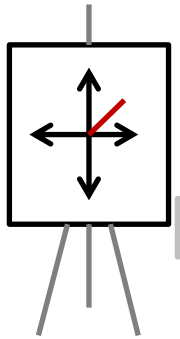
# Test Driven Development Microskills

*with Llewellyn Falco*

Week 2

# The Testing Circle



Whiteboard

English

Result

Code

## Testing Circle

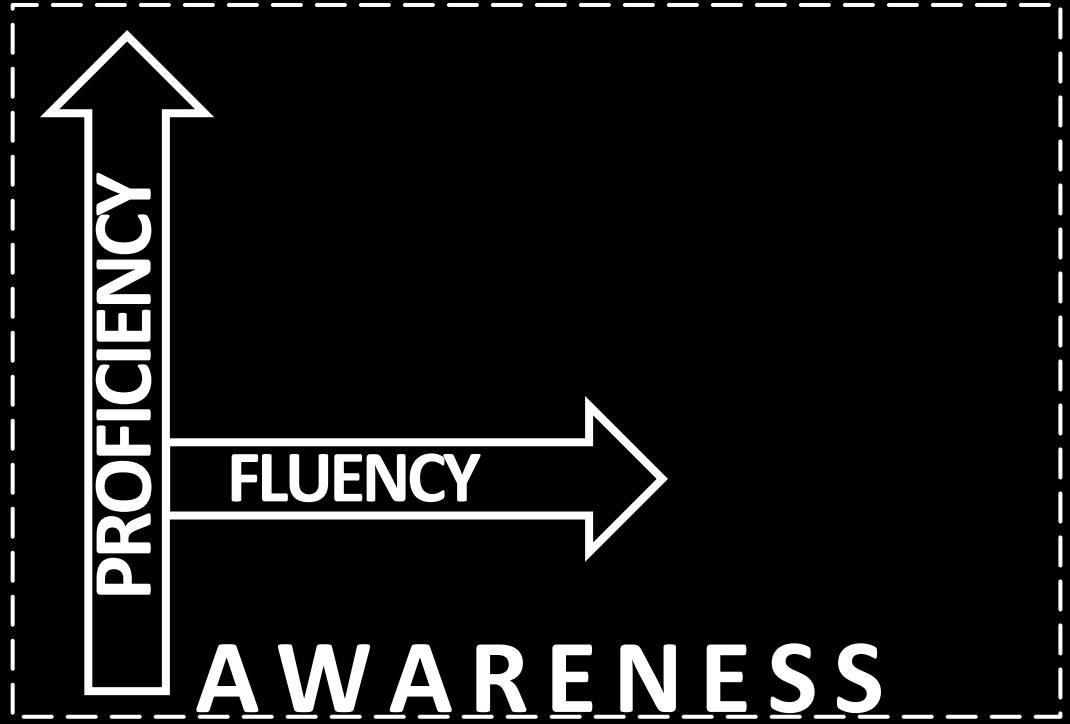
Native language

```
// Create side (0,0) – (3,4)
// Verify length
```

```
Side (0,0) – (3,4) length = 5
```

```
Side s = new Side(0,0,3,4);
Approvals.Verify(s + " length = " +s.Length);
```

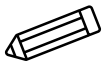
# Types of knowledge





3

# Benefits of Tests

1. Specification 
2. Feedback
3. Regression
4. Granularity

# Rules for Translating Test Scenarios

## *From the user perspective*

You are writing what the user does, not what the program does

~~*1.Get Shovel 2. Dig Hole 3. Place Pole…*~~

*setupFlagPole()*

## *Edit*

Improve on your 1st draft.
The better your English is, the better the code will be.
There will never be an easier time to refactor

