https://github.com/braddle/php-uk-2020

@braddle

Sainsbury's live well for less

TDD Workshop

PHP UK - Wednesday 19th February 2020

Mark Bradley
Principal Software Engineer

@braddle

What will we cover?

What is TDD and why use it?

Testing Pyramid

Test Doubles (Mocks, Stubs, Fakes, Spies, Dummies)

What is TDD?

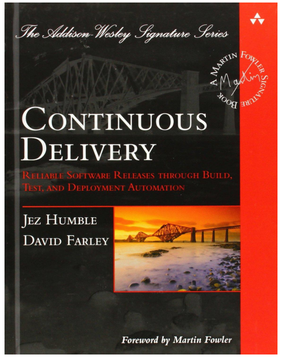
Writing tests before you write new code

Using tests to design how an application should work

Not just unit tests

Why?

Why I do use TDD?


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler

Sainsbury's live well for less

Test First vs Test After

—

Boring

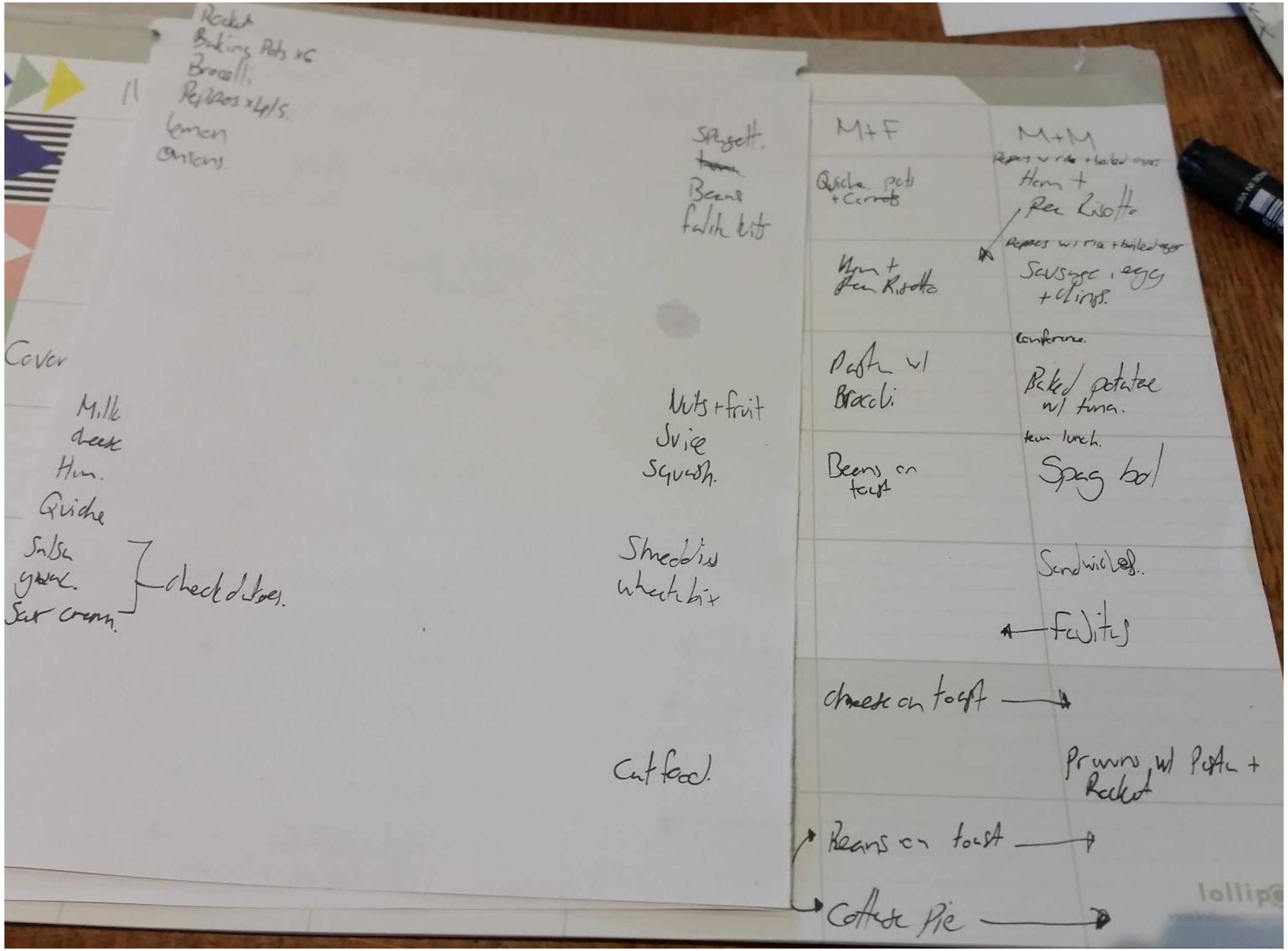
Tests are hard to write

Easy to start skipping test

Fun!

Easy to write

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



Rocket
Baking Pots x6
Brocolli
Peppers x4/5.
Lemon
Onions.

Cover

Milk
cheese
Ham.
Quiche
Salsa
guac.
Sour cream
check dates.

Spaghetti
~~ham~~
Beans
fajita kits

Nuts + fruit
Juice
Squash.

Shreddies
wheetabix

Cat food.

