Alcor Academy Foundational Training

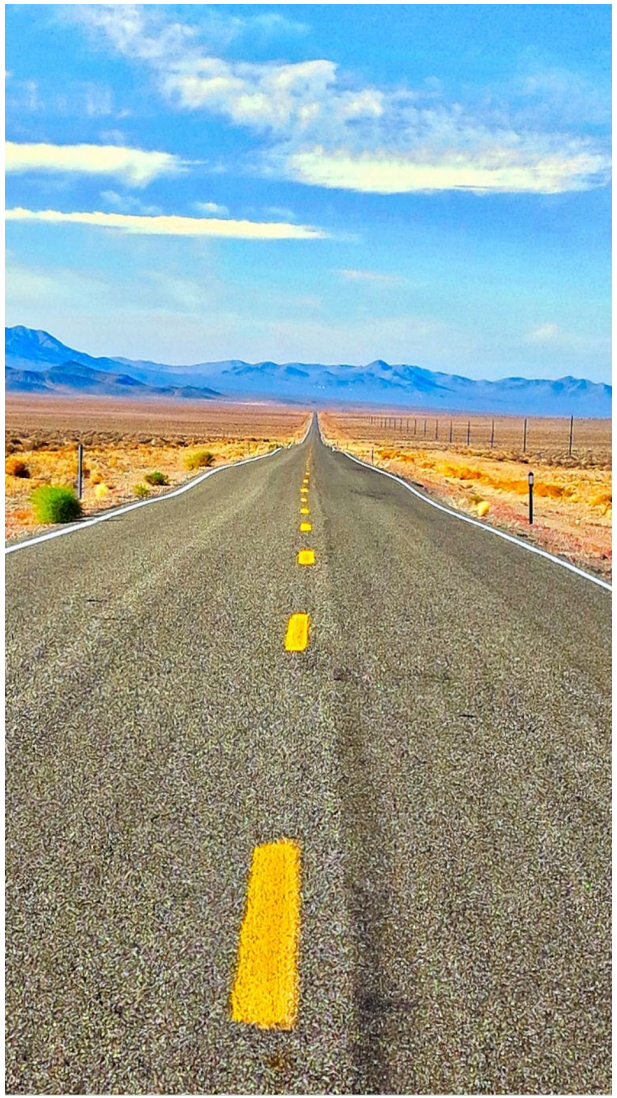
A course retrospective

Ermanno Scanagatta
github.com/ermannos

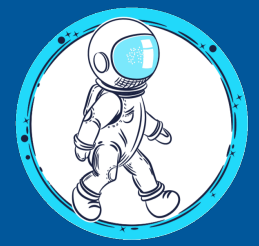
Why this presentation?

The course lasted 9 weeks, full of new concepts, new habits and great experiences.

I'd like to recap the steps and concepts to gain a final perspective view



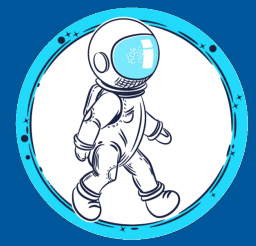
WALKING



Pair programming

- Rules and roles
- Strong type PP
- MOB Programming

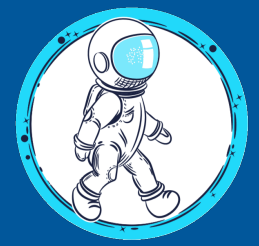
WALKING /2



TDD – how to write new code starting from a test

- Design loosely coupled modules → testable
- What to test → behaviour not implementation
- How to write tests
  - adopt a expressive naming convention (MyClassShould.DoSomethingInteresting)
  - test one behaviour at a time
  - one logical assertion per test
  - tests must be indipendent
  - don't mix state and collaboration
  - baby steps (RED test → GREEN test → REFACTOR)
  - TPP (Fake impl. → Obvious impl. → Trangulation)
  - calisthenics

About WALKING



WOW!

- ✓ I really enjoyed MOB programming, collaboration was very effective
- ✓ TDD, as experienced in Katas, seems to be a quick and useful way to grow a module

HOW ABOUT?

- ✓ At this point it was difficult for me to understand how to apply this kind of TDD in real world situations

RUNNING

Focus is on Refactoring

- When? → Rule of 3, Calisthenics broken
- What? → Maximize value with Pareto principle (readability before design)
- How? → Stay in GREEN, use IDE
- Parallel Change → expand/migrate/contract

RUNNING /2

Bad code indicators are usefult to find what to refactor first

- Code Smells
- SOLID++ principles
- Coupling and Cohesion

About RUNNING

WOW!

- Learn to effectively do reafctoring was great, we will do a lot of it
- Clear taxonomies help to have a common language and to agree a solution with the mob

HOW ABOUT?

- While I easily memorize concepts, for me is difficult to keep all the names in mind

FLYING

- **Connascence:** another way to measure entropy of the system (and to guide refactoring)
- **Test Doubles:** how to replace parts of the module to test it (Stubs & Mocks)
- **4 rules of simple design** (Correct, expressive, no repetittions, least amount of modules)
- **Onion/Hexagonal Architecture** (layers isolate concerns and helps keeping entropy low)
- **Outside-in**
- **Acceptance test** (ATDD)

About FLYING

WOW!

- ATDD Kata revealed all the beauty in the final lesson and dispelled my previous doubts

HOW ABOUT?

- I should learn Mockito

The end

Thanks Marco & Alessandro, great job!

Questions?