# TDD Workshop

PHP UK Conference - Wednesday 16th February

Mark Bradley
Senior Consulting Engineer @ Armakuni

@braddle

github.com/braddle/php-uk-2022

Armakuni

# About me

Test Driven Development

youtube.com/c/TestingAllTheThings

Twitter

twitter.com/braddle

# What will we cover?

What is TDD and why use it?

Different Types of Tests

Test Doubles

Mutation Testing

# What is TDD?

Writing tests before you write new code

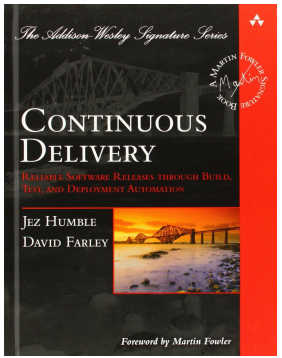
Using tests to design how an application should work

Not just unit tests

# Why?

# Why I do use TDD?





# Test First vs Test After

# Boring

# Tests are hard to write

# Easy to start skipping test

@braddle

# Fun!

# Easy to write

# You ~~don't~~ can't skip tests

# Does it slow down development?

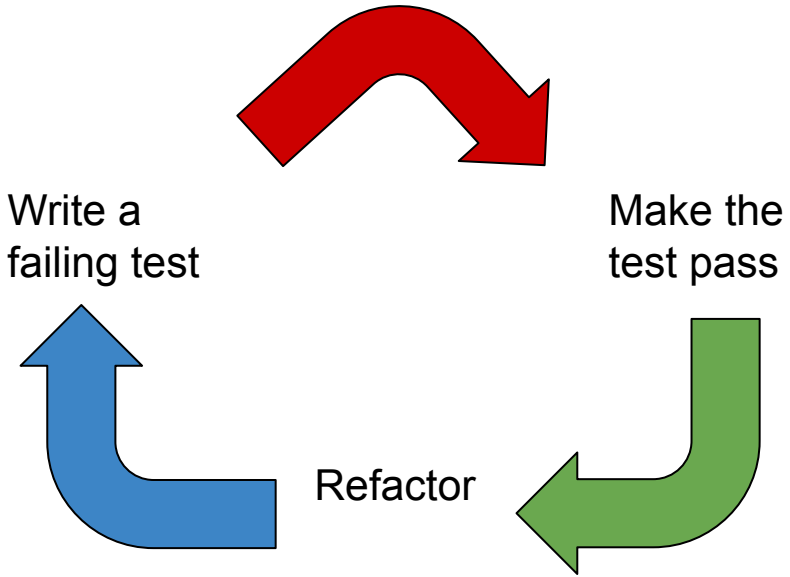
# Why use TDD

Fewer defects in your code

Only implement what is required

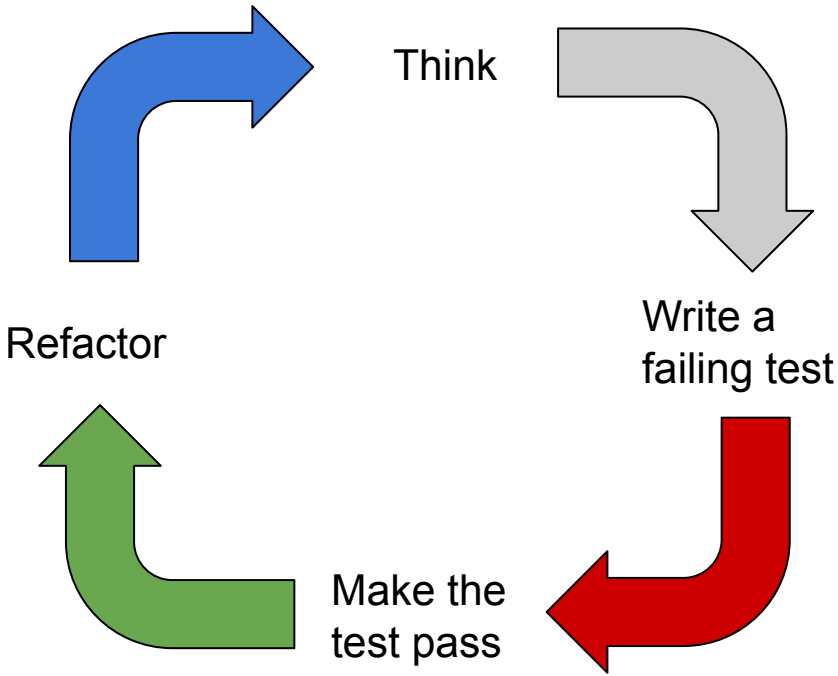
Increased code quality

# TDD Cycle



# Does it slow down development?

# TDD Cycle



# Arrange - Act - Assert

```php
public function testIsGreaterThan()
{
    // Arrange
    $five = new Integer(5);
    $four = new Integer(4);

    // Act
    $isGreaterThan = $five->isGreaterThan($four);
    $notGreaterThan = $four->isGreaterThan($five);

    // Assert
    $this->assertTrue($isGreaterThan);
    $this->assertFalse($notGreaterThan);
}
```

```php
public function testIsGreaterThan()
{
    // Arrange
    $five = new Integer(5);
    $four = new Integer(4);

    // Act & Assert
    $this->assertTrue($five->isGreaterThan($four));
    $this->assertFalse($four->isGreaterThan($five));
}
```

# Demo 1

Stack

First In Last Out

# Task 1

Queue

First In First Out (FIFO)

