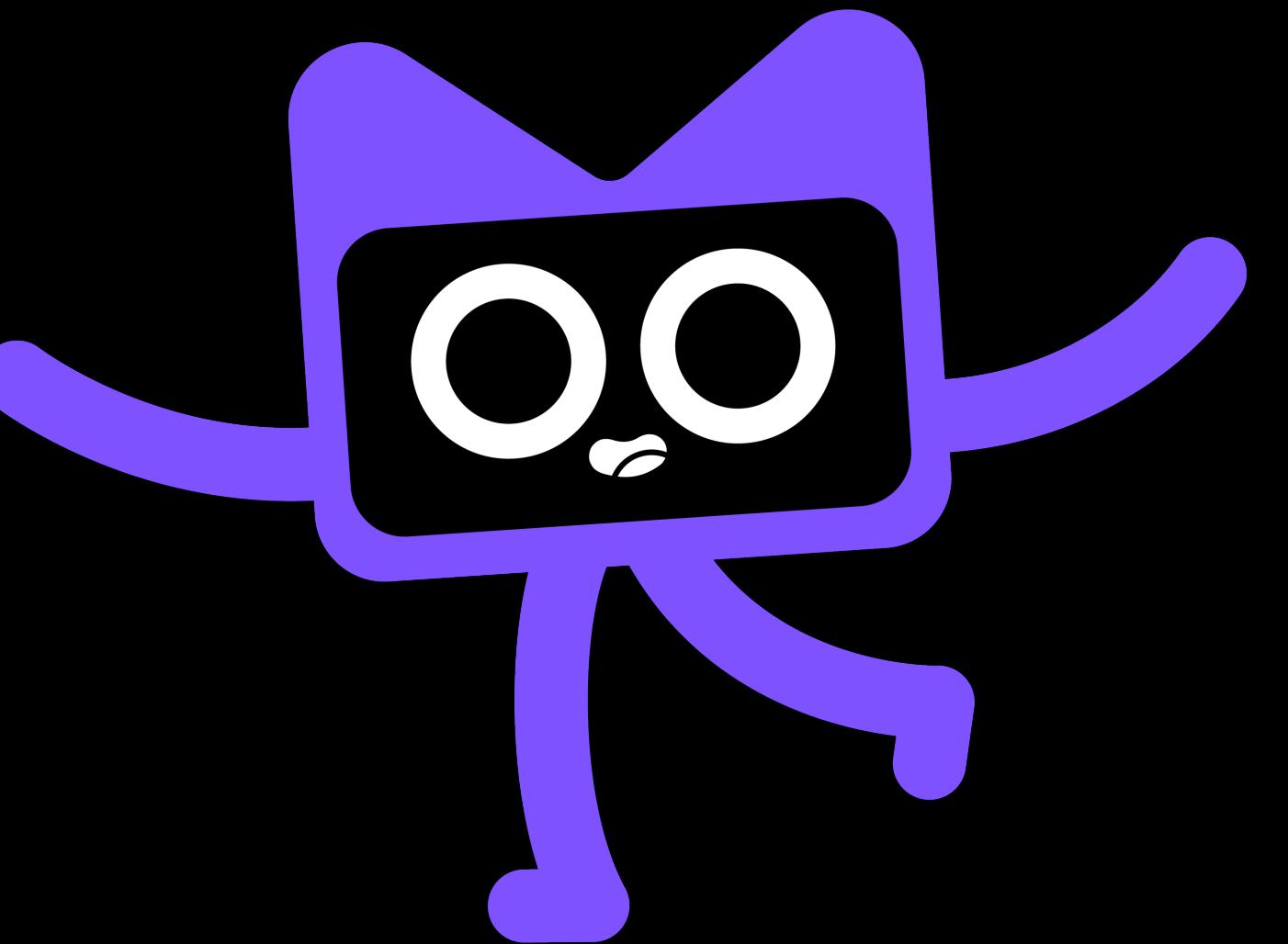
-> [Design Patter Book](#)

-> [Mocks aren't Stubs](#)

-> [Refactoring Book](#)

