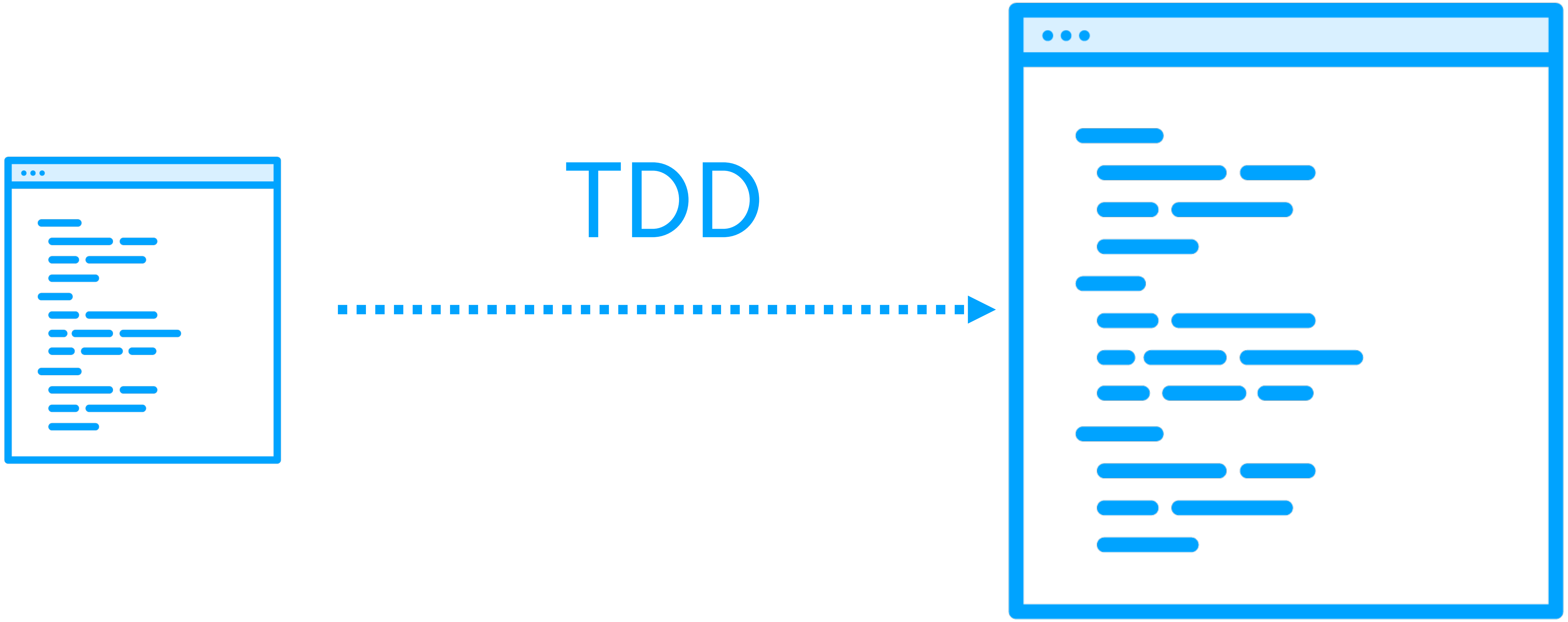