# Good Practices

Low coupling and high cohesion

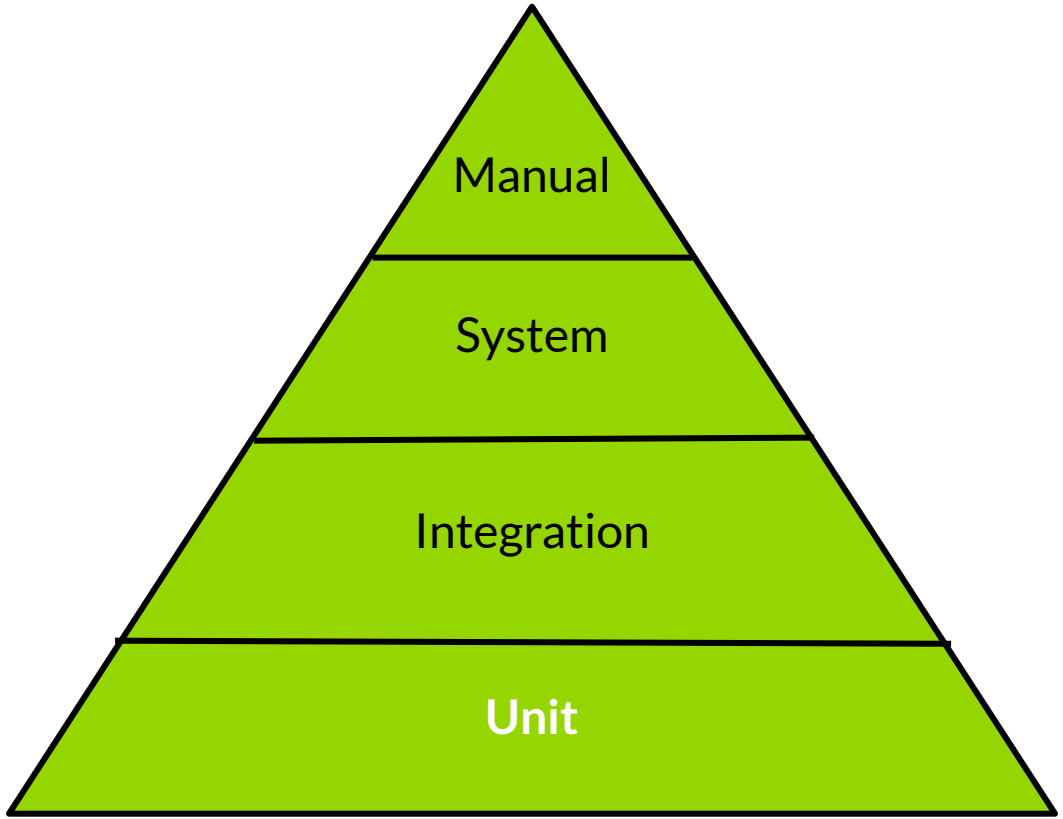
Ask don’t tell

SOLID Principles

Hexagonal Architecture

# Types of Tests

# Testing Pyramid



# Unit