~~*Ask to make a game for the category of TicTacToe*~~

*Create a TicTacToe game*

## *Verify Effect*

Make sure you have the correct outcome

*Place a 'X' at 1,2*

\+ *Check X is at 1,2*

## *Verify Cause*

Make sure it happened for the right reasons

\+ *Check 1,2 is blank*
*Place a 'X' at 1,2*
*Check X is at 1,2*

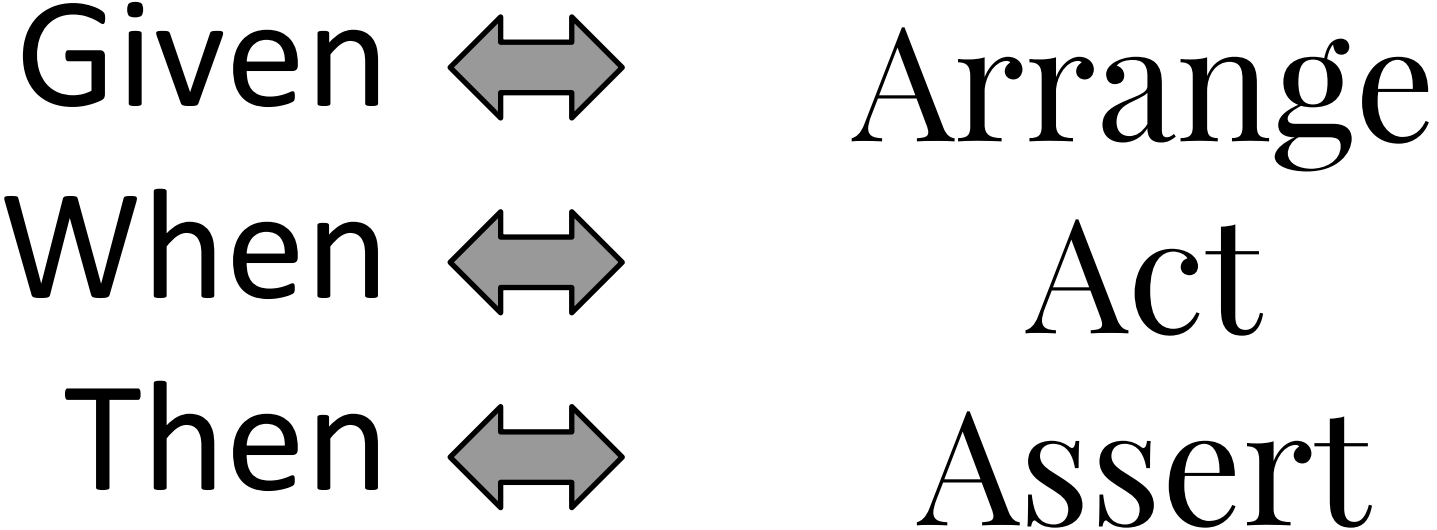
## *Complete*

The world begins and ends with your test. Make sure it has everything it needs

\+ *Create a board*
*Check 1,2 is blank*
*Place a 'X' at 1,2*
*Check X is at 1,2*