M+F	M+M
Quiche pots + Carrots	Peppers w rice + boiled eggs Ham + Pea Risotto
Ham + Pea Risotto	Peppers w rice + boiled eggs Sausage, egg + chips.
Pasta w/ Brocoli	conference. Baked potatoe w/ tuna.
Beans on toast	team lunch. Spag bol
	Sandwiches.
	Fajitas
cheese on toast	
	Prawns w/ Pasta + Rocket
Beans on toast	
Cottage Pie	

@braddle

Sainsbury's live well for less

Does it slow down development?

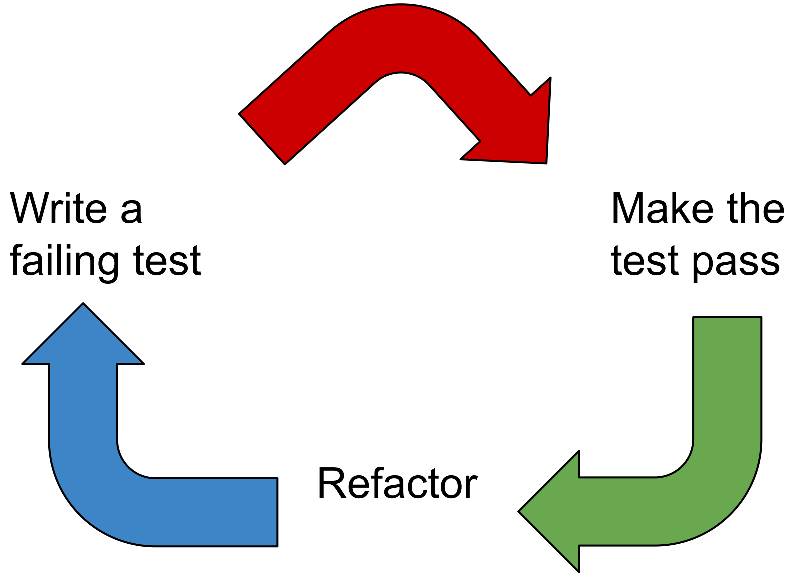
Why use TDD

Fewer defects in your code

Only implement what is required

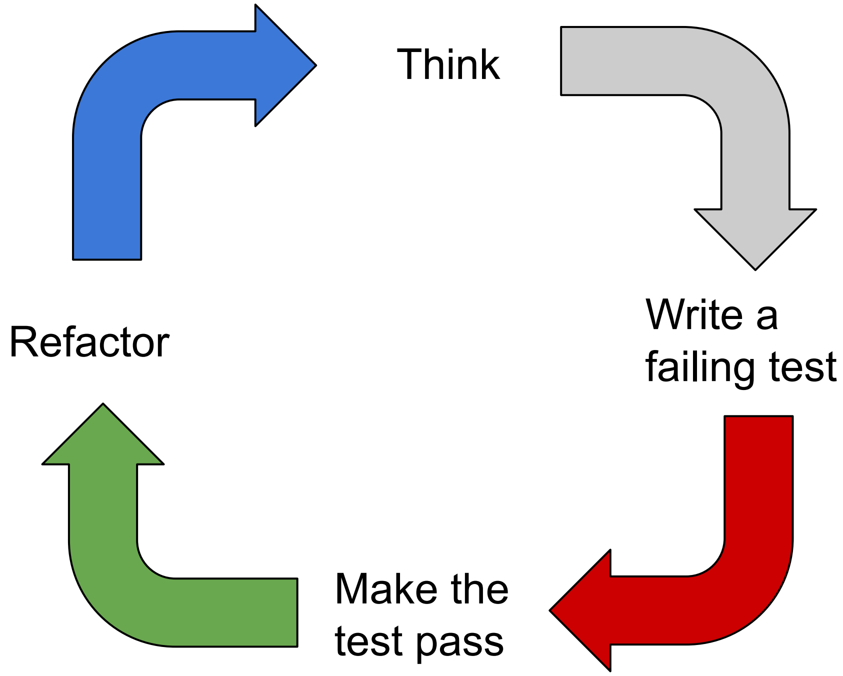
Increased code quality

TDD Cycle



Does it slow down development?

TDD Cycle



Arrange - Act - Assert

@braddle

Sainsbury's live well for less

```
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);
}
```