Tests

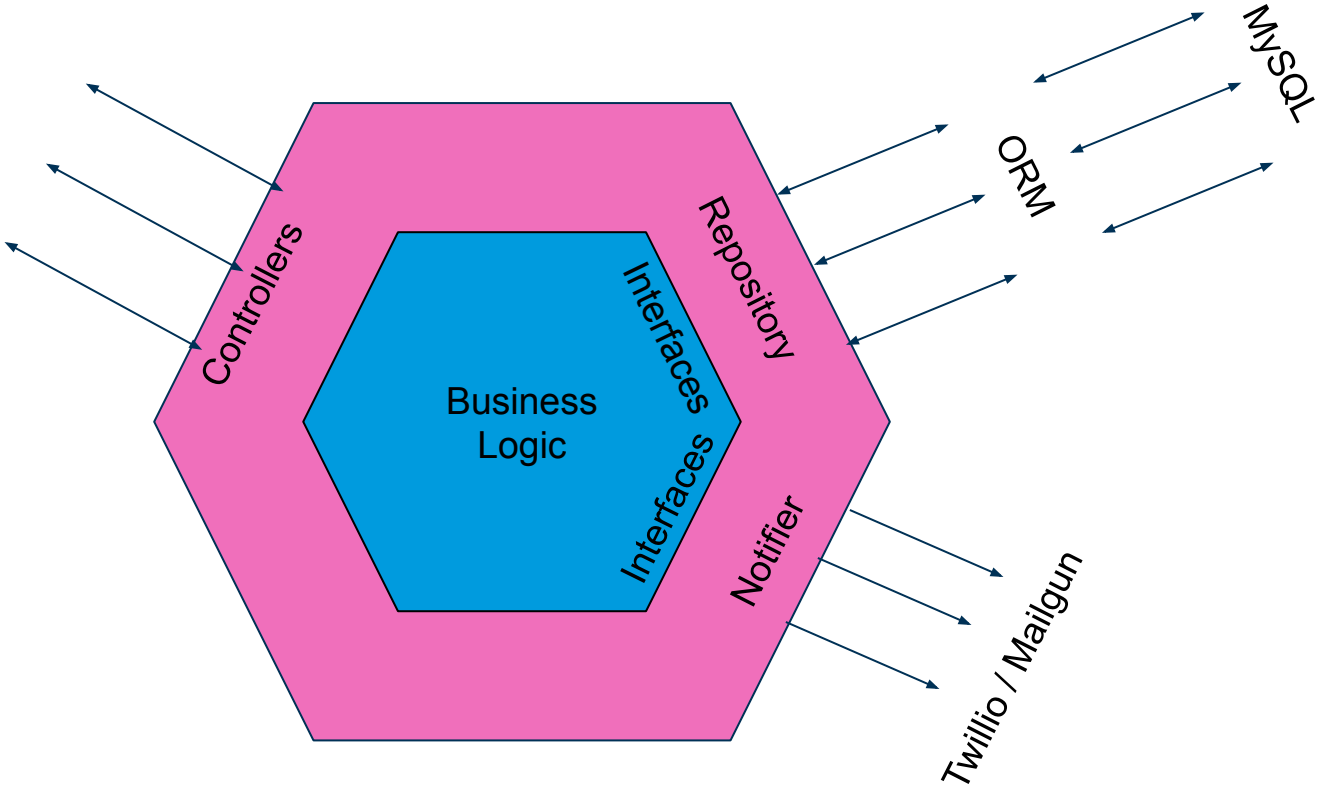
Targeted

Isolated

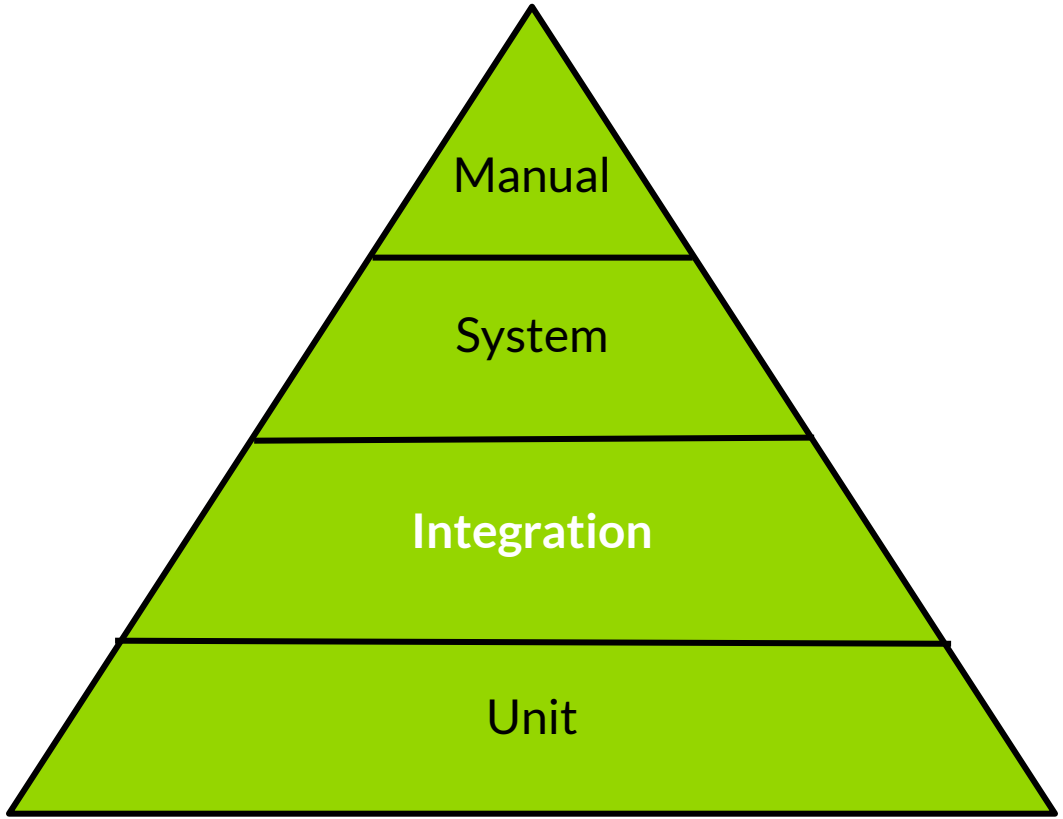
Repeatable & predictable

Fast

100% Code Coverage


MySQL
ORM
Repository
Controllers
