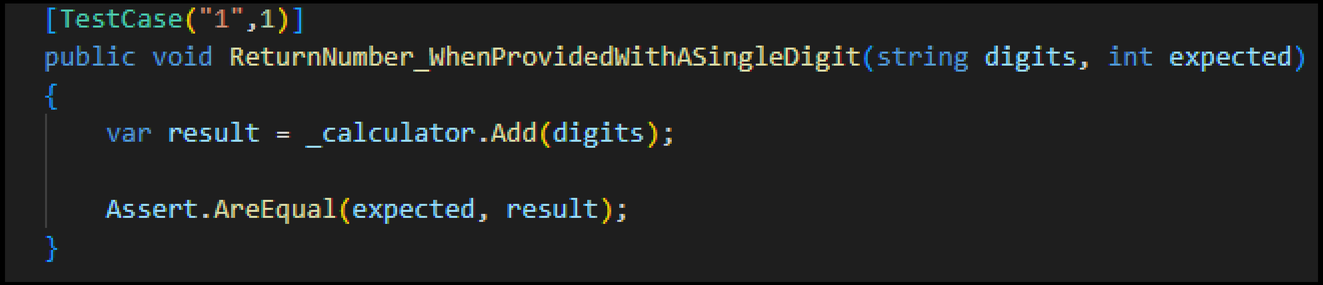
Transformation Priority Premise

- What are the three steps for driving TDD?
- What is TPP?
- Reaching a roadblock in TDD
- Case study: Word wrap kata
- Summary: Benefits and drawbacks

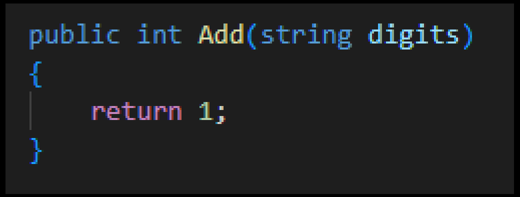
TPP – Defining Obvious Implementation

- Three steps in moving TDD code forward:
  1. Fake it
  2. Obvious implementation
  3. Triangulation

Fake Implementation

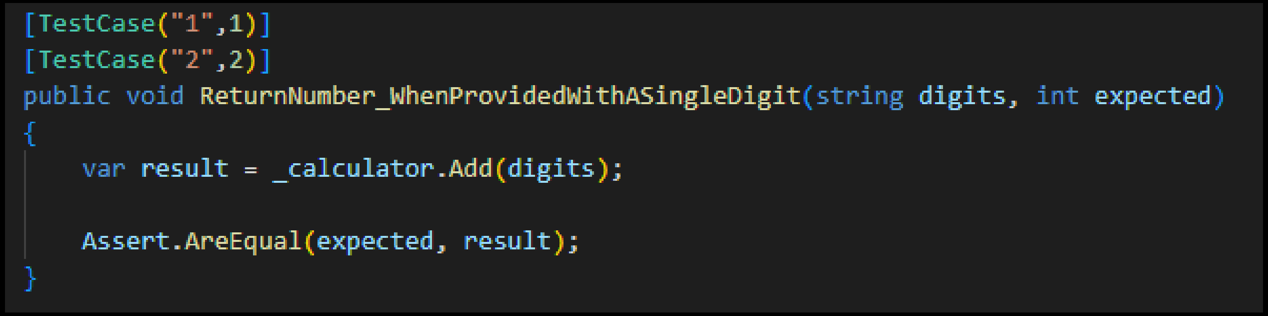


```
[TestCase("1",1)]
public void ReturnNumber_WhenProvidedWithASingleDigit(string digits, int expected)
{
    var result = _calculator.Add(digits);

    Assert.AreEqual(expected, result);
}
```

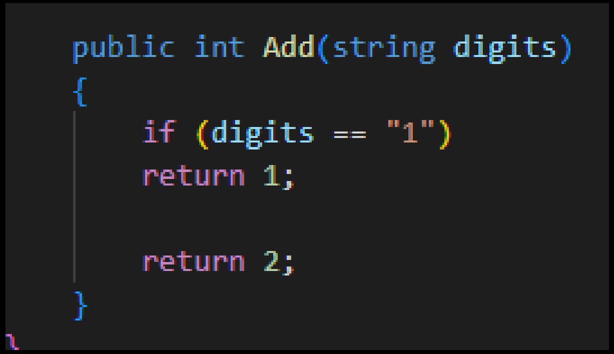


```
public int Add(string digits)
{
    return 1;
}
```

Obvious Implementation

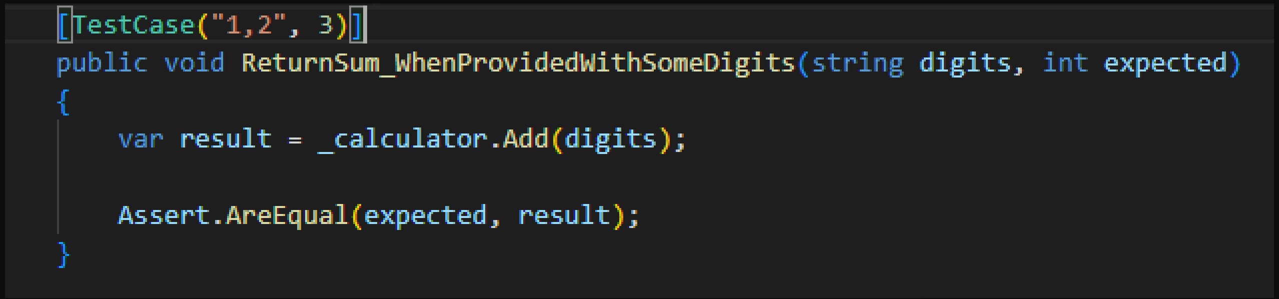


```
[TestCase("1",1)]
[TestCase("2",2)]
public void ReturnNumber_WhenProvidedWithASingleDigit(string digits, int expected)
{
    var result = _calculator.Add(digits);

    Assert.AreEqual(expected, result);
}
```