@braddle

Sainsbury's live well for less

```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

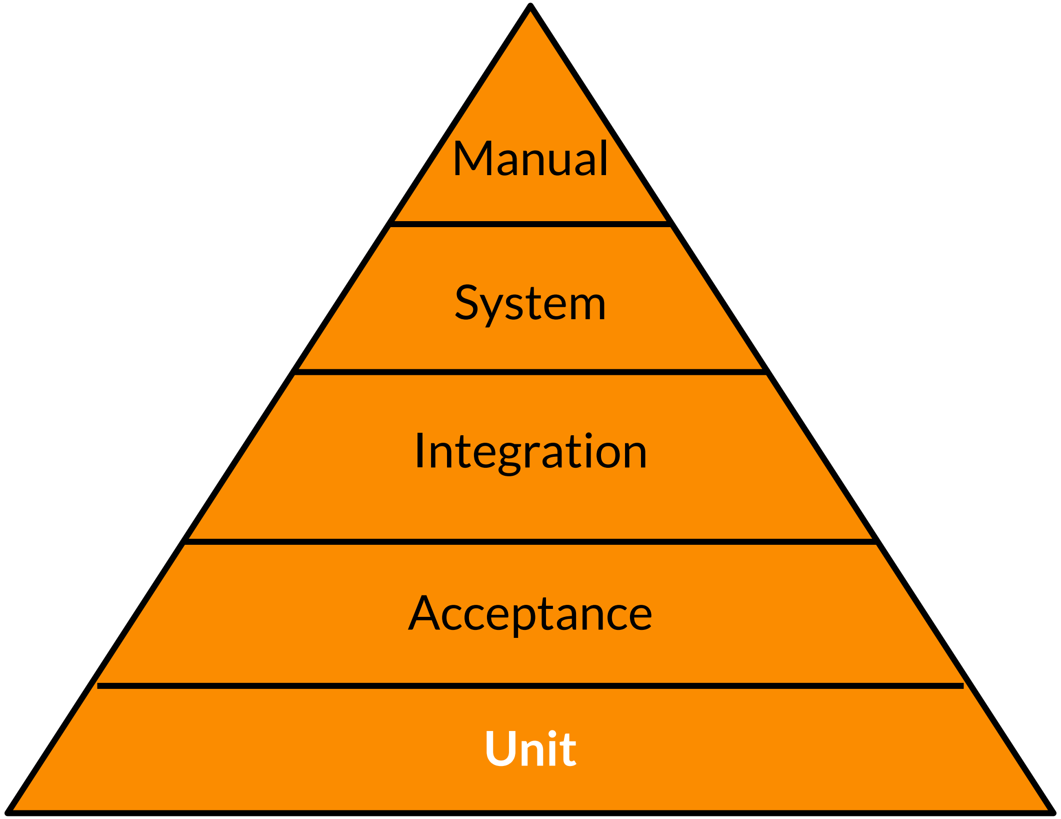
Ask don’t tell

SOLID Principles

Hexagonal Architecture

Types of Tests

Testing Pyramid



Sainsbury's live well for less

Unit Tests

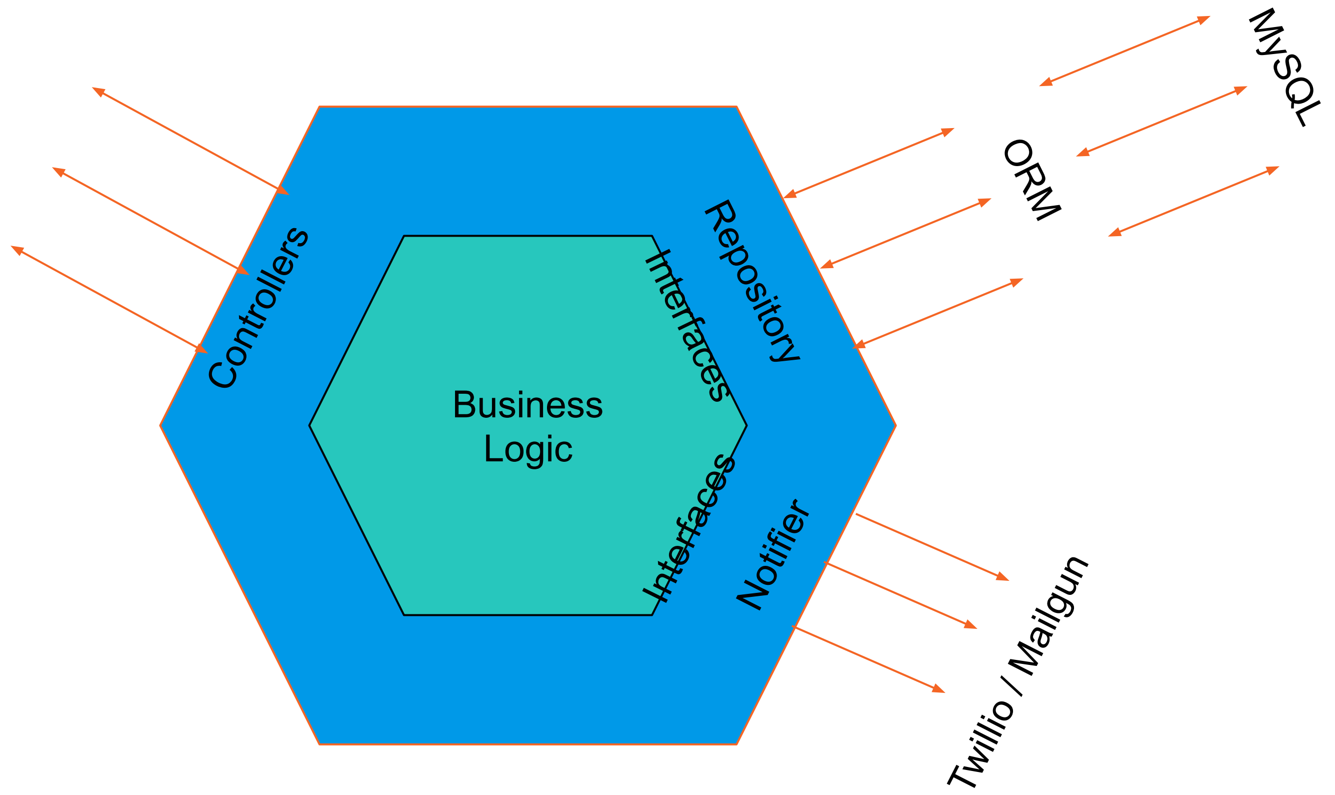
Targeted

Isolated

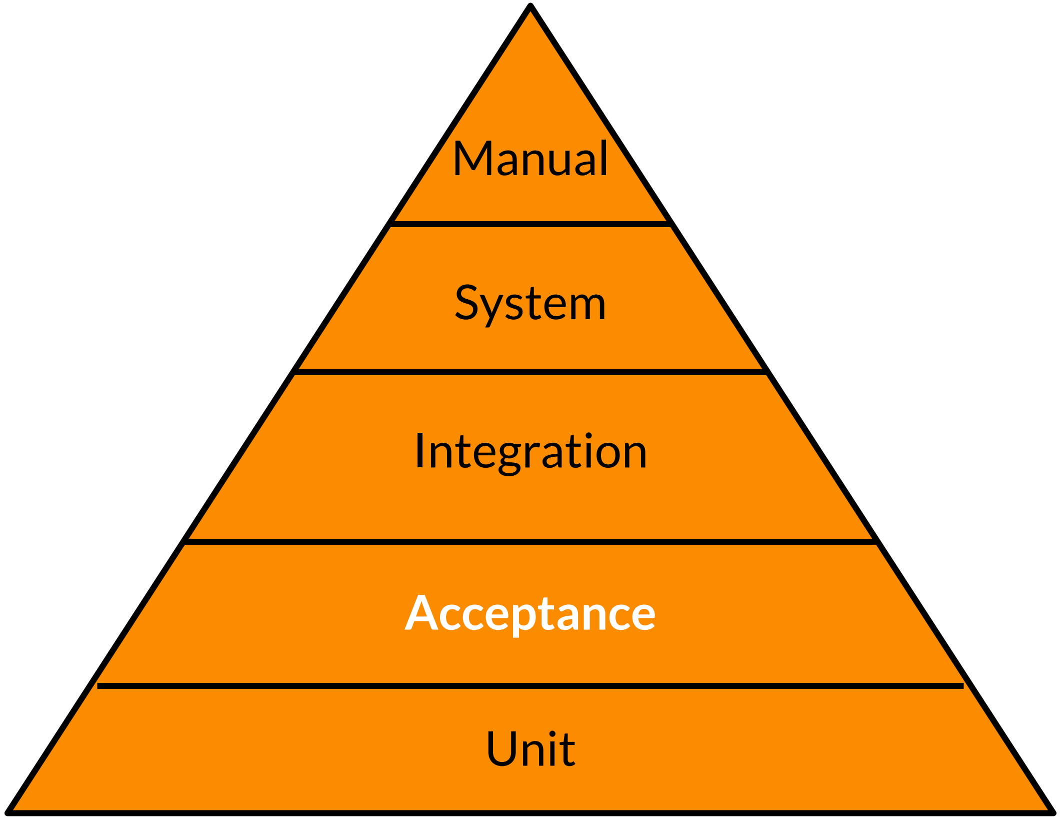
Repeatable & predictable

Fast

100% Code Coverage


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

Testing Pyramid



@braddle