Interfaces
Business
Logic
Interfaces
Notifier
Twillio / Mailgun

# Testing Pyramid



# Integration Tests

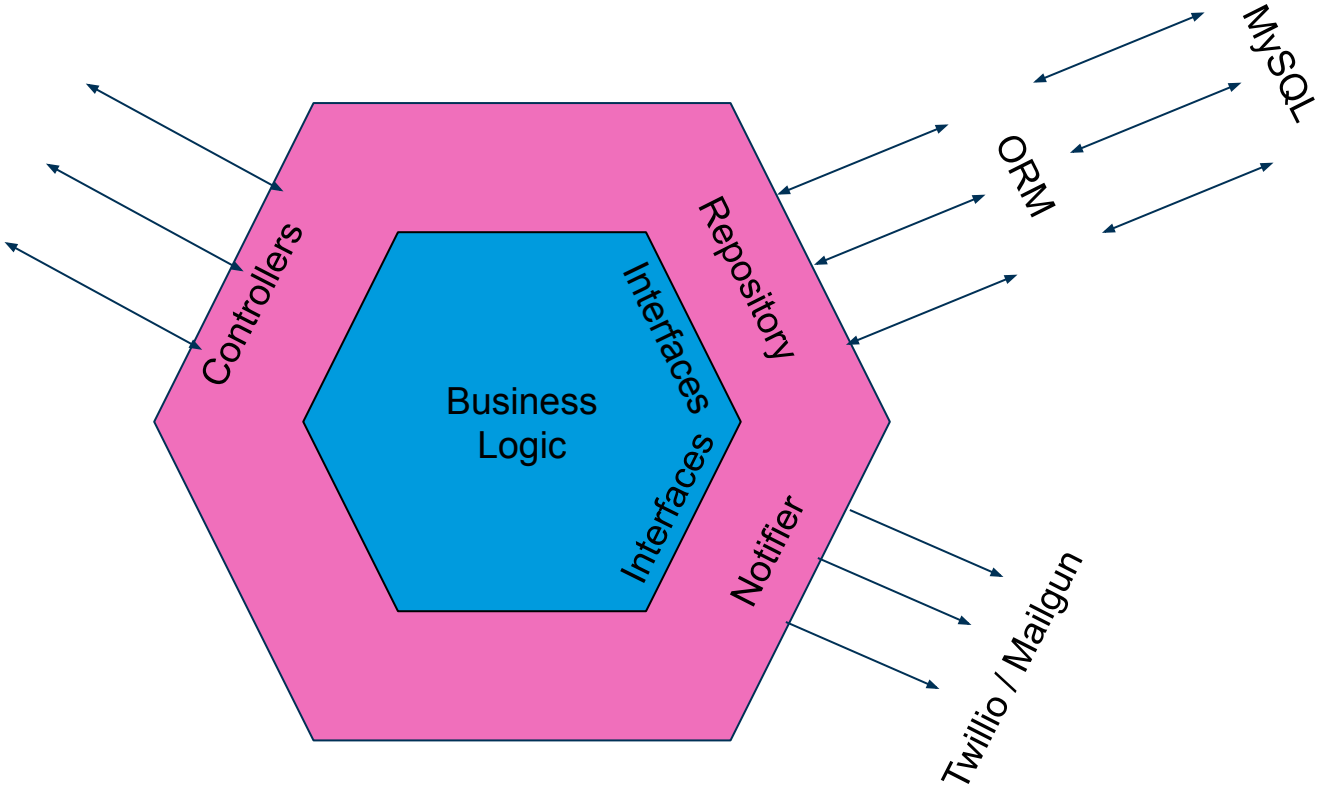
Tests a small number of units or component together