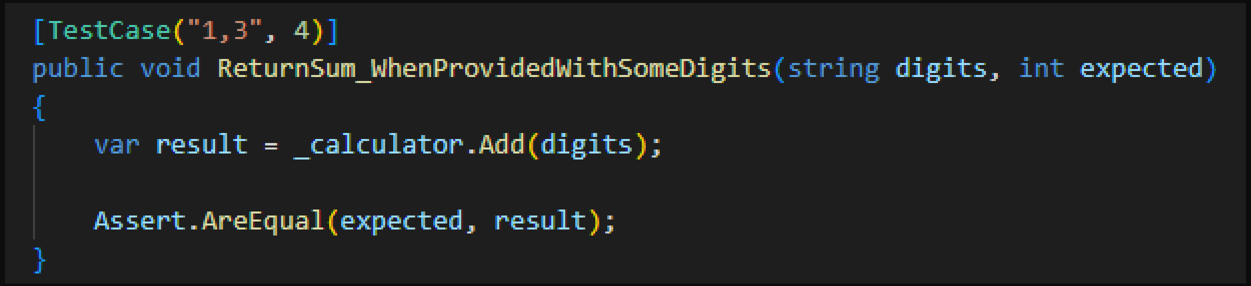


```
public int Add(string digits)
{
    if (digits == "1")
    return 1;

    return 2;
}
```

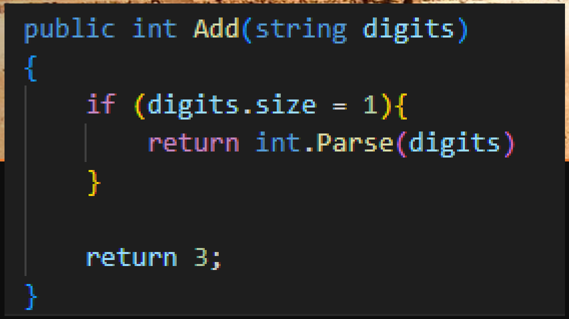
Triangulation



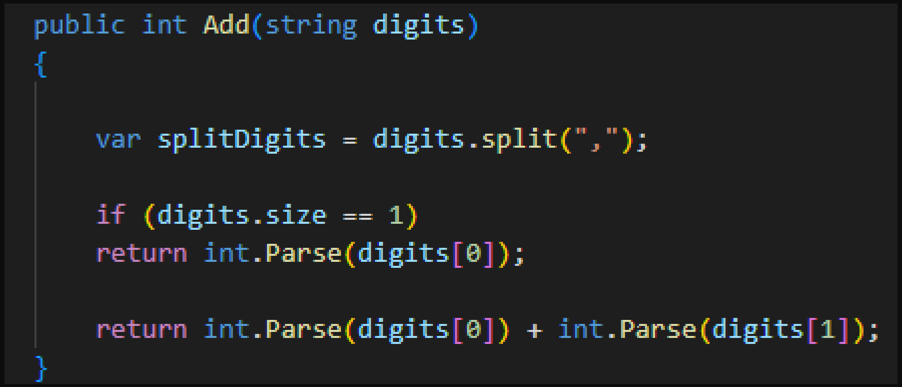
```
[TestCase("1,2", 3)]
public void ReturnSum_WhenProvidedWithSomeDigits(string digits, int expected)
{
    var result = _calculator.Add(digits);

    Assert.AreEqual(expected, result);
}
```



```
[TestCase("1,3", 4)]
public void ReturnSum_WhenProvidedWithSomeDigits(string digits, int expected)
{
    var result = _calculator.Add(digits);

    Assert.AreEqual(expected, result);
}
```