Sainsbury's live well for less

Acceptance Tests

Could use Behaviour Driven Development if Product Owner engaged

If Product Owner is not engaged just use plain automated testing

50% Code Coverage

Behaviour Driven Development

Conversation between Developers and Product Owners

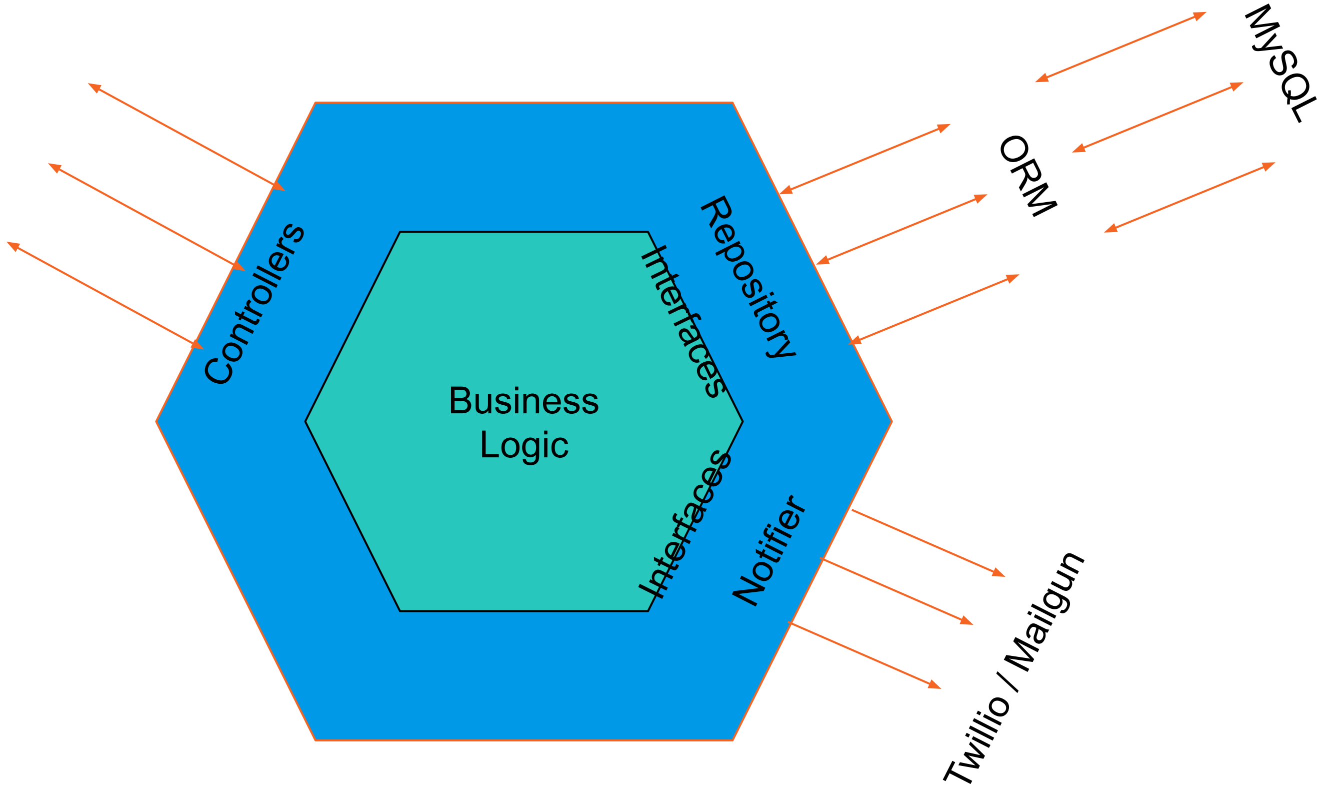
Drive out expected behaviour of a feature

Uses Gerkin to create scenarios understood by devs and POs

Passing tests == feature ready to ship

No need for manual acceptance

