Fundamentals of Java Testing

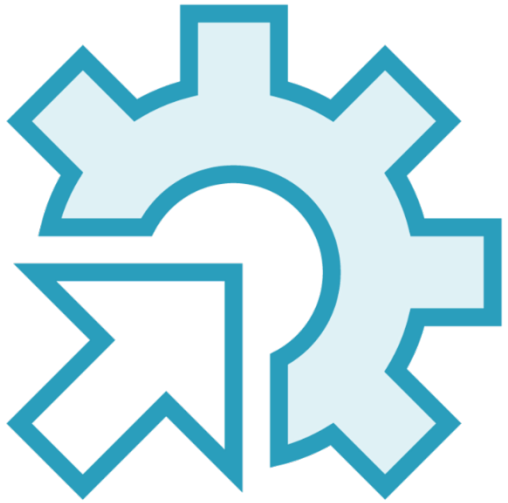
TESTING CODE

Richard Warburton
JAVA CHAMPION, AUTHOR AND PROGRAMMER

@richardwarburto www.monotonic.co.uk

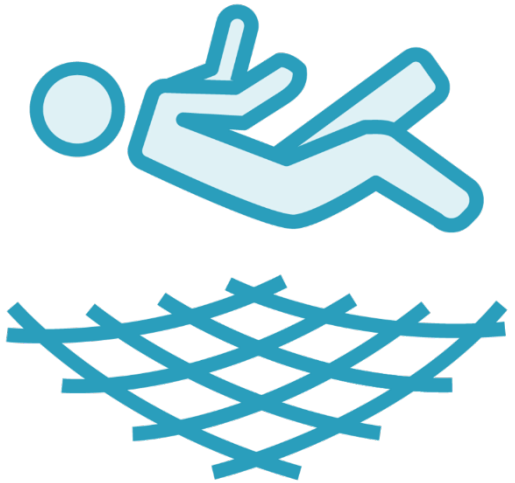
Motivation and Outline

Software Testing



Quality & reliability

Meeting functional requirements



Remove fear of change

Course Overview

Testing Code

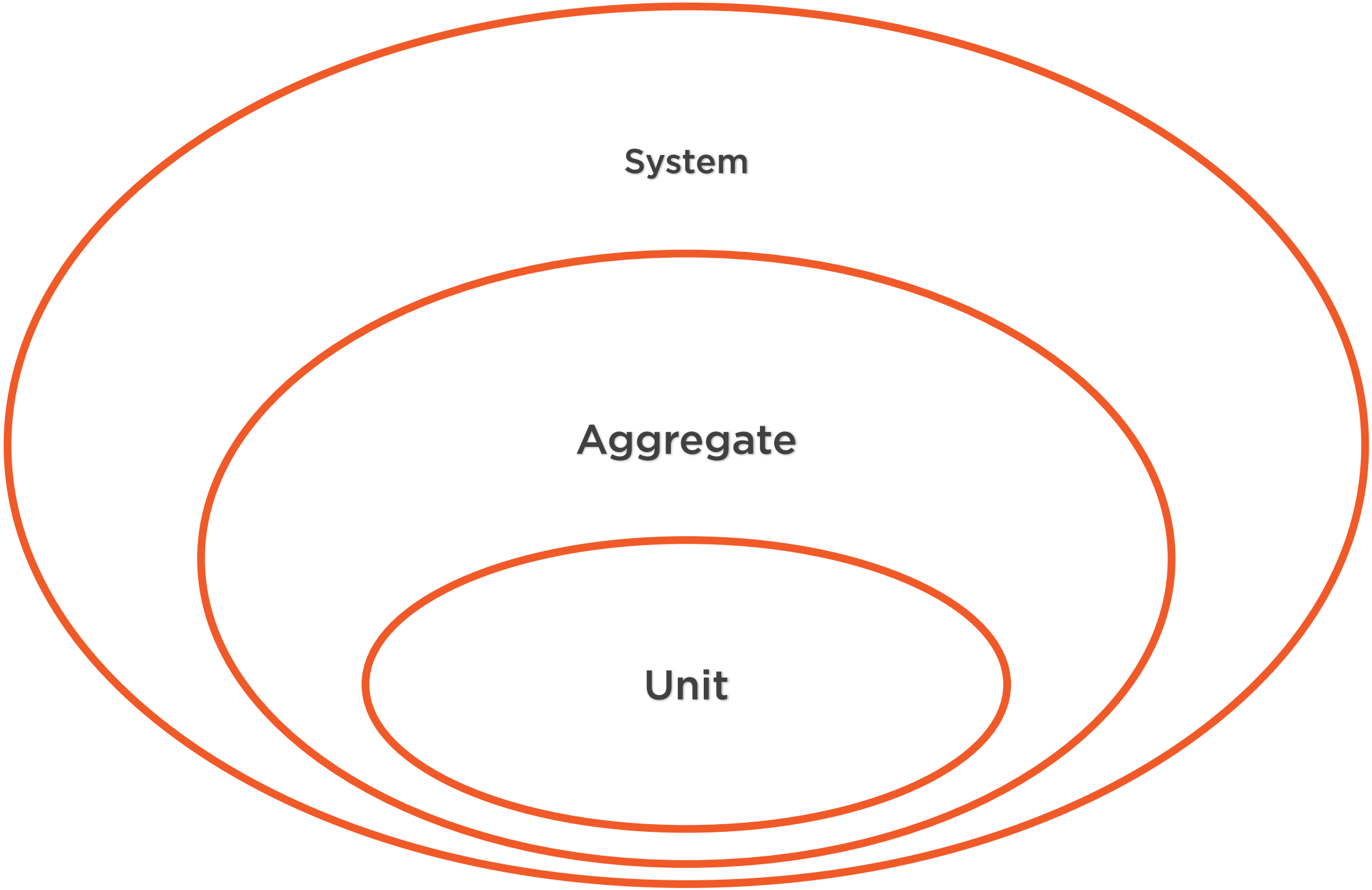
Writing Good Tests

Introducing Test Drive Development

Testing with Dependencies

Introducing Outside-In Testing

The Testing Hierarchy

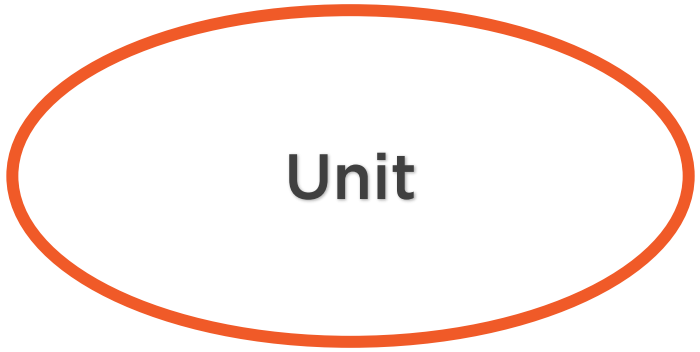

System
Aggregate
Unit

1 Class or Method

```
Math.min(0, 1) == 0
```

No non-trivial dependencies

Fast