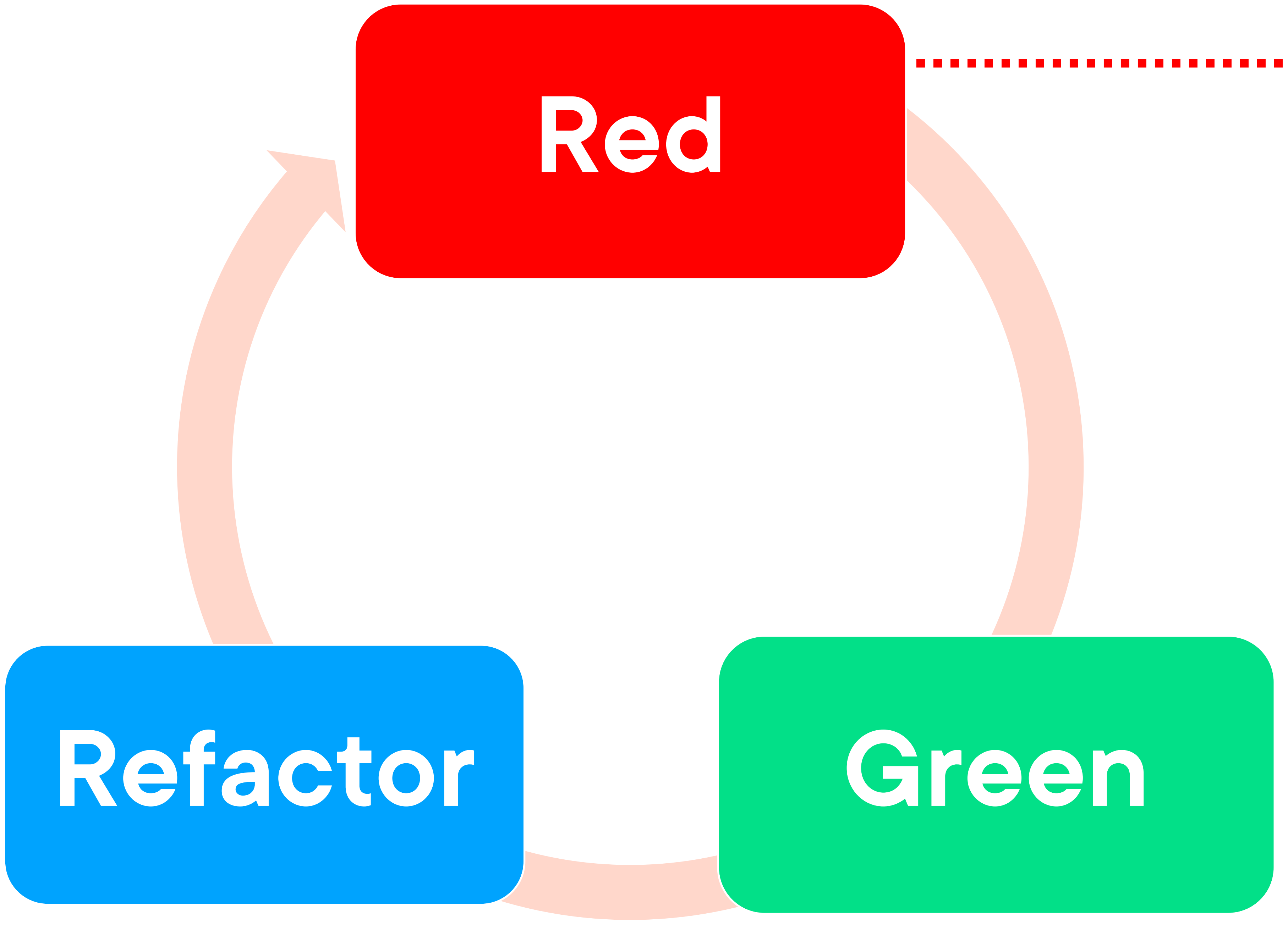
Starting Small with TDD