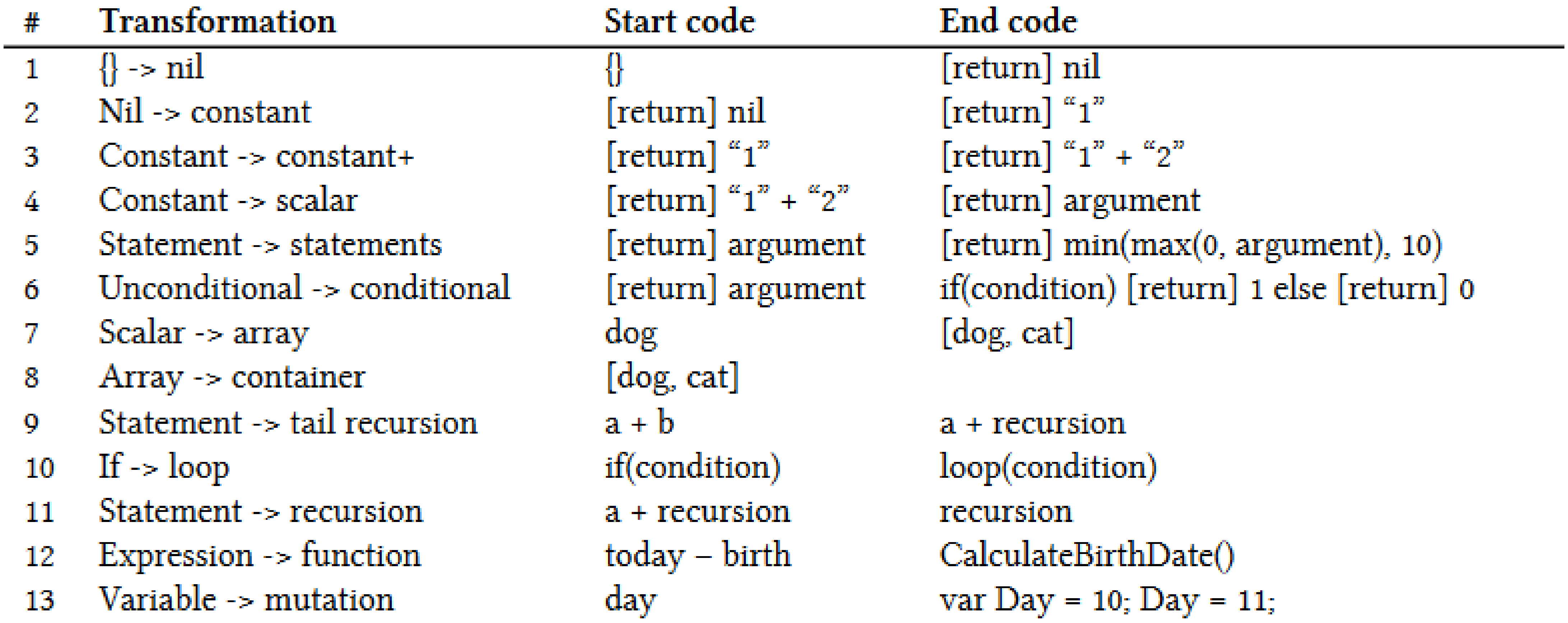
```
public int Add(string digits)
{
    if (digits.size = 1){
        return int.Parse(digits)
    }

    return 3;
}
```



```
public int Add(string digits)
{

    var splitDigits = digits.split(",");

    if (digits.size == 1)
    return int.Parse(digits[0]);

    return int.Parse(digits[0]) + int.Parse(digits[1]);
}
```

TPP – Defining the Obvious Implementation

TPP table



#	Transformation	Start code	End code
1	{} -> nil	{}	[return] nil
2	Nil -> constant	[return] nil	[return] “1”
3	Constant -> constant+	[return] “1”	[return] “1” + “2”
4	Constant -> scalar	[return] “1” + “2”	[return] argument
5	Statement -> statements	[return] argument	[return] min(max(0, argument), 10)
6	Unconditional -> conditional	[return] argument	if(condition) [return] 1 else [return] 0
7	Scalar -> array	dog	[dog, cat]
8	Array -> container	[dog, cat]	
9	Statement -> tail recursion	a + b	a + recursion
10	If -> loop	if(condition)	loop(condition)
11	Statement -> recursion	a + recursion	recursion
12	Expression -> function	today – birth	CalculateBirthDate()
13	Variable -> mutation	day	var Day = 10; Day = 11;

What Is a Transformation

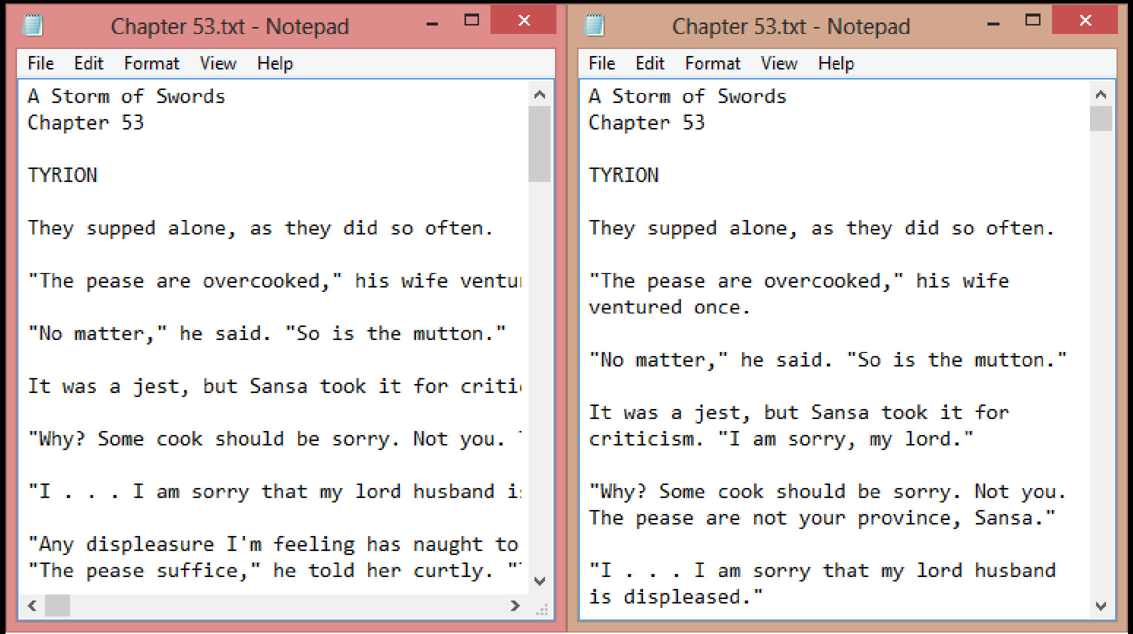
- Uncle Bob suggests it is a counterpart to refactorings
- Simple operations that change the behavior of code

Solving the Impasse Problem

- Writing tests and implementations for a problem without TPP can make you reach a test that is impossible to make pass without rewriting the whole algorithm

Case Study: Word Wrap Kata

- Write a method that takes in a string and a colum number.
- Insert newlines such that each line is never longer than the column number, try to break at word boundaries