Pointless without conversation or PO using for acceptance


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

Gerkin

Feature: Name of the feature/functionality to be implemented
As an Who the feature will help
I want to What the person want to be able to do
In order to What the feature achieves

Scenario: a specific interaction with the new functionality
Given Put the system into a known state
When Perform the action to being tested
Then Observe the outcomes of the action

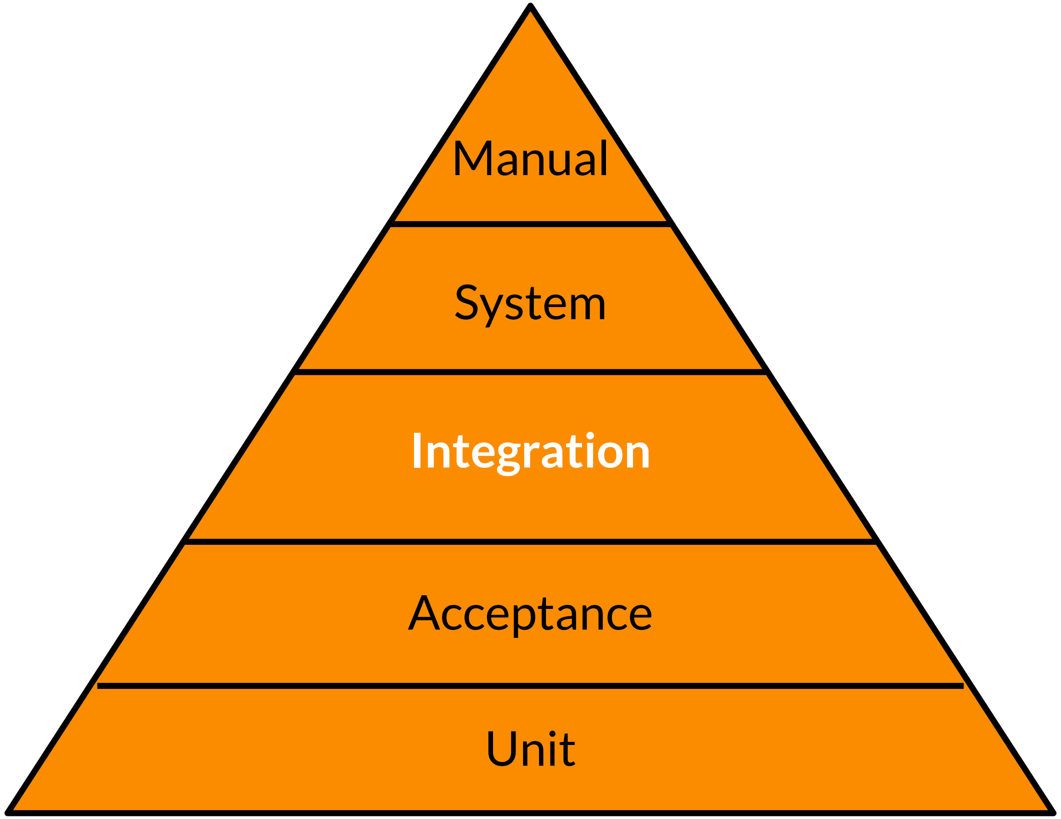
Feature