Andrejs Doronins
Software Developer in Test



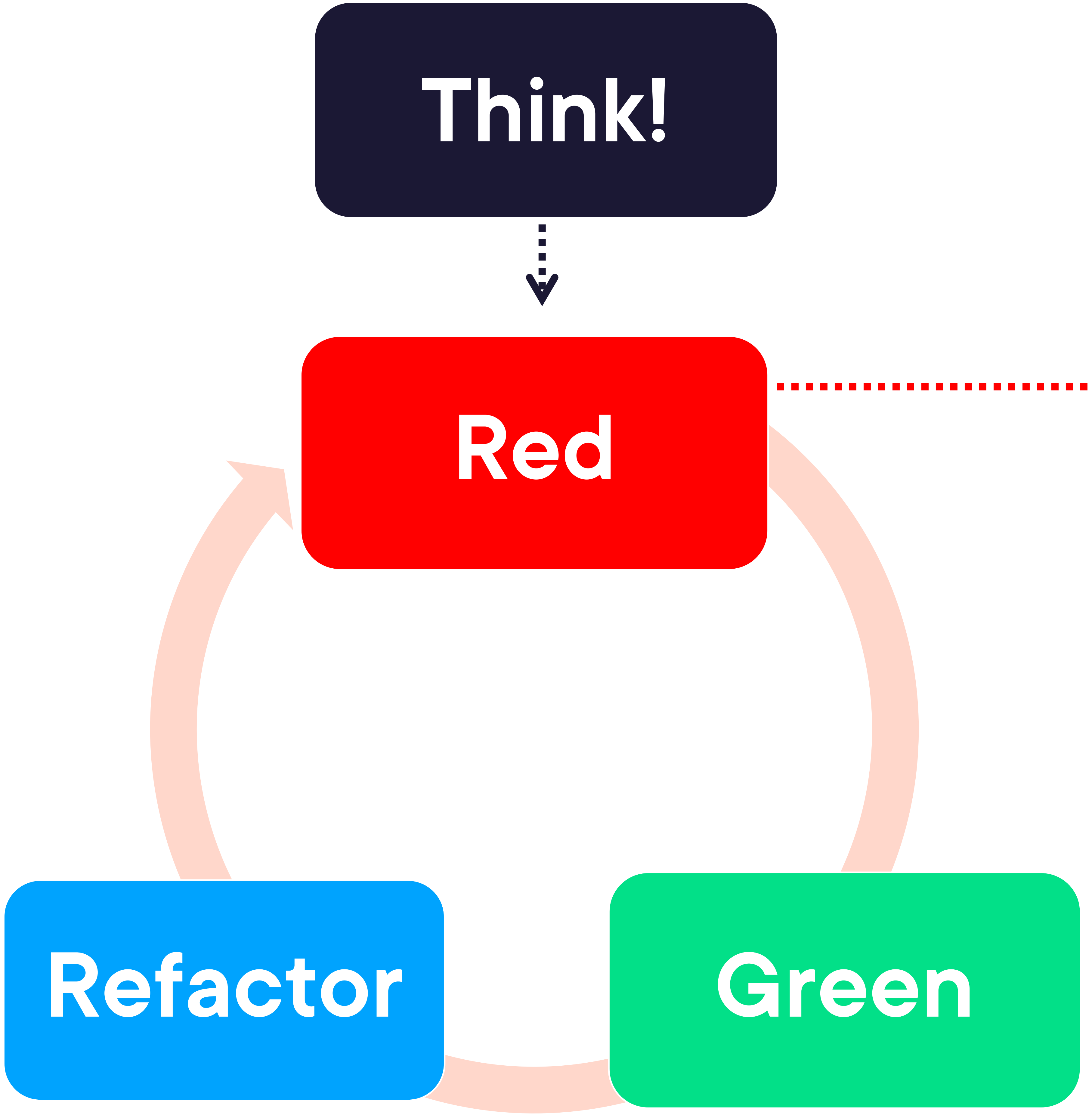
TDD



Write a failing test.
(compilation failure
counts)

Make it pass.
Write the simplest
possible thing

Clean it up!



Think!

Red

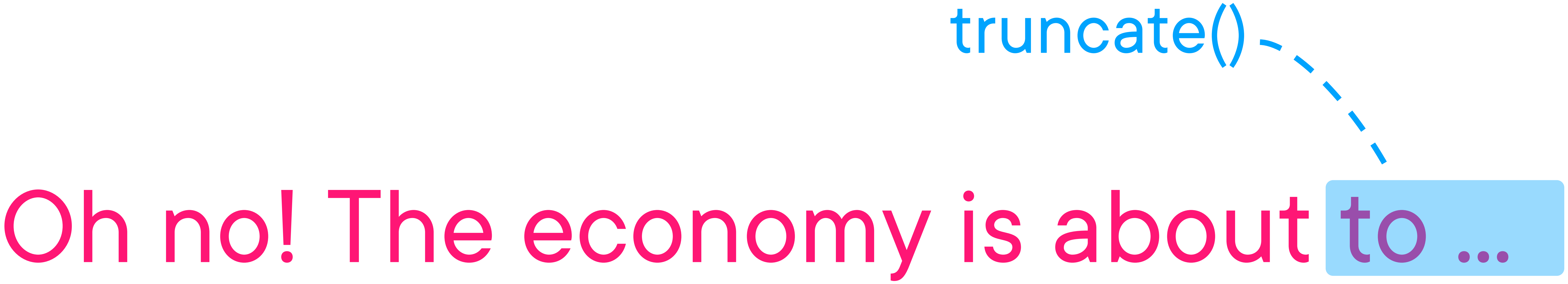
Write a failing test.
(compilation failure
counts)

Green

Make it pass.
Write the simplest
possible thing

Refactor

Clean it up!



truncate()

Oh no! The economy is about to ...