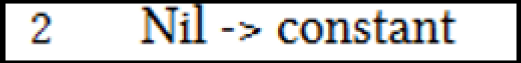
```
A Storm of Swords
Chapter 53

TYRION

They supped alone, as they did so often.

"The pease are overcooked," his wife ventu

"No matter," he said. "So is the mutton."

It was a jest, but Sansa took it for criti

"Why? Some cook should be sorry. Not you. "

"I . . . I am sorry that my lord husband i

"Any displeasure I'm feeling has naught to
"The pease suffice," he told her curtly. "
```

No Word Wrap

```
A Storm of Swords
Chapter 53

TYRION

They supped alone, as they did so often.

"The pease are overcooked," his wife
ventured once.

"No matter," he said. "So is the mutton."

It was a jest, but Sansa took it for
criticism. "I am sorry, my lord."

"Why? Some cook should be sorry. Not you.
The pease are not your province, Sansa."

"I . . . I am sorry that my lord husband
is displeased."
```

Word Wrap

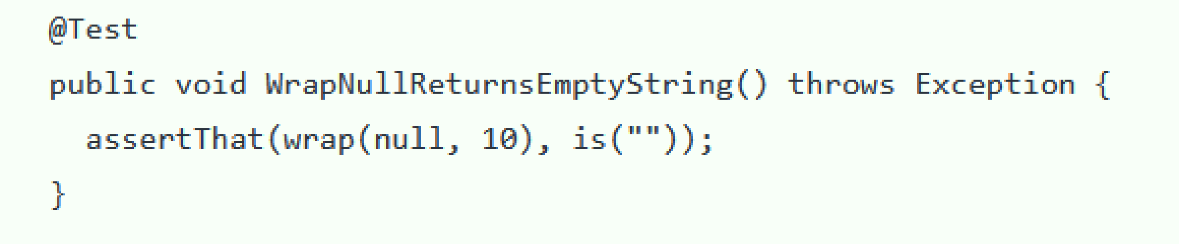
Null String



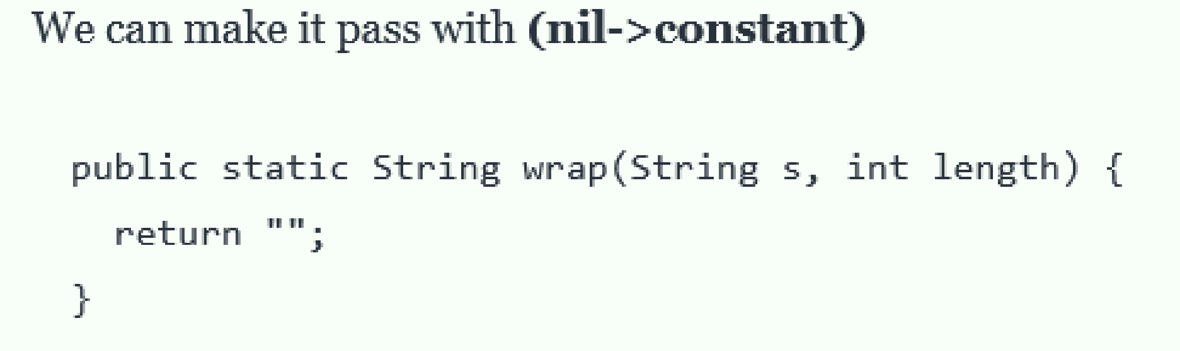
2 Nil -> constant

Implementation

Test



```
@Test
public void WrapNullReturnsEmptyString() throws Exception {
  assertThat(wrap(null, 10), is(""));
}
```

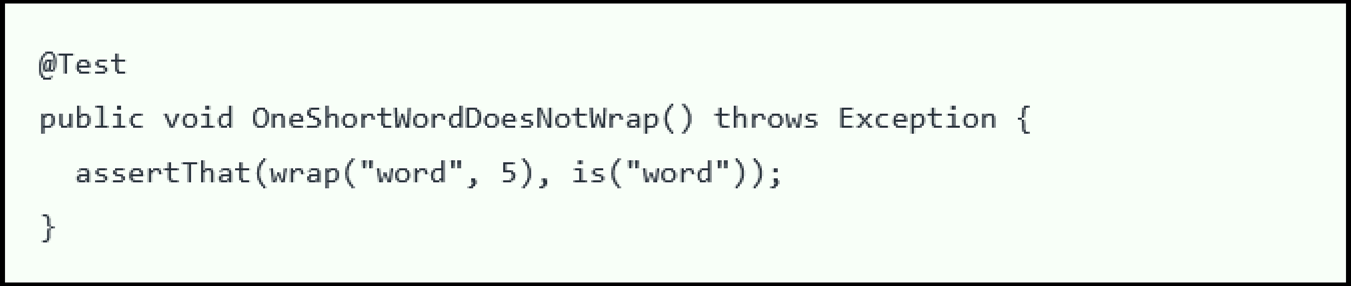


We can make it pass with **(nil->constant)**

```
public static String wrap(String s, int length) {
  return "";
}
```