Ensure the different units or component work together as expected

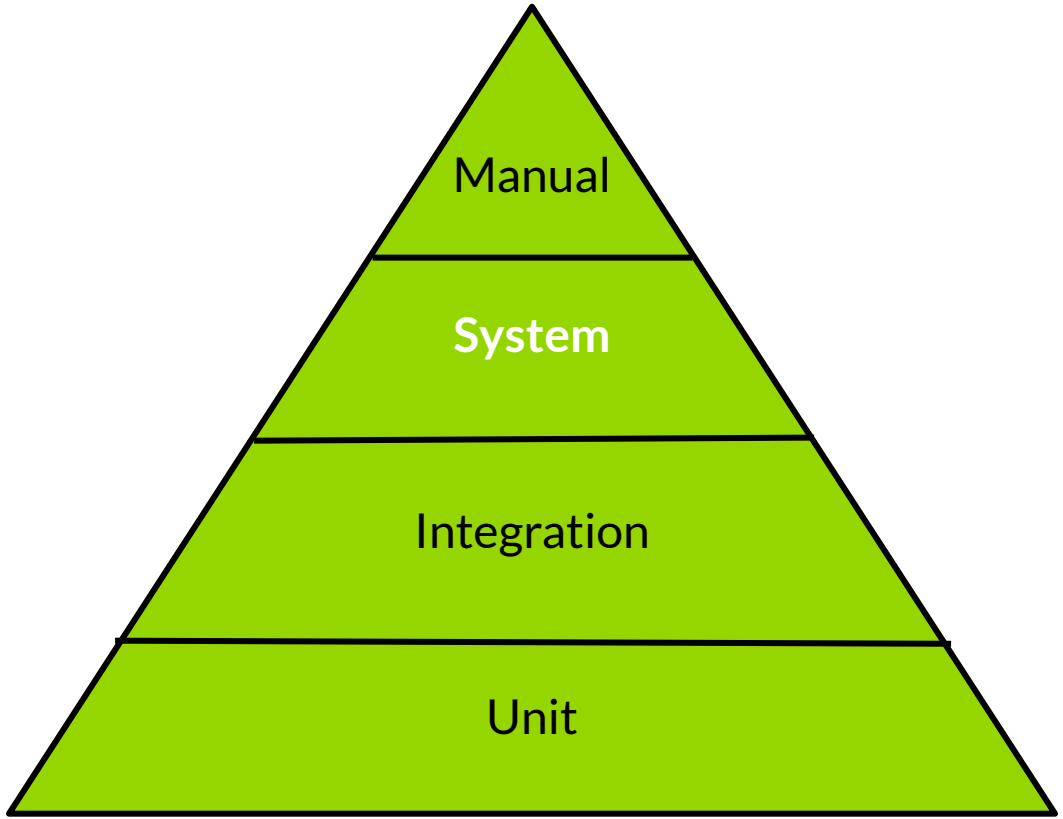
Mocks external dependencies (Database, Email, ….)

Could be behavioural Tests

10% Code Coverage


MySQL
ORM
Repository
Controllers
Interfaces
Business
Logic
Interfaces
Notifier
Twillio / Mailgun

