JUnit

TESTING

NOSERYOUNG

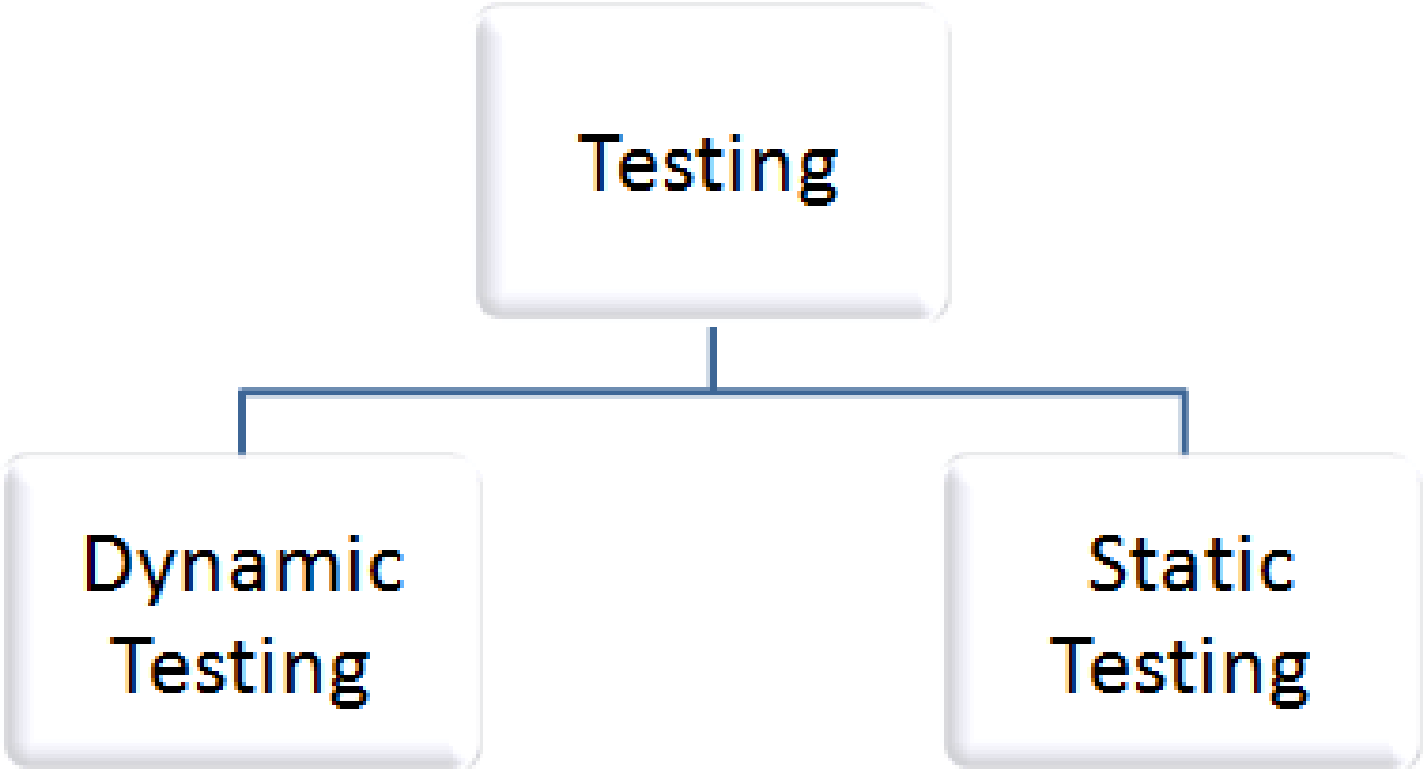
Common Missconceptions

1. Testing is an activity that comes at the end
2. It is performed only by testers not by programmers
3. Its more expensive than usefull