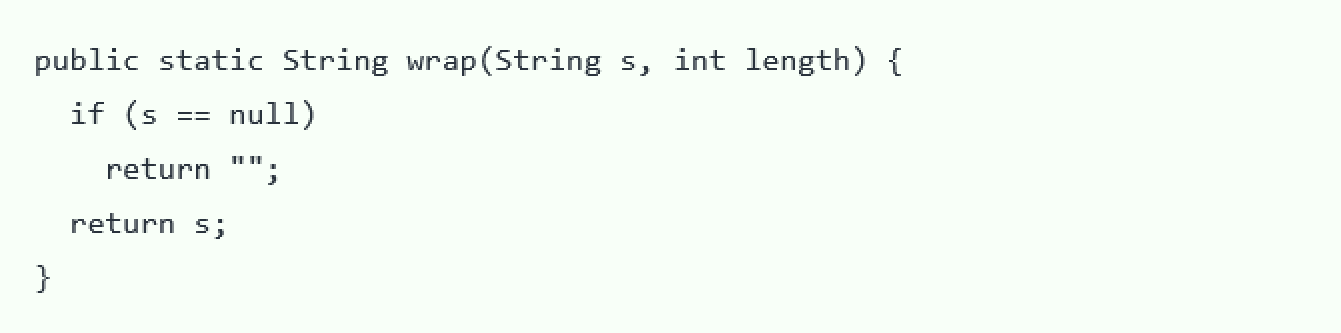
Short Word

Test

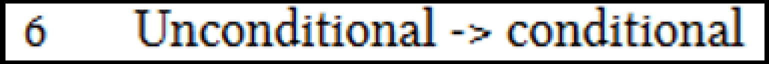


```
@Test
public void OneShortWordDoesNotWrap() throws Exception {
  assertThat(wrap("word", 5), is("word"));
}
```

Implementation



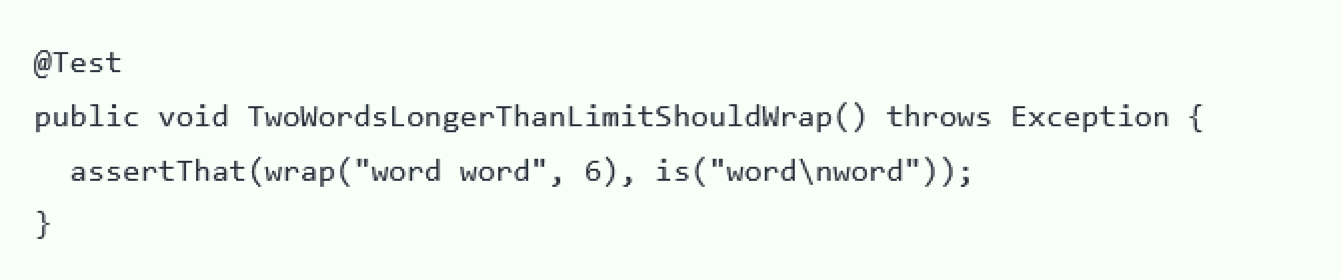
```
public static String wrap(String s, int length) {
  if (s == null)
    return "";
  return s;
}
```



6 Unconditional -> conditional

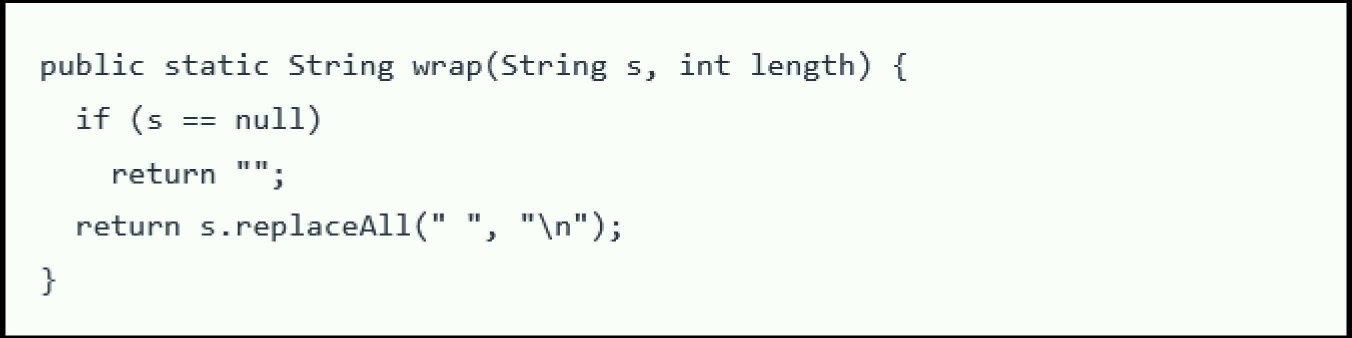
New Test

Test

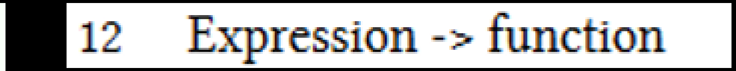


```
@Test
public void TwoWordsLongerThanLimitShouldWrap() throws Exception {
  assertThat(wrap("word word", 6), is("word\nword"));
}
```

Implementation



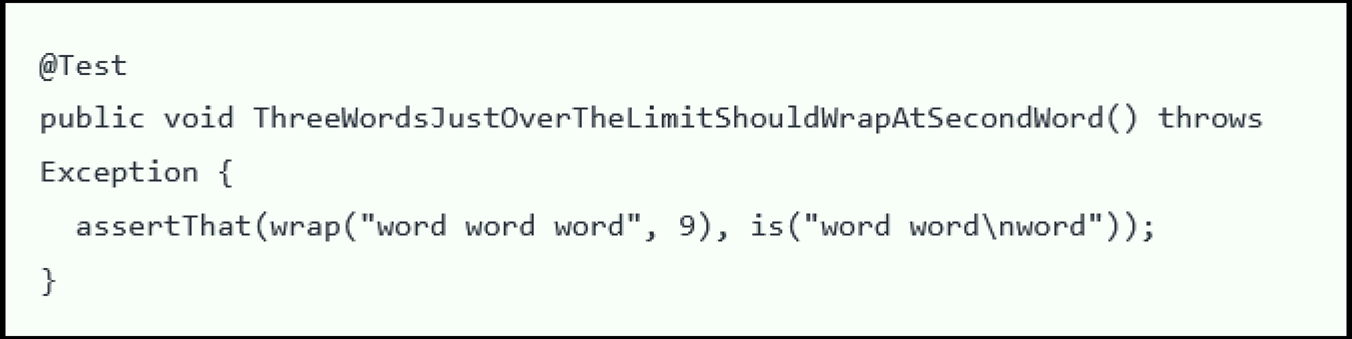
```
public static String wrap(String s, int length) {
  if (s == null)
    return "";
  return s.replaceAll(" ", "\n");
}
```