Get to green quickly

Fake it:

- return a hard-coded value
  - “”, 0, new Thing()

Obvious real implementation:

- return a + b

ZOMBIES

Zero

One

Many

Boundary behavior

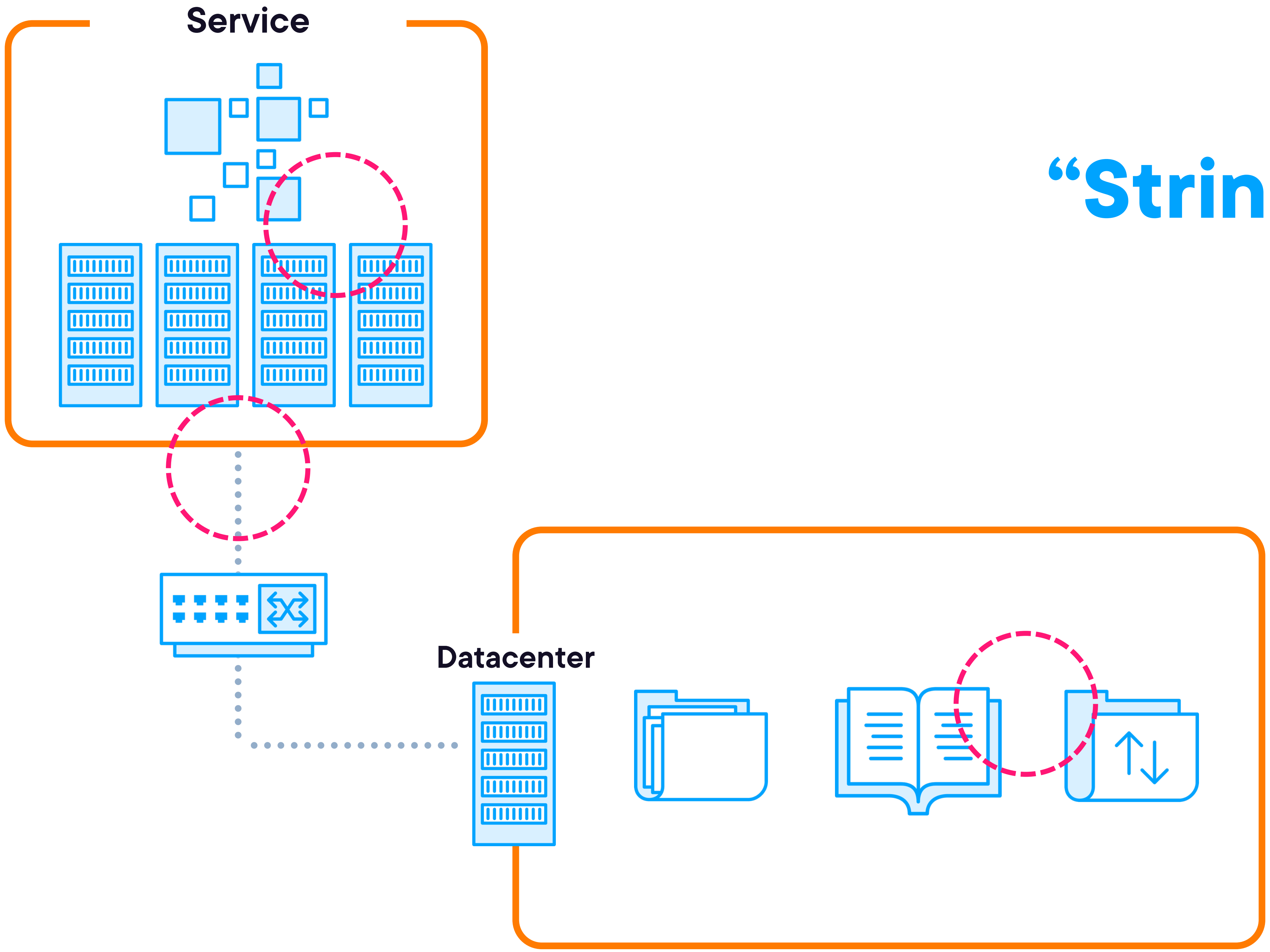
Interface definition

Exceptional behavior

Simple scenarios - simple solutions

(Credit: James Grenning)