NOSERYOUNG

Software Testing

Dynamic vs Static



Black Box Tests, White Box Tests

Review / Static Analyis

NOSERYOUNG

Static Testing

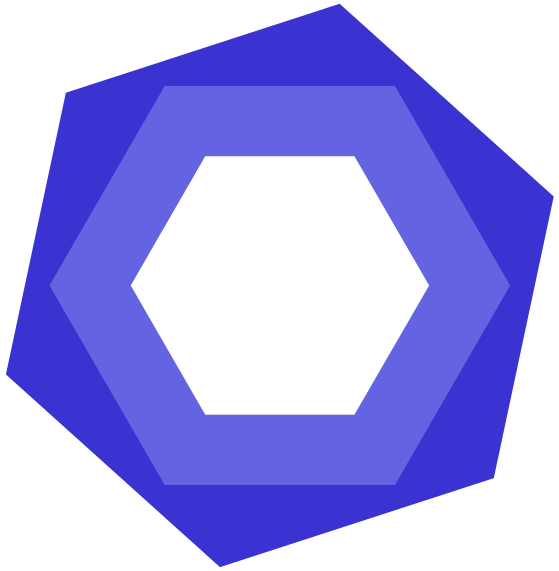
- Static tests are performed on the source code, **without executing it**
- It tests the codes **form, structure, content and documentation**
- **Pros:**
  - Very early error detection,
  - Detects error in code that is not executed.
- **Cons:**
  - Cannot test complex interactions or problems that occur during runtime
- Example: Linters, Review

NOSERYOUNG

Static Software Testing

Linters

- A linter is a **static code analysis tool** used to flag programming **errors, bugs, stylistic errors** and suspicious constructs
- Especially **useful for dynamically typed languages** like JavaScript and Python



NOSERYOUNG

Dynamic Testing

- Tests software by **executing it and observing ist effects**
- It tests he codes **function**, as well as **nonunctional** aspects (e.g. performance)
- **Pro:**
  - More «realistic» test
  - Tests the interplay of parts of the system
- **Con**:
  - Expensive to implement
  - Only tests parts of the code that are executed
- Examples: Unit Tests, System Tests

NOSERYOUNG

Dynamic Software Testing

Box-Testing

NOSERYOUNG

White Box Testing

What is tested for?

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis

NOSERYOUNG

White Box Testing

How?

- Understand Source Code!
- Understand Requirenments!
- Create Test cases
  - Code Coverage

- Unit Testing
- Penetration Testing
- Testing for Memory Leaks

NOSERYOUNG

White Box Testing

Pros & Cons

Advantages

- Code optimization by finding hidden errors.
- Can be easily automated.
- Testing is more thorough as most code paths are usually covered.
- Testing can start early in development even if GUI is not available.

Disadvantages

- Quite complex and expensive.
- Requires a detailed understanding of programming and implementation.
- Time-consuming

NOSERYOUNG

Black Box Testing

What is tested?

- **Functional Testing**
  - Does the Software meet the requirements? I.e. are alluser stories implemented & all use cases covered
- **Non-Functional Testing**
  - Performance, scalability, usability etc.
  - Penetration tests (simulate external hacking)
  - Stress testing

NOSERYOUNG

Black Box Testing

How?

- Understand Requirenments!
- Evaluate the set of valid inputs and test scenarios to test the software
- Prepare the test cases to cover a maximum range of inputs.
  - Test Boundary values and invalid cases
- Maunal Testing
- Automated UI Testing to simulate usage scenarios

NOSERYOUNG

Black Box Testing

Pros & Cons

Advantages

- No technical knowledge required
- Testing is performed independent of development.
- Tests the “big picture” functionality of the software.

Disadvantages

- Limited coverage
- Test cases are difficult to design.
- Blind coverage, since the tester cannot target specific code segments or errorprone areas.