# Testing Pyramid



# System (End to End) Tests

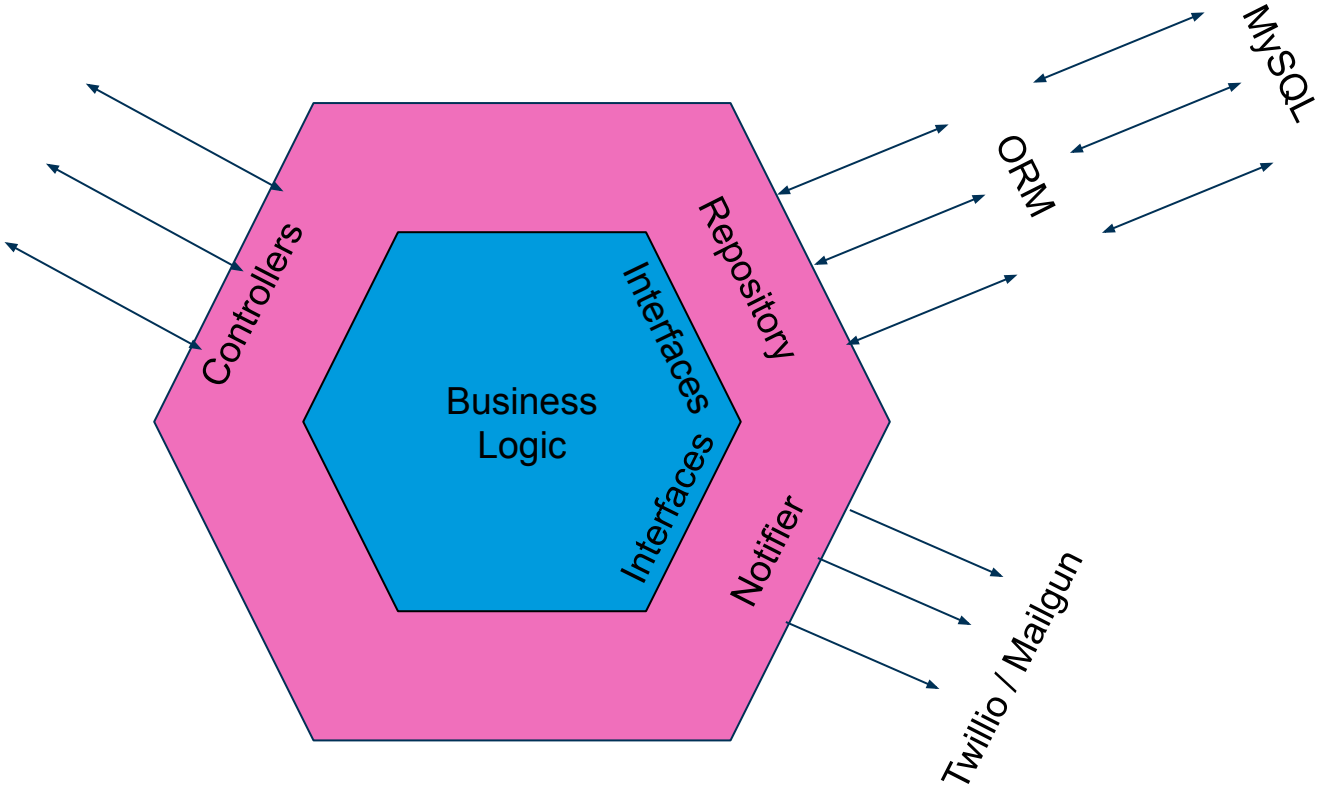
Flows through you application, usually a few core journeys

Uses all really services (Database, Email, …)

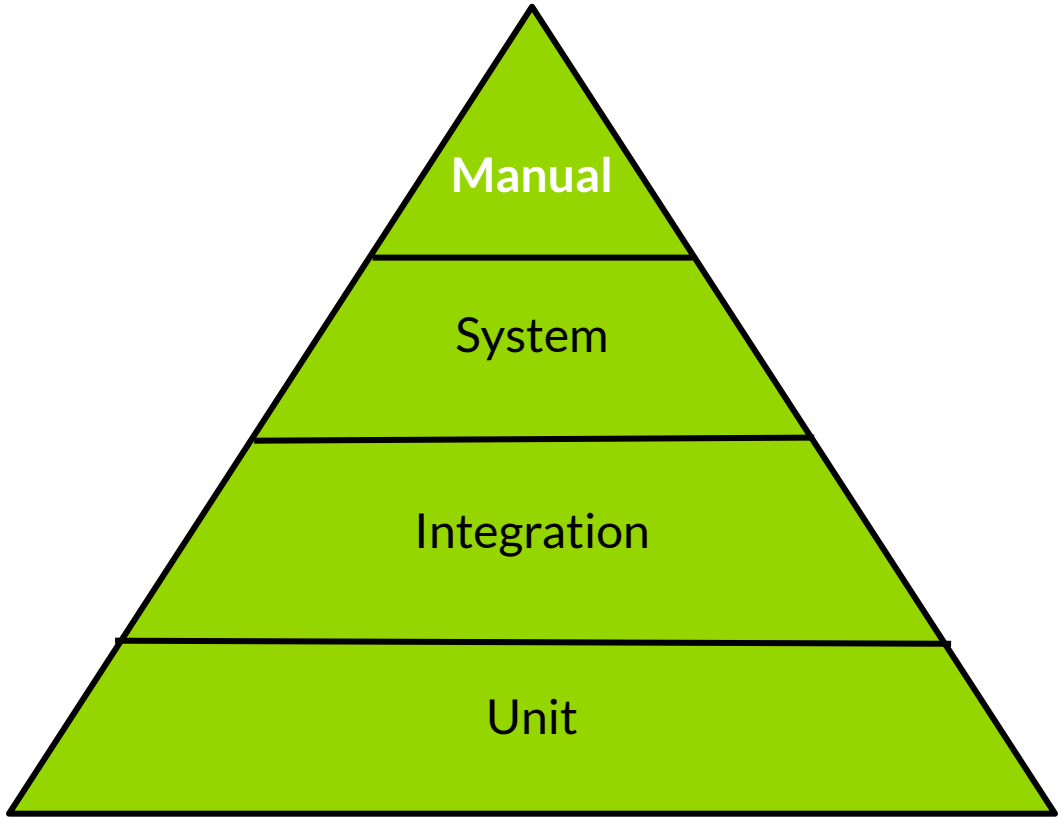
May interact with many different parts of you app in a single test