12 Expression -> function

Reaching the Impasse

Test

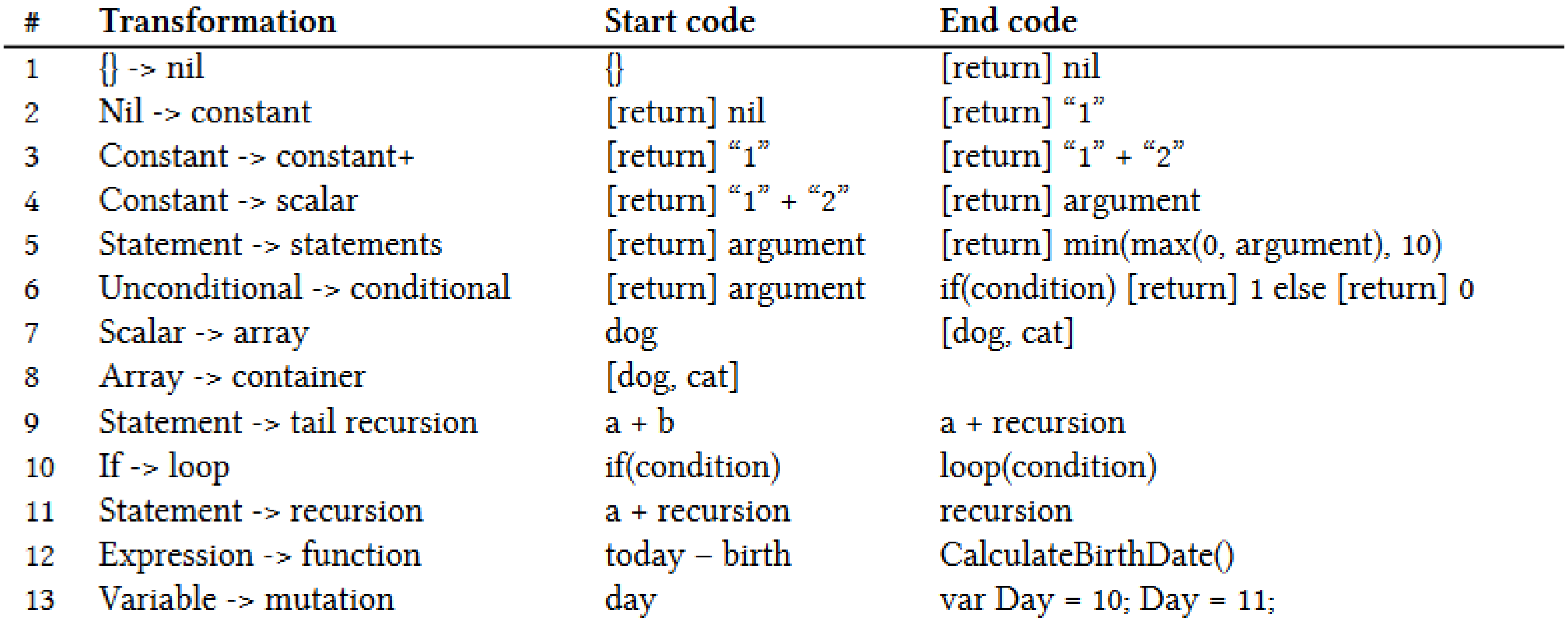


```
@Test
public void ThreeWordsJustOverTheLimitShouldWrapAtSecondWord() throws
Exception {
  assertThat(wrap("word word word", 9), is("word word\nword"));
}
```

Implementation

?