NOSERYOUNG

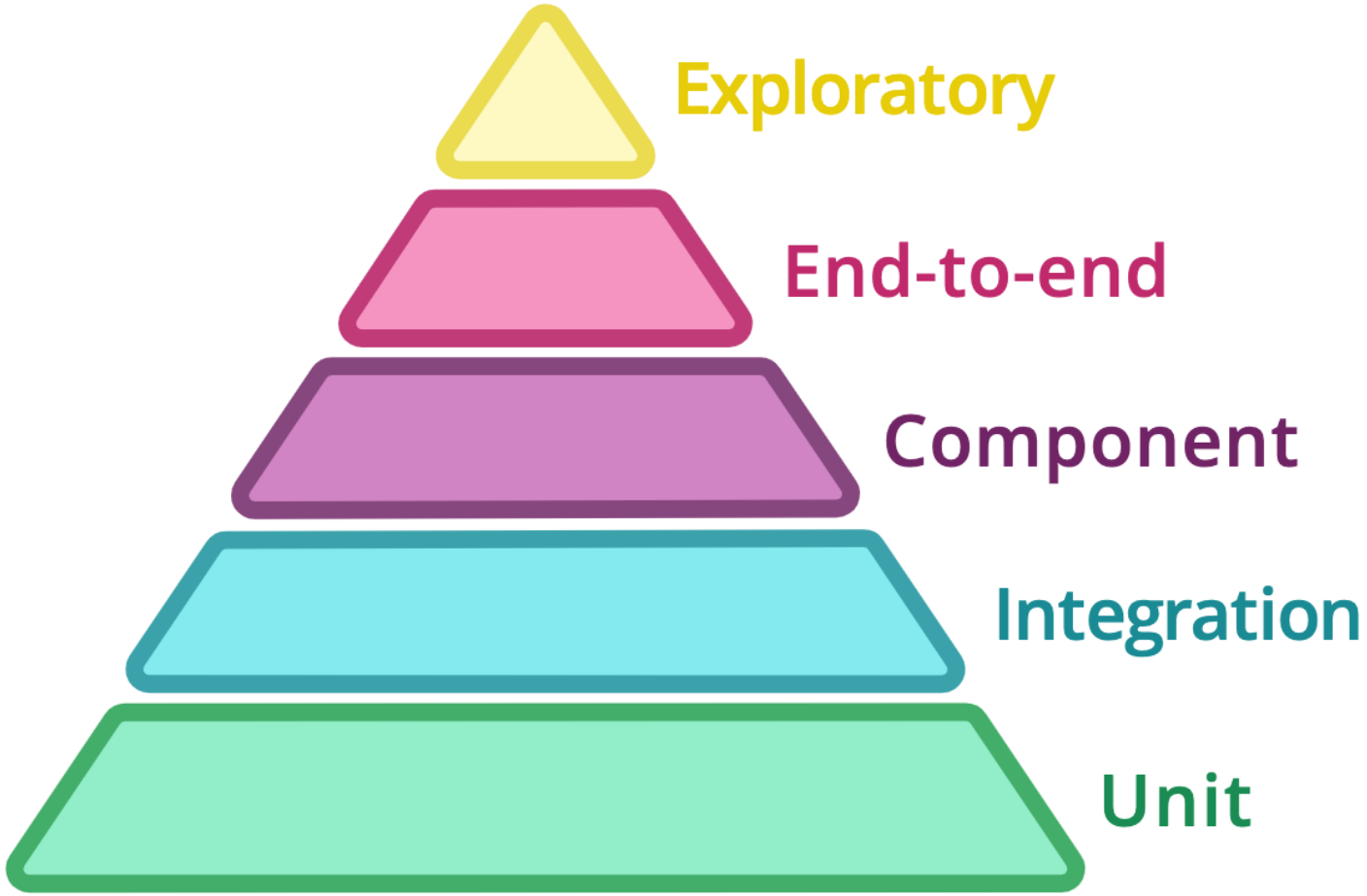
Black Box vs White Box

Zusammenfassung

	Black-box testing	White-box testing
Definition:	Software testing method where the internal structure of the system is not known	Software testing method where the internal structure of the system is known
Used For:	Verifying input methods and outputs of the system.	Verifying internal structure of system's components
Performed By:	Testers	Developers
Applicable To:	Systems and Acceptance testing	Unit testing
Perspective	User	Developer
Introspection	No	Yes
Coding Knowledge:	No	Yes
Implementation Knowledge:	No	Yes
Test Cases:	Based on requirements	Based on detailed design