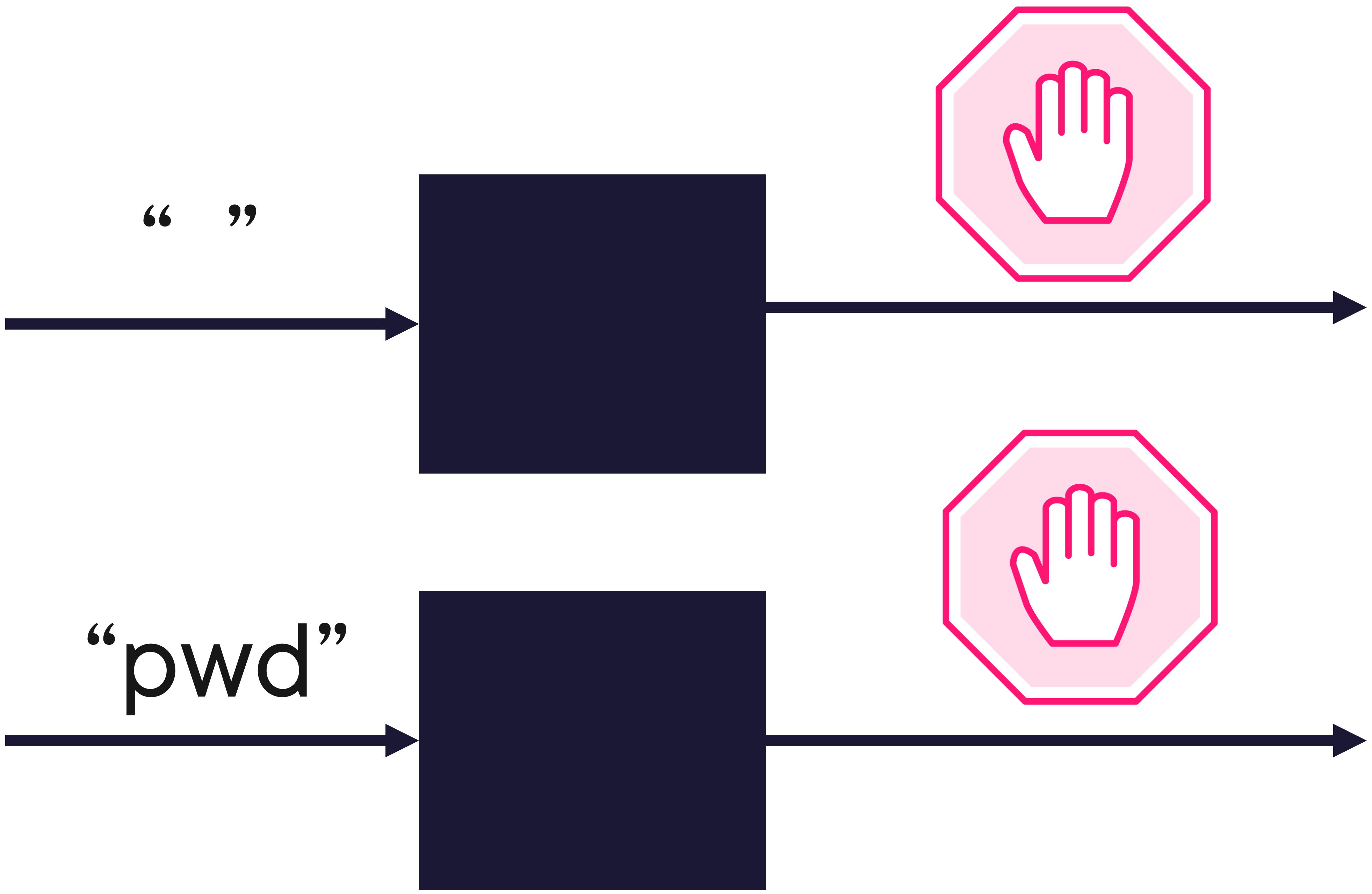
Boundaries