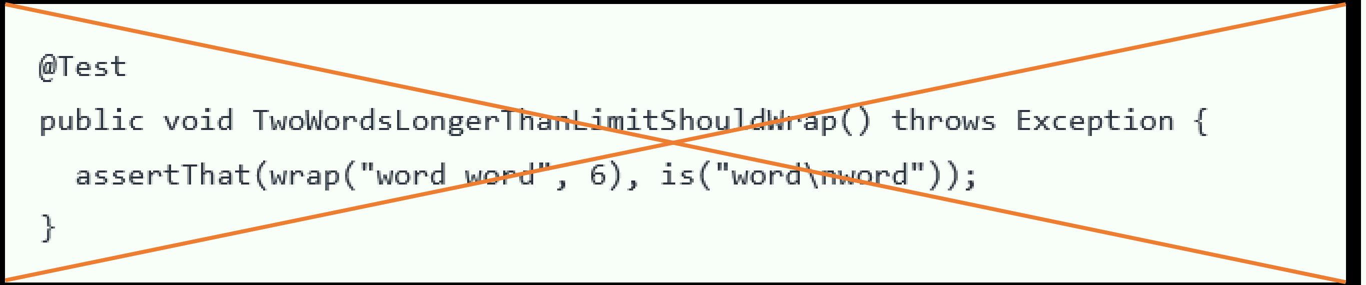
TPP table



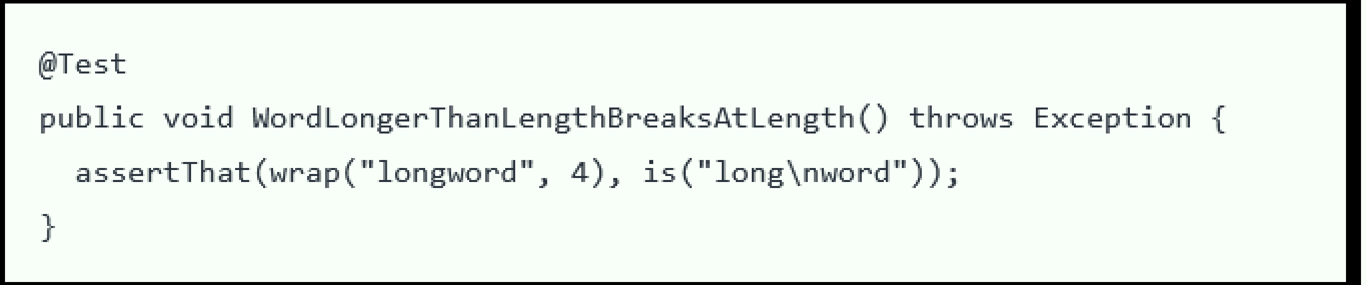
#	Transformation	Start code	End code
1	{} -> nil	{}	[return] nil
2	Nil -> constant	[return] nil	[return] “1”
3	Constant -> constant+	[return] “1”	[return] “1” + “2”
4	Constant -> scalar	[return] “1” + “2”	[return] argument
5	Statement -> statements	[return] argument	[return] min(max(0, argument), 10)
6	Unconditional -> conditional	[return] argument	if(condition) [return] 1 else [return] 0
7	Scalar -> array	dog	[dog, cat]
8	Array -> container	[dog, cat]	
9	Statement -> tail recursion	a + b	a + recursion
10	If -> loop	if(condition)	loop(condition)
11	Statement -> recursion	a + recursion	recursion
12	Expression -> function	today – birth	CalculateBirthDate()
13	Variable -> mutation	day	var Day = 10; Day = 11;

Let’s Go Back

Test

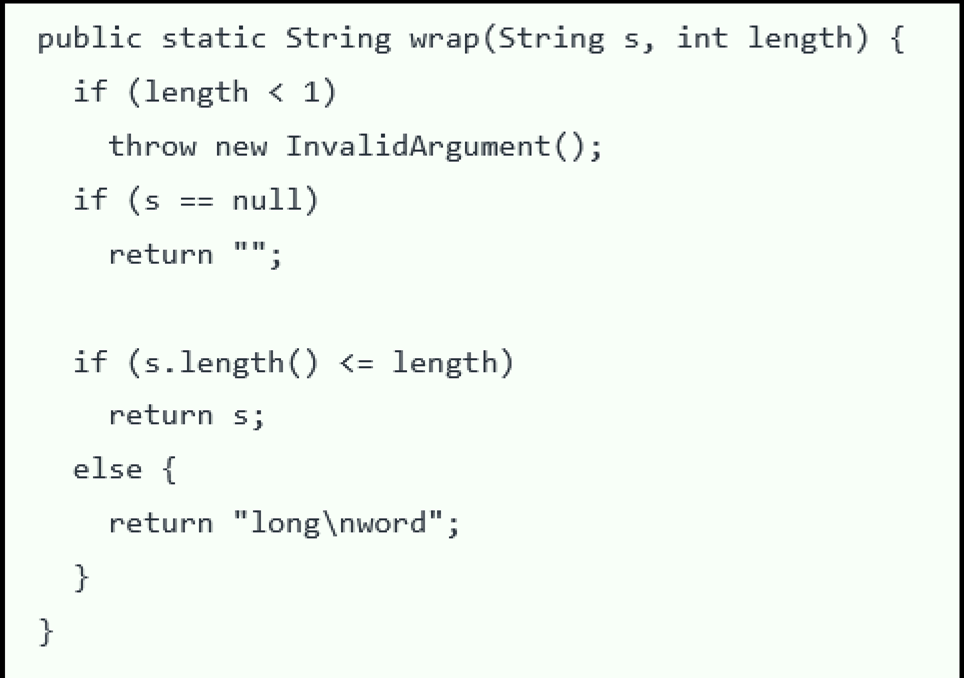


```java
@Test
public void TwoWordsLongerThanLimitShouldWrap() throws Exception {
  assertThat(wrap("word word", 6), is("word\nword"));
}
```



```java
@Test
public void WordLongerThanLengthBreaksAtLength() throws Exception {
  assertThat(wrap("longword", 4), is("long\nword"));
}
```