NOSERYOUNG

Testing Levels

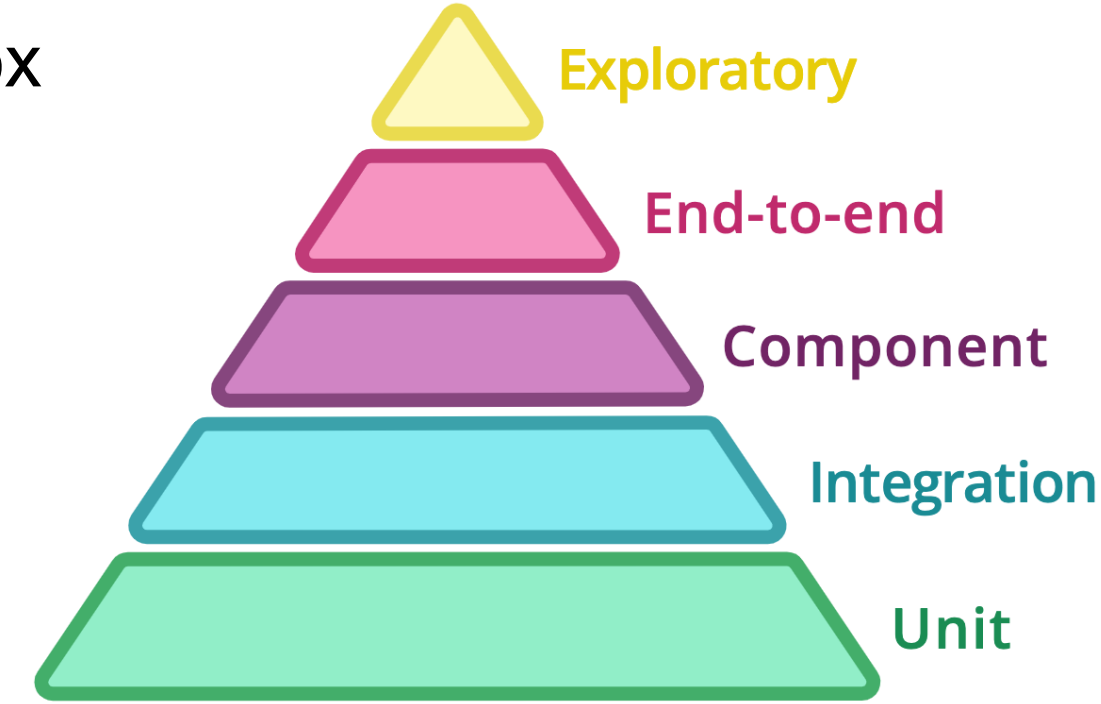


NOSERYOUNG

Testing Levels

Why?

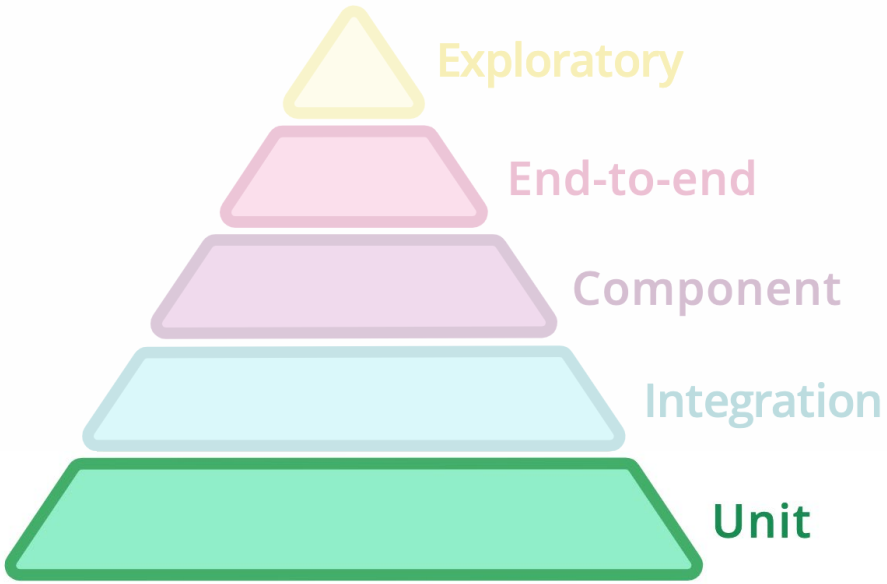
- Differnet levels implement Black Box or White Box Testing
- Differnet levels allow for granular testing of differnet levels of the applicaiton



NOSERYOUNG

Unit Tests

- In Unit Testing **individual units or components of a software** are tested.
- The test validates if a certain piece of code produces the expected output given a specified input



NOSERYOUNG

Unit Tests

Example

- Should cover a large part of the codebase
- Unit tests are fast!