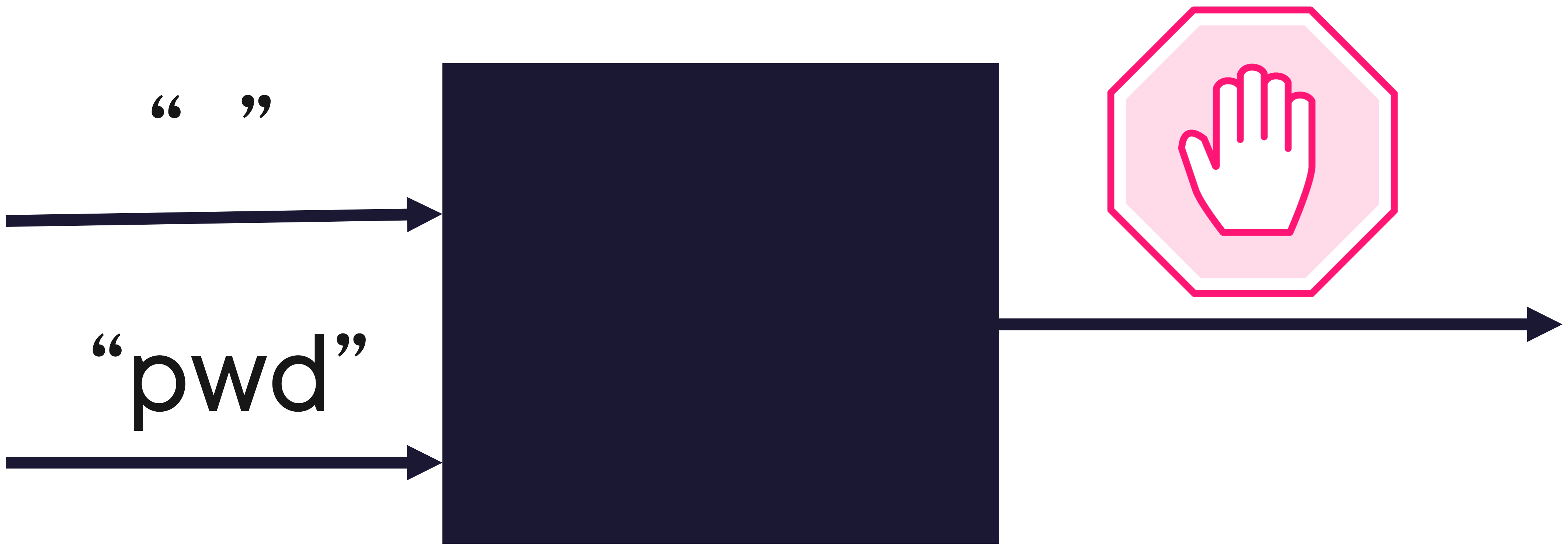
0...10

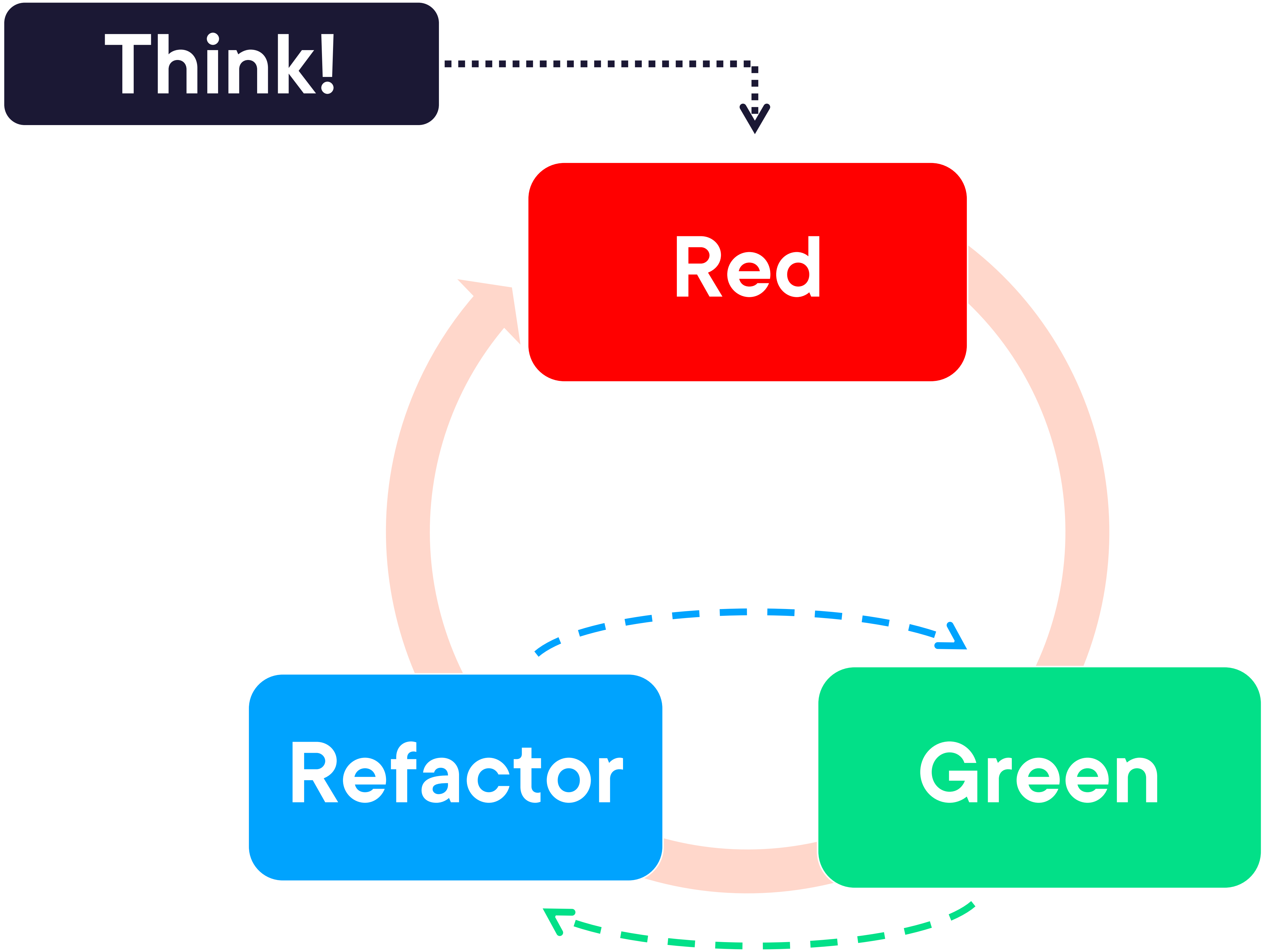
"String length"

When to Parameterize?



When to Parameterize?