Arrange
Act
Assert

| | |
|---|---|
| Given | Arrange |
| When | Act |
| Then | Assert |

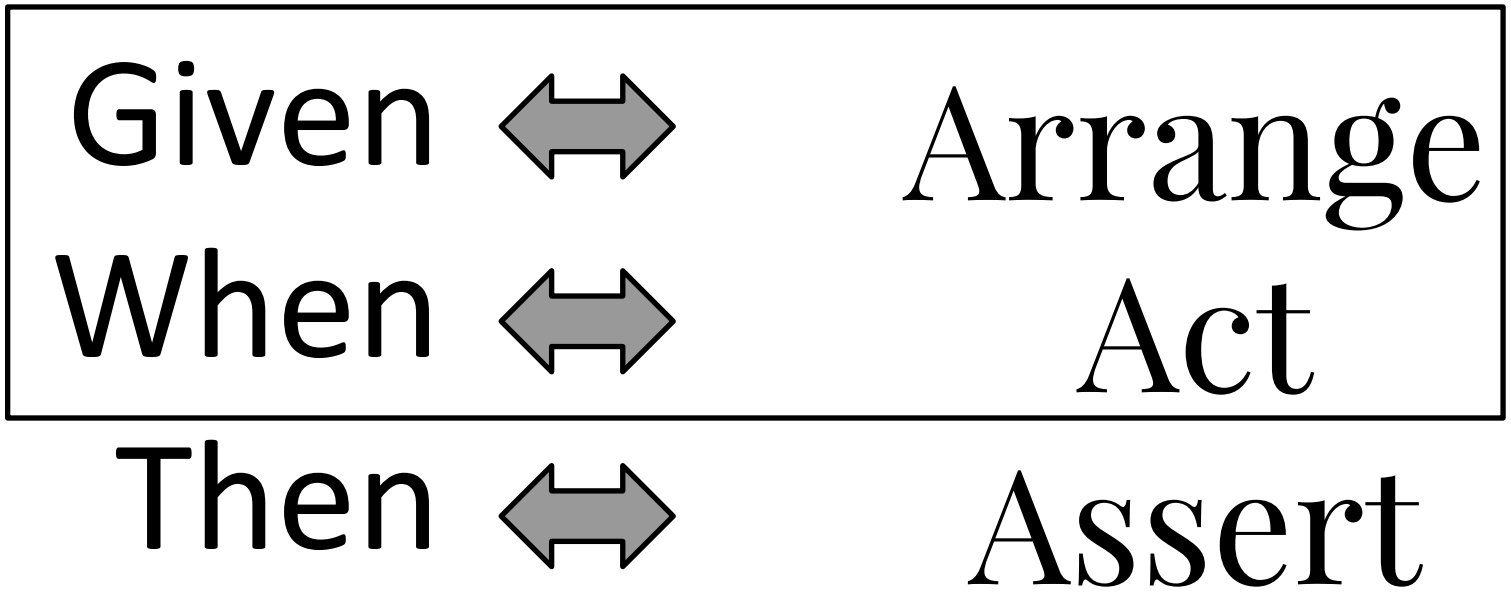

Given
When
Then
Arrange
Act
Assert

#1

```
TEST_CASE("Test Setup") {
    TrigMath math; // Arrange
    REQUIRE(math.DEG_TO_RAD == Approx(0.0174532925)); // Assert
}
```

#2

```
TEST_CASE("Test Function") {
    auto result = pow(2, 3); // Act
    REQUIRE(result == Approx(8)); // Assert
}
```


Given
When
Then
Arrange
Act
Assert

# Do
# Verify

# 2.
# Practice – Translating Test Scenarios

1. Minesweeper
2. [redacted]
3. [redacted]
4. [redacted]
5. [redacted]
6. [redacted]
7. [redacted]
8. [redacted]
9. family tree
10. [redacted]
11. Anagrams
12. Bank account
13. [redacted]
14. Hangman

# 3.
# Practice – Translating English to Code

# Micro-API Design

# Rules for Creating Code

## *Use your imagination*

Write the code you ***want*** to exist, regardless of what currently does

~~*array.isEmpty() ? null : array.get(0)*~~

*array.first()*

✓

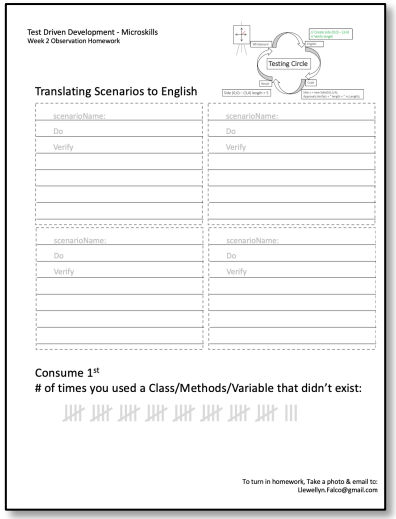
## *Evaluate the consequences*

Ask yourself “what are the resulting ***classes*** and ***methods*** from this implementation?”

# 4.
# Practice – Triangles

1. A side has a distance
2. A side has endpoints
3. 3 points
4. 3 sides
5. Perimeter
6. Get Sides touching a point
7. Get sides opposite a point
8. The angle of 2 sides touching a point
9. 3 angles
10. Right Triangle

# Homework


Test Driven Development - Microskills
Week 2 Observation Homework

Testing Circle

Translating Scenarios to English

scenarioName:
Do
Verify

scenarioName:
Do
Verify

scenarioName:
Do
Verify

scenarioName:
Do
Verify

Consume 1st
# of times you used a Class/Methods/Variable that didn't exist:

To turn in homework, Take a photo & email to:
Llewellyn.Falco@gmail.com

*Write 4 scenarios*

*Count # times you used a class/method/variable that didn't exist*

*thanks!*

# Any questions?

Please connect through LinkedIn & Twitter

@LlewellynFalco

Llewellyn.Falco@gmail.com