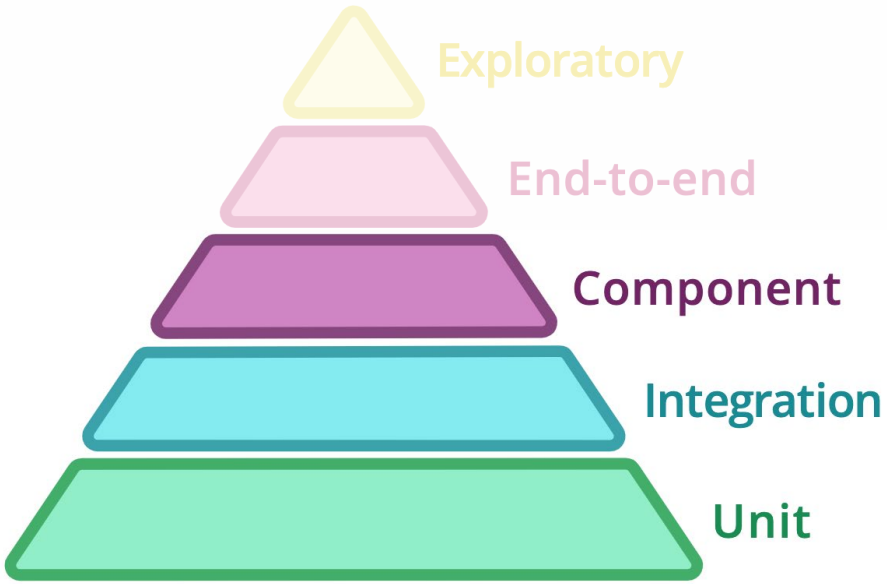
  ➔can be run regularly

```
class CalculatorTest {

    Calculator calculator;

    @Test

    @DisplayName("Simple multiplication")

    void testMultiply() {
        assertEquals(20,
      calculator.multiply(4, 5), "Regular
      multiplication");
    }
}
```

NOSERYOUNG

Integration & Component Tests

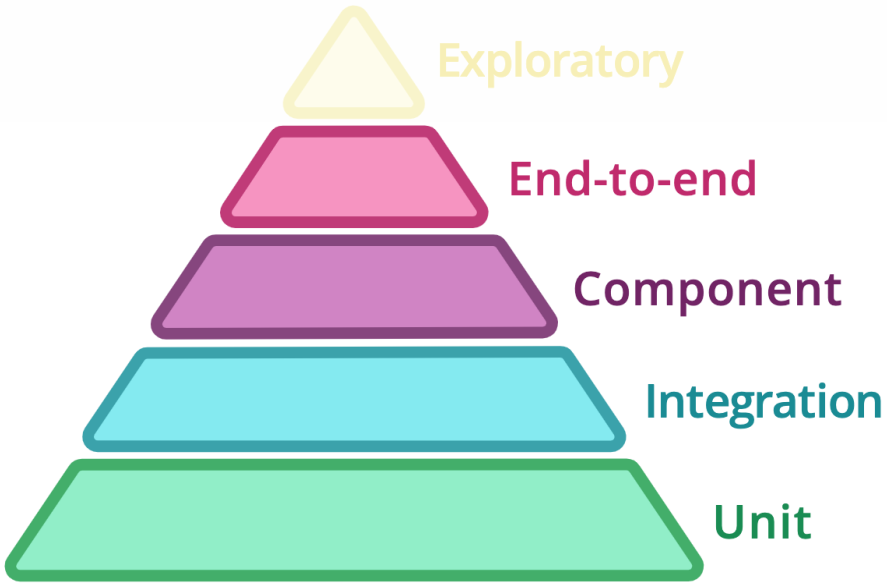
- Test the correct **inter-operation of multiple subsystems**
- Test the modules which are working fine individually don't have issues when they are integrated.
- This detects errors related to the API design



