Clean Code Fundamentals

TDD

Pre-work

- Video: https://cleancoders.com/episode/clean-code-episode-6-p1
- Exam: https://cleancoders.com/episode/clean-code-episode-6-p1/exam

Chapters

Chapter	Time
Overview	00:53
Fear & Code Rot	04:47
The Big Clean Up	07:02
Eliminating Fear	10:34
Demonstration	14:16
The Real World	31:38
The Three Laws of TDD	33:06

Chapter	Time
Debugging Time	37:45
Design documents	41:24
Decoupling	43:15
Courage to Change	44:53
Trust	48:42
Conclusion	50:35

Timetable

Activity	Time
Warmup	5 min
Exercise 1	20 min
Exercise 2	20 min
Exercise 3	20 min
Wrap up	5 min

Warmup

- What tools and frameworks do you use for testing?
    - Type in the meeting chat

Exercise 1

- Prompt
  - What kind of changes to the code would you do if you had a comprehensive set of unit tests?
- Time limit: 10 minutes

Possible answers

- Use better names according to their scope to communicate intent
- Extract explanatory variables to improve readability
- Extract methods and classes to reduce duplication
- Remove unused code to reduce complexity
- Format code to improve readability
- Optimize slow code to improve performance

Exercise 2

- Prompt
  - List and categorize different kinds of software testing
  - Which ones are compatible with TDD?
- Time limit: 10 minutes

The testing pyramid

- Unit tests (bottom)
  - Test smallest possible unit of code
  - Focused on testing individual parts of the system
  - Fast
- Integration tests (middle)
  - Test how different parts of the system work together
  - Focused on testing the interactions between different parts of the system
  - Slower
- End-to-end tests (top)
  - Test the system from the user's perspective
  - Focused on testing the system as a whole
  - Slowest

Exercise 3

- Three Laws of TDD
  - Write *NO* production code except to pass a failing test
  - Write only *enough* of a test to demonstrate a failure
  - Write only *enough* production code to pass the test
- Prompt
  - What objections do *you* have to TDD? Discuss objections and see if you can come to an agreement.
- Time limit: 10 minutes

Discussion

- Groups to share their findings

Summary

- TDD is a technique to write code that is easy to change
  - Eliminates fear of change
  - Increases confidence in code
  - Reduces debugging time
  - Helps you create more decoupled code
  - Prevents code rot
- TDD uses three laws
  - Write *NO* production code except to pass a failing test
  - Write only *enough* of a test to demonstrate a failure
  - Write only *enough* production code to pass the test

What is next?

- Next session
  - Continue the topic of Test Driven Development
  - There will the next video to watch - TDD Part 2
  - 2/3 of the session will be coding dojo
  - 1/3 of the session will be discussion
- Expect an e-mail with instructions for upcoming coding dojo

Final words

Always leave the code better than you found it.
– The Software Craftsmanship Rule