```
Feature: Adding to basket
    As a Customer
    I want to be able to add products to my basket
    In order to buy them

    Scenario: The one where the customer buys a Playstation
        Given There is a product “Playstation”
        When I add a “playstation” to my basket
        Then the “Playstation” is in my basket
```

Feature - Bad Example

```
Feature: Adding to basket
    As a Customer
    I want to be able to add products to my basket
    In order to buy them

    Scenario: The one where the customer buys a Playstation
        Given There is a product "Playstation"
        When I goto "/product/playstation/123"
        And I click the "Add to basket" button
        Then the "Playstation" is in my basket
```

Testing Pyramid



Integration Tests

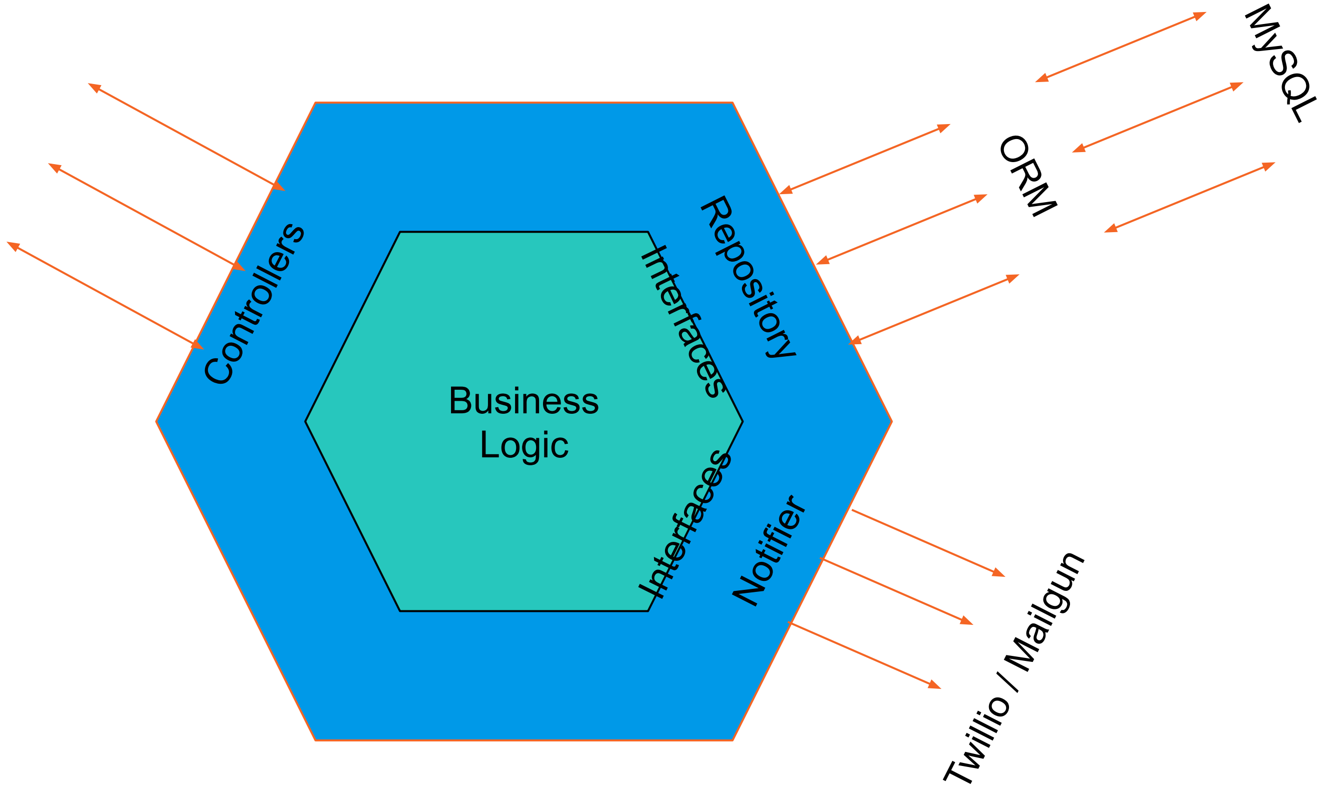
Tests a small number of units or component together

Ensure the different units or component work together as expected

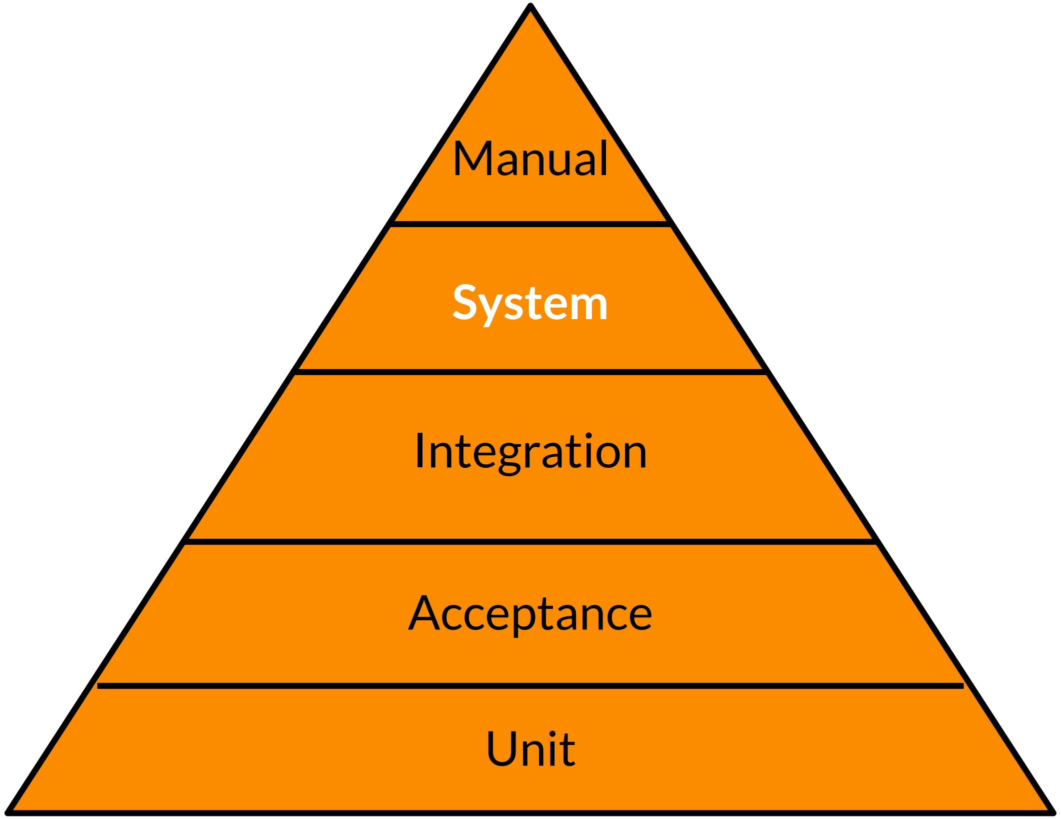