May require some seeding of external services

5% Code Coverage


MySQL
ORM
Repository
Controllers
Interfaces
Business
Logic
Interfaces
Notifier
Twillio / Mailgun

# Testing Pyramid



# Manual Tests

Still has its place

Exploratory Testing

Penetration Testing

Load Testing

# Task 2

Bowling Game Calculator

# Bowling Score Calculator

Given a string that represents a complete game of 10 pin bowling return the score of that came

X X X X X X X X X X X X = 300

-- -- -- -- -- -- -- -- -- -- -- -- = 0

5/ 5/ 5/ 5/ 5/ 5/ 5/ 5/ 5/ 5/ 5/ = 150

1- -1 1- -1 1- -1 1- -1 1- -1 = 10

11 11 11 11 11 11 11 11 11 11 = 20

# Dependencies

# Test Doubles

Imitating the functionality of dependencies

Allowing for isolation of unit and integration tests

Do not necessarily require a mocking framework

Implement the interface of a dependency

# Stubs

Implements the interface

Returns specific values

```
public class ZeroNumberGeneratorStub implements RandomNumberGenerator {

    public String createRandomNumber(int numberOfDigits) {
        return "000";
    }
}
```

```
public class EligableDriverStub implements Person {

  public boolean isEligibleForDrivingLicence() {
     return true;
  }

  public String getInitials() {
     return "MDB";
  }

  public String getFormattedDateOfBirth(String format) {
     return "19970612";
  }
}
```

@braddle

```
public class DrivingLicenceNumberGeneratorTest {

  @Test
  public void validDrivingLicence() throws Exception {
     RandomNumberGenerator stub = new ZeroNumberGeneratorStub();
     Person eligbleDrive = new EligableDriverStub();

     DrivingLicenceNumberGenerator generator = new DrivingLicenceNumberGenerator(stub);
     String actualLicenceNumber = generator.generate(eligbleDrive);

     assertEquals("MDB19970612000", actualLicenceNumber);
  }
}
```

```
public class DrivingLicenceNumberGenerator {

  private RandomNumberGenerator numberGenerator;

  public String generate(Person person) throws Exception {
      …

      return person.getInitials() + person.getFormattedDateOfBirth("Ymd") + numberGenerator.createRandomNumber(3);
  }
}
```

# Spies

Implements the interface

Returns specific values

Tracks call count and calling values

```java
public class LoggerSpy implements Logger {

  public int errorCallCount = 0;
  public String lastErrorMessage;

  @Override
  public void error(String log) {
    errorCallCount++;
    lastErrorMessage = log;
  }
}
```

```
public class DrivingLicenceNumberGeneratorTest {

  @Test
  public void ineligableLicenceRequestLogged() {
      …
      LoggerSpy logger = new LoggerSpy();

      DrivingLicenceNumberGenerator generator = new DrivingLicenceNumberGenerator(numberGeneratorDummy, logger);

      …

      assertEquals(1, logger.errorCallCount);
      assertEquals(
              "Request made for a driving licence number by an ineligible person: 1234",
              logger.lastErrorMessage
      );
  }
}
```

```
public class DrivingLicenceNumberGenerator {

  private RandomNumberGenerator numberGenerator;
  private Logger logger;


  public void generate(Person person) throws Exception {
      if (!person.isEligibleForDrivingLicence()) {
          logger.error("Request made for a driving licence number by an ineligible person: " + person.getId());
          throw new Exception("message does not matter");
      }
      …
  }
}
```