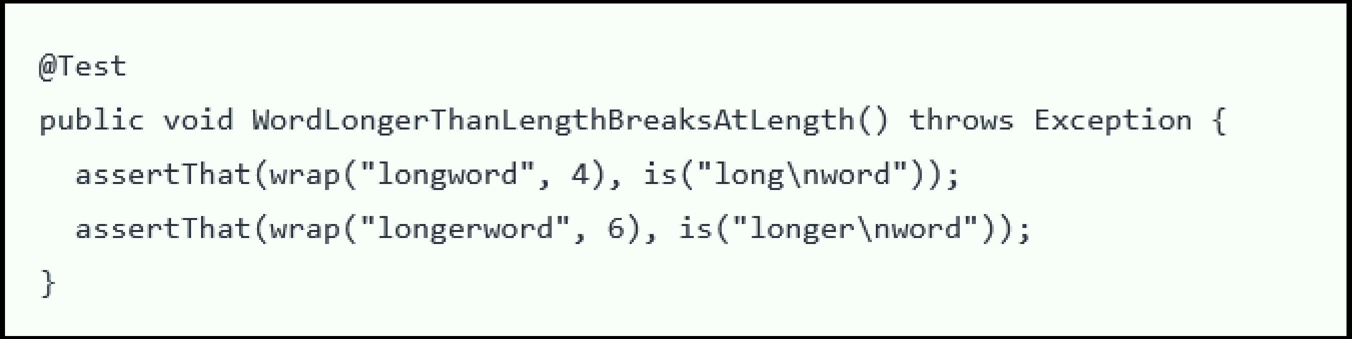
Implementation



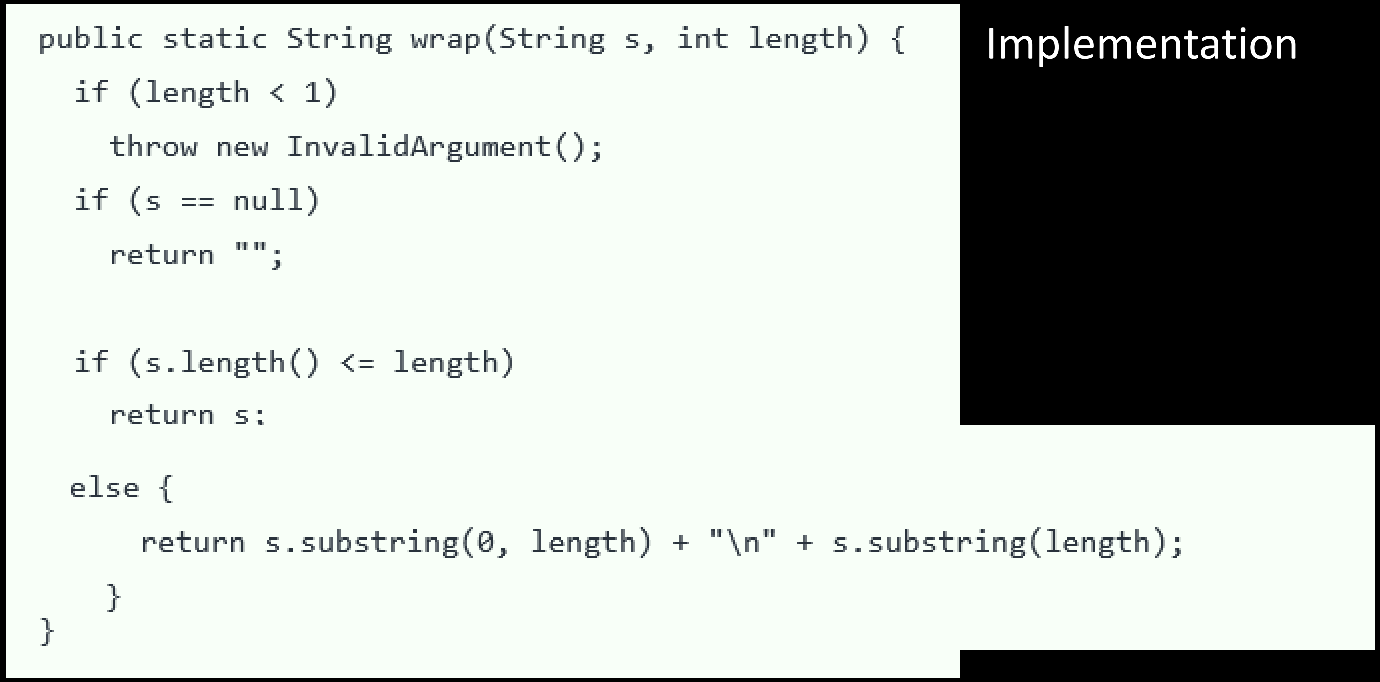
```java
public static String wrap(String s, int length) {
  if (length < 1)
    throw new InvalidArgument();
  if (s == null)
    return "";

  if (s.length() <= length)
    return s;
  else {
    return "long\nword";
  }
}
```

Let’s Triangulate

Test



```
@Test
public void WordLongerThanLengthBreaksAtLength() throws Exception {
  assertThat(wrap("longword", 4), is("long\nword"));
  assertThat(wrap("longerword", 6), is("longer\nword"));
}
```



```
public static String wrap(String s, int length) {
  if (length < 1)
    throw new InvalidArgument();
  if (s == null)
    return "";

  if (s.length() <= length)
    return s:
  else {
     return s.substring(0, length) + "\n" + s.substring(length);
    }
}
```

Implementation

Case Study Summary

- Makes it clear how TPP helps us find the obvious implementation
- Shows how we are less likely to reach an impasse/roadblock
- We are changing behavior in smaller steps
- Demonstrates TPP role in the three TDD steps for moving code forward
- Shows how TPP moves us from a specific to a more general form
- Full example from Uncle Bob’s blog: https://blog.cleancoder.com/uncle-bob/2013/05/27/TheTransformationPriorityPremise.html

Benefits of the TPP process

- When passing a test, prefer higher priority transformations.
- When writing a new test, choose one that can be passed with higher priority transformations.
- When an implementation seems to require a low priority transformation, backtrack to see if there is a simpler test to pass

Drawbacks of TPP

- Not a complete list of transformation
- Is a transformation the correct one?
  - What are the criteria?
- What decides the order/priority
  - How can we decide how complex a transformation is?

Questions?

Sources

- Uncle Bob’s blog - https://blog.cleancoder.com/uncle-bob/2013/05/27/TheTransformationPriorityPremise.html
- Agile Technical Practices Distilled (2019) – Santos, Consolaro, Di Gioia