Mocks external dependencies (Database, Email, ...)

Could be behavioural Tests

10% Code Coverage


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

Testing Pyramid



System (End to End) Tests

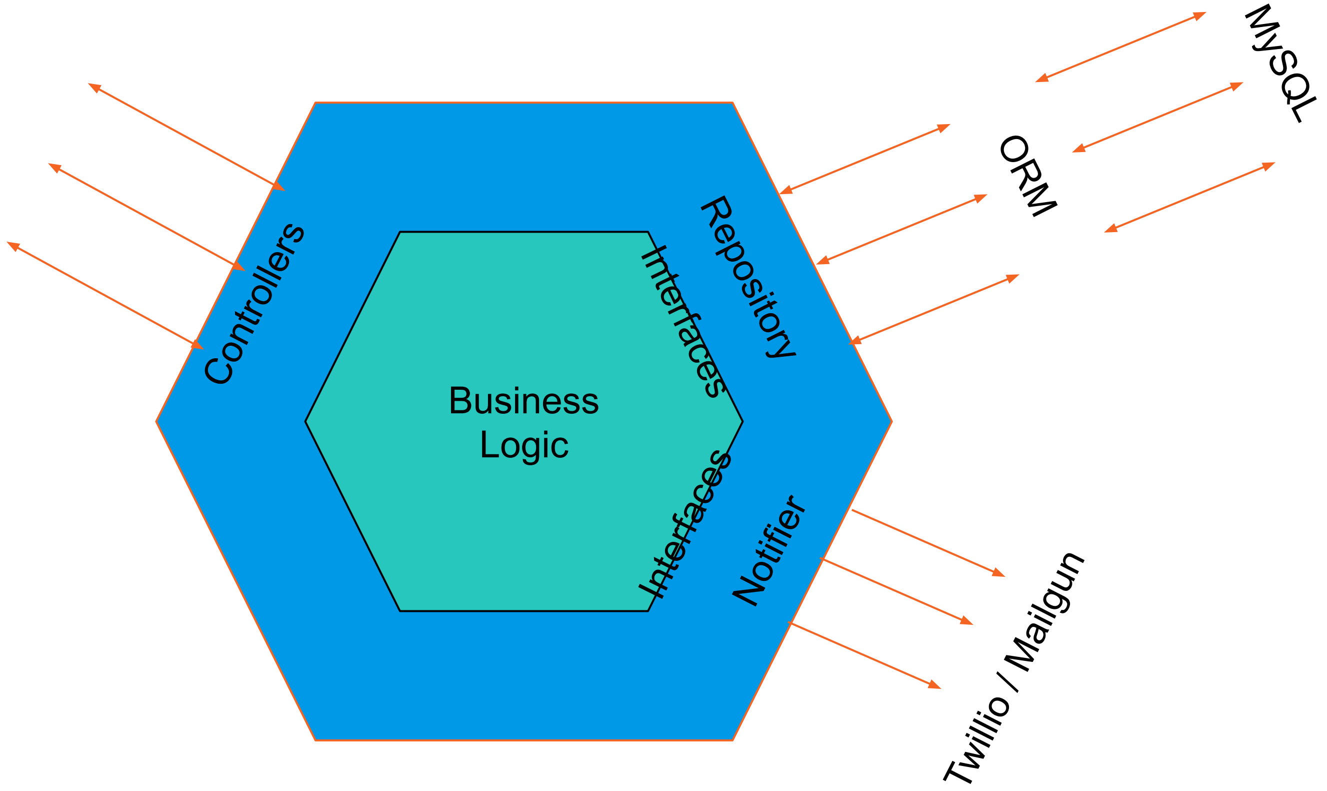
Flows through you application, usually a few core journeys

Uses all really services (Database, Email, …)

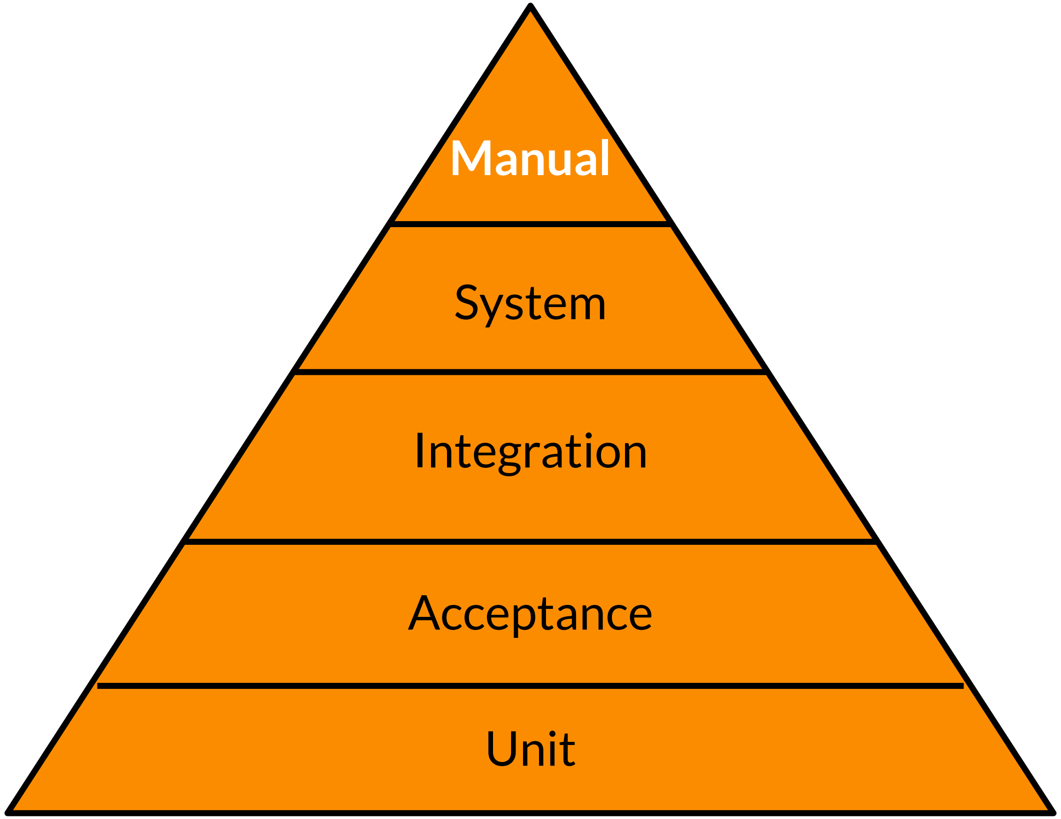
May interact with many different parts of you app in a single test

May require some seeding of external services

5% Code Coverage


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

Testing Pyramid



Manual Tests

Still has its place

Exploratory Testing

Penetration Testing

Other Types of Tests

API Contract Testing

Load (Volume and Performance)

Mutation 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 XXX = 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

Dummies

Just Implements the inteface

All functions are stubs