# Mocks

Implement the interface

Returns specific values

Tracks call count and calling values

Setup by the test

```
public class DrivingLiceneceNumberGeneratorTest {

  @Test
  public void licenceNumbersAreAtleast14Characaters() throws Exception {
     RandomNumberGenerator numberGeneratorMock = mock(RandomNumberGenerator.class);
     when(numberGeneratorMock.createRandomNumber(3)).thenReturn("333");
     when(numberGeneratorMock.createRandomNumber(4)).thenReturn("4444");
     when(numberGeneratorMock.createRandomNumber(5)).thenReturn("55555");

     DrivingLiceneceNumberGenerator generator = new DrivingLiceneceNumberGenerator(numberGeneratorMock);

     Person oneInitials = new EligiblePersonStub("A");
     Person twoInitials = new EligiblePersonStub("AB");
     Person threeInitials = new EligiblePersonStub("ABC");
     Person fourInitials = new EligiblePersonStub("ABCD");

     assertEquals("A2010111255555", generator.generate(oneInitials));
     assertEquals("AB201011124444", generator.generate(twoInitials));
     assertEquals("ABC20101112333", generator.generate(threeInitials));
     assertEquals("ABCD20101112333", generator.generate(fourInitials));
  }
}
```

```
public class DrivingLiceneceNumberGenerator {
  private RandomNumberGenerator numberGenerator;

  public DrivingLiceneceNumberGenerator(RandomNumberGenerator numberGenerator) {
     this.numberGenerator = numberGenerator;
  }

  public String generate(Person person) throws Exception {
     if (!person.isEligibleForDrivingLicence()) {
        throw new Exception("Message does not matter");
     }

     String licence =  person.getInitials() + person.getFormattedDateOfBirth("TODO");

     if (licence.length() < 11) {
        licence = licence + numberGenerator.createRandomNumber(14 - licence.length());
     } else {
        licence = licence + numberGenerator.createRandomNumber(3);
     }

     return licence;
  }
}
```

# Fakes

Implement the interface

Will contain some “Real” business logic

```java
public class ZeroNumberGeneratorFake implements RandomNumberGenerator {

  @Override
  public String createRandomNumber(int numberOfDigits) {
    String numbers = "";

    for (int i = 0; i < numberOfDigits; i++) {
      numbers = numbers + "0";
    }

    return numbers;
  }
}
```

```java
public class DrivingLiceneceNumberGeneratorTest {

  @Test
  public void licenceNumbersAreAtleast14Characaters() throws Exception {
    RandomNumberGenerator numberGeneratorFake = new ZeroNumberGeneratorFake();

    DrivingLiceneceNumberGenerator generator = new DrivingLiceneceNumberGenerator(numberGeneratorFake);

    Person oneInitials = new EligiblePersonStub("A");
    Person twoInitials = new EligiblePersonStub("AB");
    Person threeInitials = new EligiblePersonStub("ABC");
    Person fourInitials = new EligiblePersonStub("ABCD");

    assertEquals("A2010111200000", generator.generate(oneInitials));
    assertEquals("AB201011120000", generator.generate(twoInitials));
    assertEquals("ABC20101112000", generator.generate(threeInitials));
    assertEquals("ABCD20101112000", generator.generate(fourInitials));
  }
}
```

```
public class DrivingLiceneceNumberGenerator {
  private RandomNumberGenerator numberGenerator;

  public DrivingLiceneceNumberGenerator(RandomNumberGenerator numberGenerator) {
     this.numberGenerator = numberGenerator;
  }

  public String generate(Person person) throws Exception {
     if (!person.isEligibleForDrivingLicence()) {
        throw new Exception("Message does not matter");
     }

     String licence =  person.getInitials() + person.getFormattedDateOfBirth("TODO");

     if (licence.length() < 11) {
        licence = licence + numberGenerator.createRandomNumber(14 - licence.length());
     } else {
        licence = licence + numberGenerator.createRandomNumber(3);
     }

     return licence;
  }
}
```

# Test Doubles

Imitating the functionality of dependencies

Allowing for isolation of unit and integration tests

Do not necessarily require a mocking framework

Implement the interface of a dependency

# Task 3

Test Doubles

# Demo 2

Mutation Testing

# Resources

- 30 Days of TDD by James Bender (@jamesbender)
  - http://www.telerik.com/blogs/30-days-tdd-day-one-what-is-tdd
- Code Coverage: Testing Private Functions (Me)
  - http://mark-bradley.net/2017/03/11/code-coverage-testing-private-functions/
- Testing All The Things
  - youtube.com/c/TestingAllTheThings
-

# Katas & Dojos

- codingdojo.org/kata
- Kata-log.rocks

- meetup.com/London-Code-Dojo/
- meetup.com/london-software-craftsmanship/

# Feedback

Twitter: @braddle

Email: braddle@gmail.com

# Thank you

@braddle