Does a component do its job

A Storage system can list saved albums

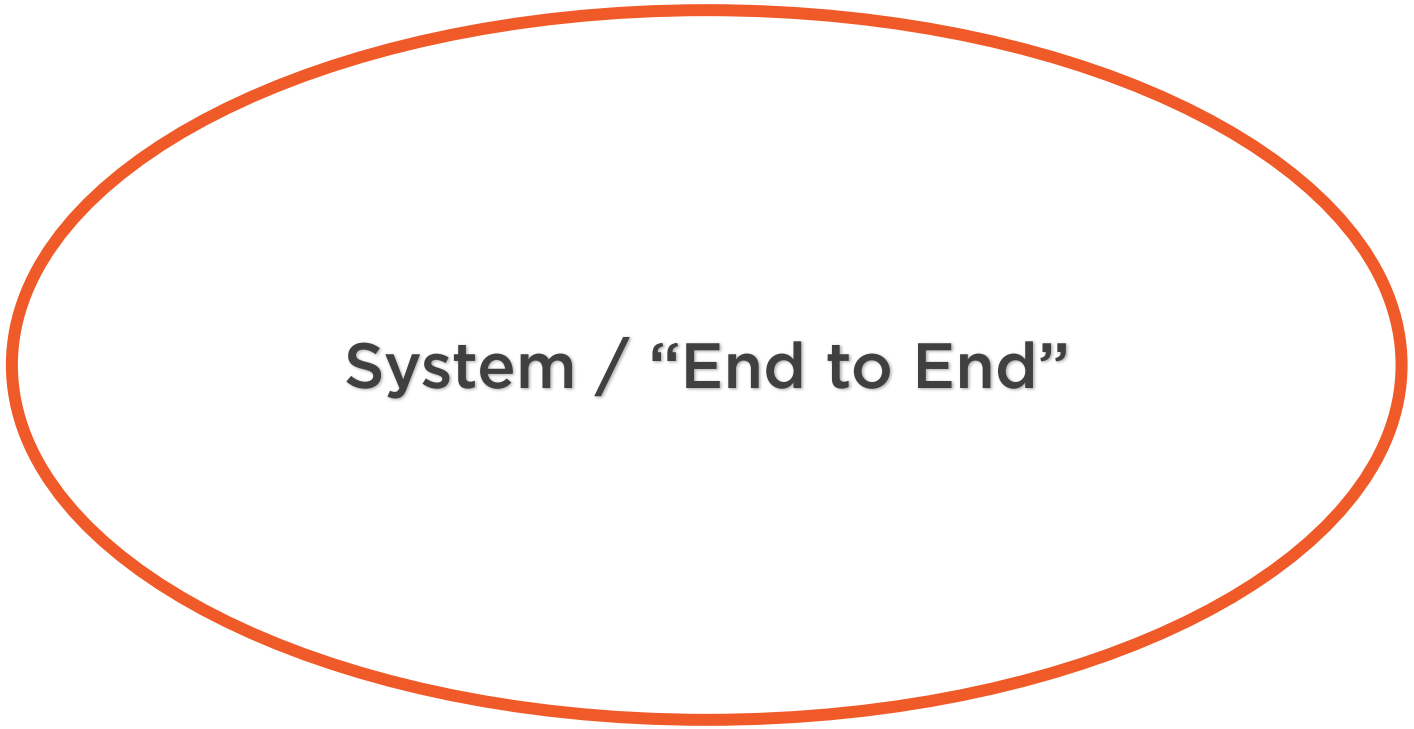
Slower

Meets requirements

System works together

Eg: Can buy widgets from our E-commerce app

Slowest



Writing Test Code

Junit

A Testing Framework

Common solutions to common problems

Practices > Frameworks

There are other frameworks – our focus is the practices

Cafe

`CoffeeType`

Enum of different types of Coffee

`Coffee`

Class for a single, brewed, cup

`Cafe`

Class for brewing coffee, including beans and milk

Components of a Test

Given

Preconditions

What state should your system be in when the behaviour happens?

When

The behaviour

What is being tested?

Then

The postcondition

What are the changes that happened

The Whole Test

Exceptions, Failures, and Errors

Sometimes an exception is the correct result!

Checking for Exceptions

```
// Junit 4
@Test(expected = IllegalArgumentException.class)


// Junit 5
assertThrows(IllegalArgumentException.class, () -> {
  // Code under test
}, “message");
```

Why test Exceptions?

They provide an executable form of documentation.

Failures vs. Errors

Failure	Error
Tried to check behaviour	At any point in the test
An assertion failed	A non-assert exception was thrown
The code is probably broken	The test is probably broken

Summary

You've learned:

- Why we should automate tests
- The structure of tests
- Failures vs. errors

Testing can be really simple!