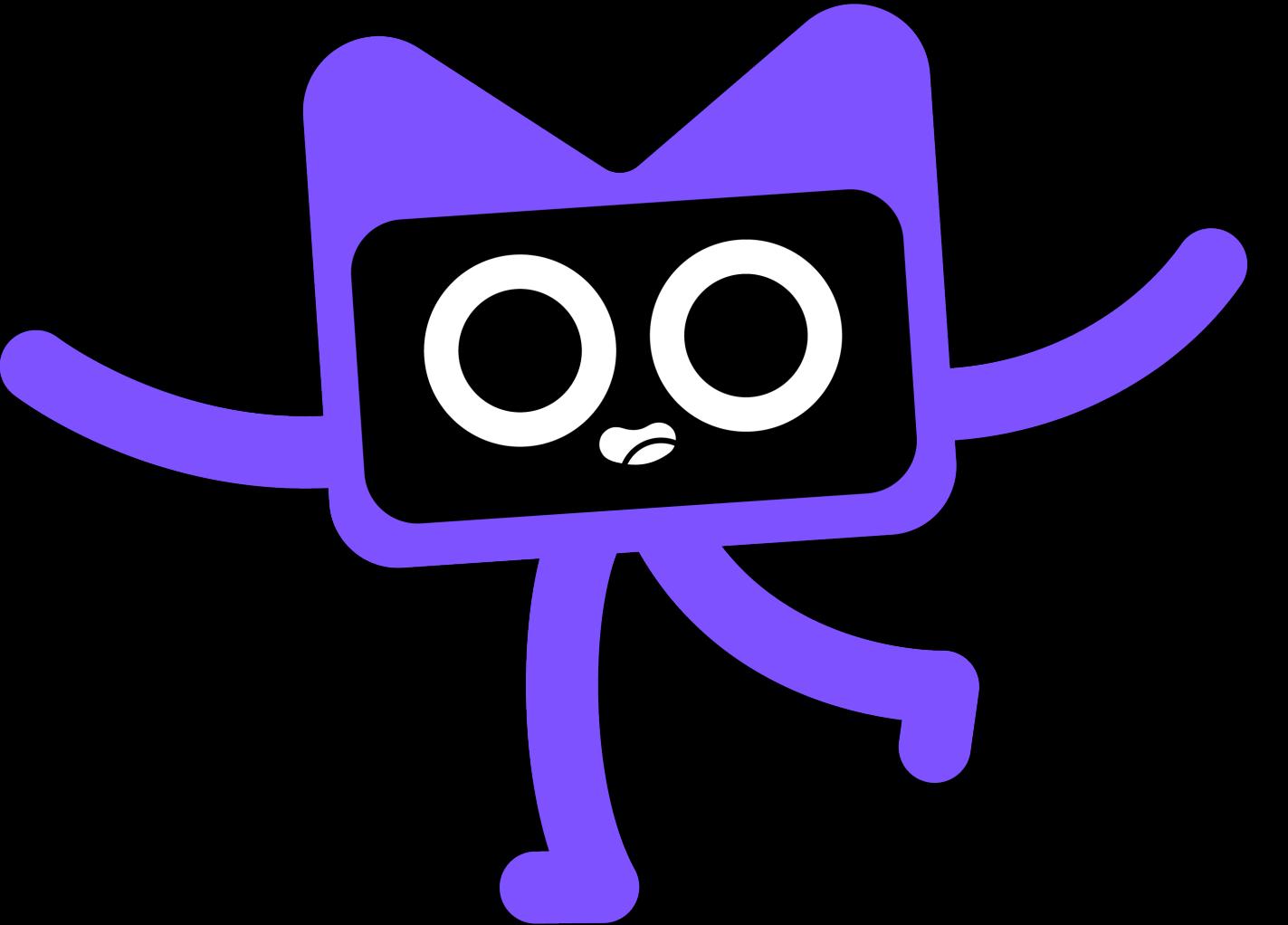
What we are up today?!

1. Prologue ✓
2. A brief intro into TDD ✓
3. Let's make our hands „dirty” ←
4. Give Feedback



What we are up today?!

1. Prologue ✓
2. A brief intro into TDD ✓
3. Let's make our hands „dirty” ✓
4. Give Feedback ←



Please give Feedback



Done?!