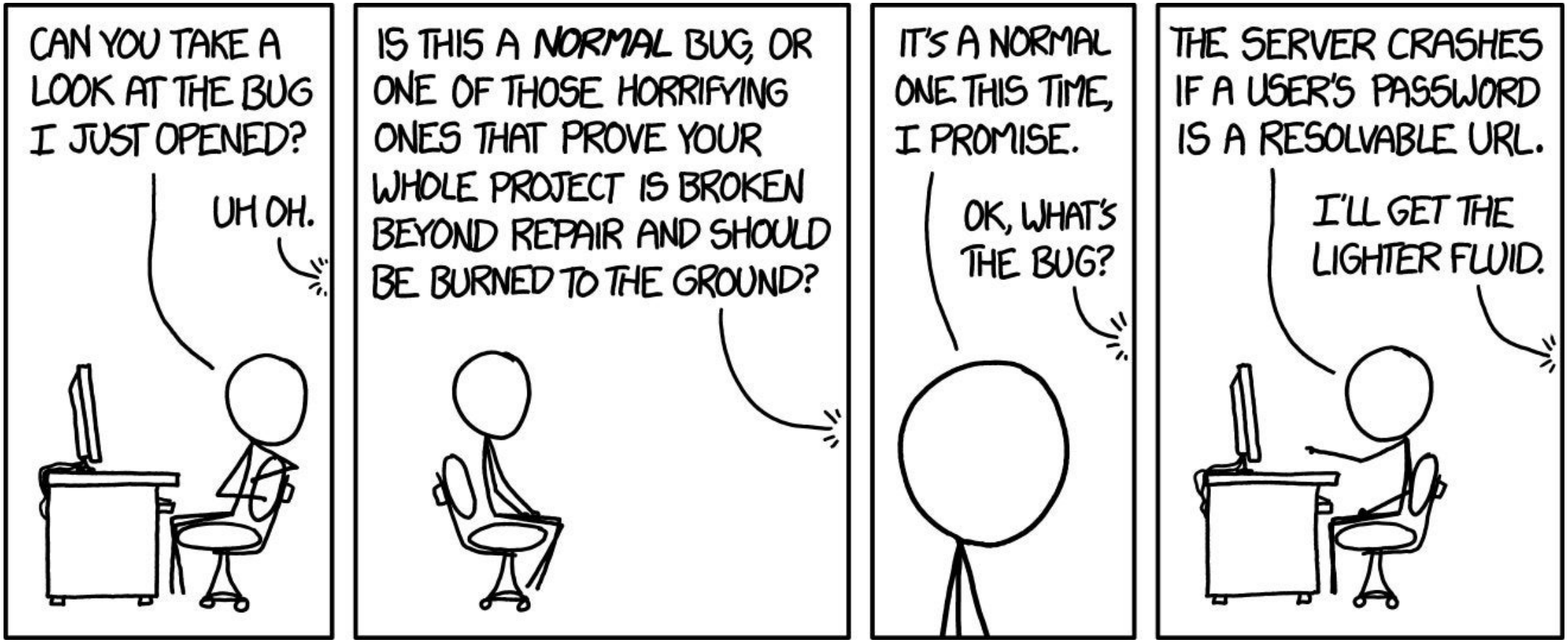
NOSERYOUNG

End-to-end / System Test

- Tests a system as a black box.
- Ensures that both **functional and non-functional requirements** are met
- Aims to replicate real user scenarios so that the system can be validated for integration and data integrity.