```java
public class RandomNumberGeneratorDummy implements RandomNumberGenerator {

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

```java
public class DrivingLicenceNumberGeneratorTest {

  @Test(expected = Exception.class)
  public void exceptionThrowForIneligiblePerson() throws Exception {
     RandomNumberGenerator dummy = new RandomNumberGeneratorDummy();
     Person ineligbleDrive = new IneligableDriverStub();

     DrivingLicenceNumberGenerator generator = new DrivingLicenceNumberGenerator(dummy);
     generator.generate(ineligbleDrive);
  }
}
```

```java
public class DrivingLicenceNumberGenerator {

  public DrivingLicenceNumberGenerator(RandomNumberGenerator numberGenerator) {
  }

  public void generate(Person person) throws Exception {
      if (!person.isEligibleForDrivingLicence()) {
          throw new Exception("Person is not elibible for a driving license");
      }
  }
}
```

Stubs

Implements the interface

Returns specific values

```java
public class ZeroNumberGeneratorStub implements RandomNumberGenerator {

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

@braddle

Sainsbury's live well for less

```
public class EligableDriverStub implements Person {

  @Override
  public boolean isEligibleForDrivingLicence() {
      return true;
  }

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

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

```java
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 DrivingLicenceNumberGenerator(RandomNumberGenerator numberGenerator) {
     this.numberGenerator = numberGenerator;
  }

  public String generate(Person person) throws Exception {
     if (!person.isEligibleForDrivingLicence()) {
        throw new Exception("Person is not elibible for a driving license");
     }

     return person.getInitials() + person.getFormattedDateOfBirth("TODO") + 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;
  }
}
```

```java
public class DrivingLicenceNumberGeneratorTest {

    @Test
    public void ineligableLicenceRequestLogged() {
        Person ineligablePerson = new IneligableDriverStub();
        RandomNumberGenerator numberGeneratorDummy = new RandomNumberGeneratorDummy();
        LoggerSpy logger = new LoggerSpy();

        DrivingLicenceNumberGenerator generator = new DrivingLicenceNumberGenerator(numberGeneratorDummy, logger);

        try {
            generator.generate(ineligablePerson);
        } catch (Exception e) {
            // Do nothing
        }

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

@braddle

```java
public class DrivingLicenceNumberGenerator {

  private RandomNumberGenerator numberGenerator;
  private Logger logger;

  public DrivingLicenceNumberGenerator(RandomNumberGenerator numberGenerator, Logger logger) {
      this.numberGenerator = numberGenerator;
      this.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

```java
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));
  }
}
```

```java
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;
  }
}
```

@braddle

```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));
  }
}
```

```java
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

Hello World

Hello, World (Interface)

Public function __construct (Clock $clock)

Public function greet(string $name) : string;

Hello World

Create an class that can make a time specific greeting to the given name

06:00 - 11:59 Good morning, <name>

12:00 - 16:29 Good afternoon, <name>

16:30 - 20:29 Good evening, <name>

20:30 - 23:59 Good night, <name>

24:00 - 05:59 Go to bed, <name>

Demo 2

Mutation Testing

—

"Learn the rules like a pro, so you can break them like an artist."
Pablo Picasso

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/
- SOLID Principles (Billie Thompson @PurpleBooth)
  - https://purplebooth.co.uk/blog/2015/02/23/s-is-for-single-responsibility-principle/
-

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

Feedback: https://joind.in/talk/1ada5

Sainsbury's live well for less

—

Thank you

@braddle

Sainsbury's live well for less