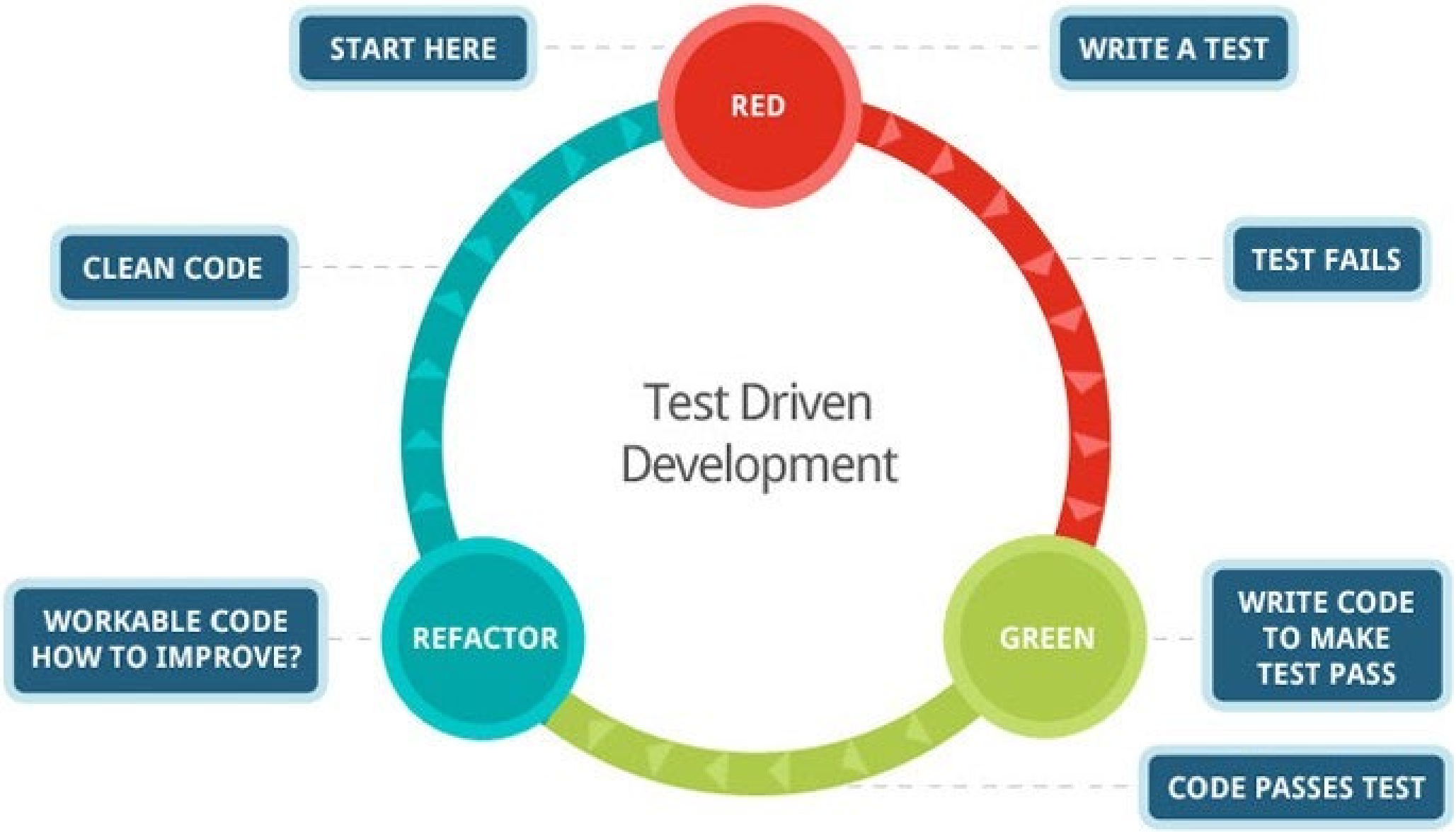


NOSERYOUNG


CAN YOU TAKE A LOOK AT THE BUG I JUST OPENED?
UH OH.
IS THIS A NORMAL BUG, OR ONE OF THOSE HORRIFYING ONES THAT PROVE YOUR WHOLE PROJECT IS BROKEN BEYOND REPAIR AND SHOULD BE BURNED TO THE GROUND?
IT'S A NORMAL ONE THIS TIME, I PROMISE.
OK, WHAT'S THE BUG?
THE SERVER CRASHES IF A USER'S PASSWORD IS A RESOLVABLE URL.
I'LL GET THE LIGHTER FLUID.

NOSERYOUNG

TEST DRIVEN DEVELOPMENT (TDD)


START HERE
RED
WRITE A TEST
CLEAN CODE
TEST FAILS
Test Driven Development
WORKABLE CODE HOW TO IMPROVE?
REFACTOR
GREEN
WRITE CODE TO MAKE TEST PASS
CODE PASSES TEST

Vorteile von TDD

- kein ungetesteter Code
- saubere/testbare Architektur durch TDD als Designstrategie
- keine/wenig Redundanzen durch gnadenloses rechtzeitiges Refactoring
- kein unnötiger Code, welcher nicht gebraucht wird
- Software Entwicklung macht Spass, wenn immer wieder etwas grün wird

Nachteile von TDD

- Hoher Zeitaufwand
- Software nur so gut wie der Test
- Erfordert gewisse Erfahrung in der Software Architektur um sich im Vorfeld vorzustellen, was (welche Methoden) getestet werden soll.

MOCK-OBJEKTE MIT MOCKITO

- Mock = Attrappe / Täuschung