Think!
Red
Green
Refactor

ZOMBIES

Zero

One

Many

Boundary behavior

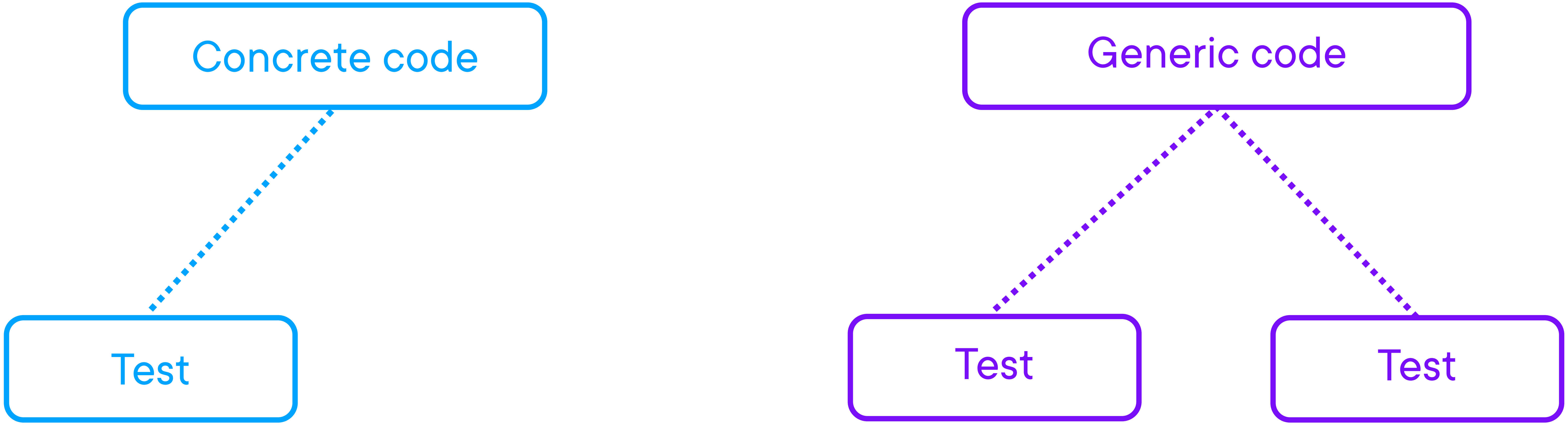
Interface definition

Exceptional behavior

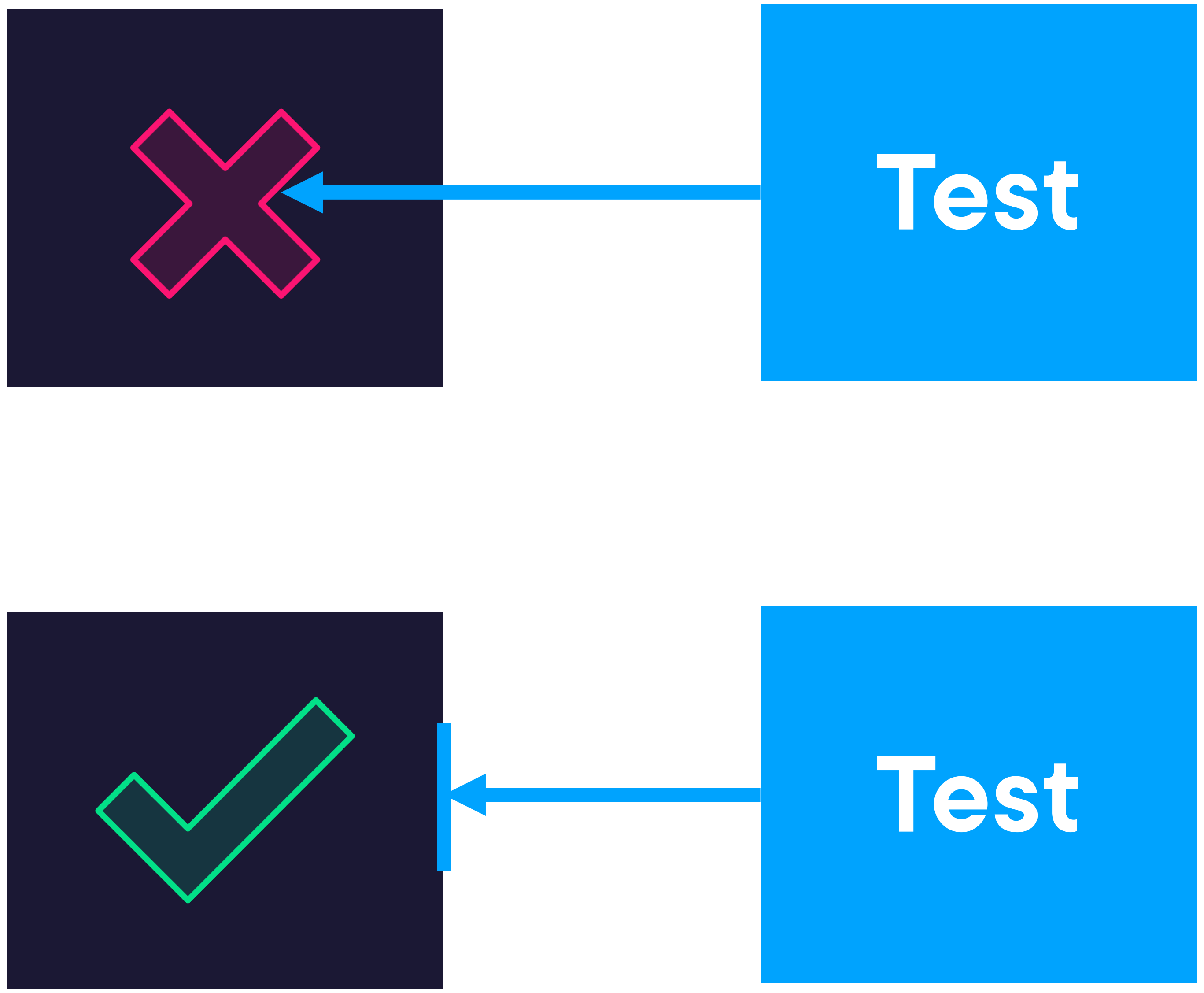
Simple scenarios - simple solutions

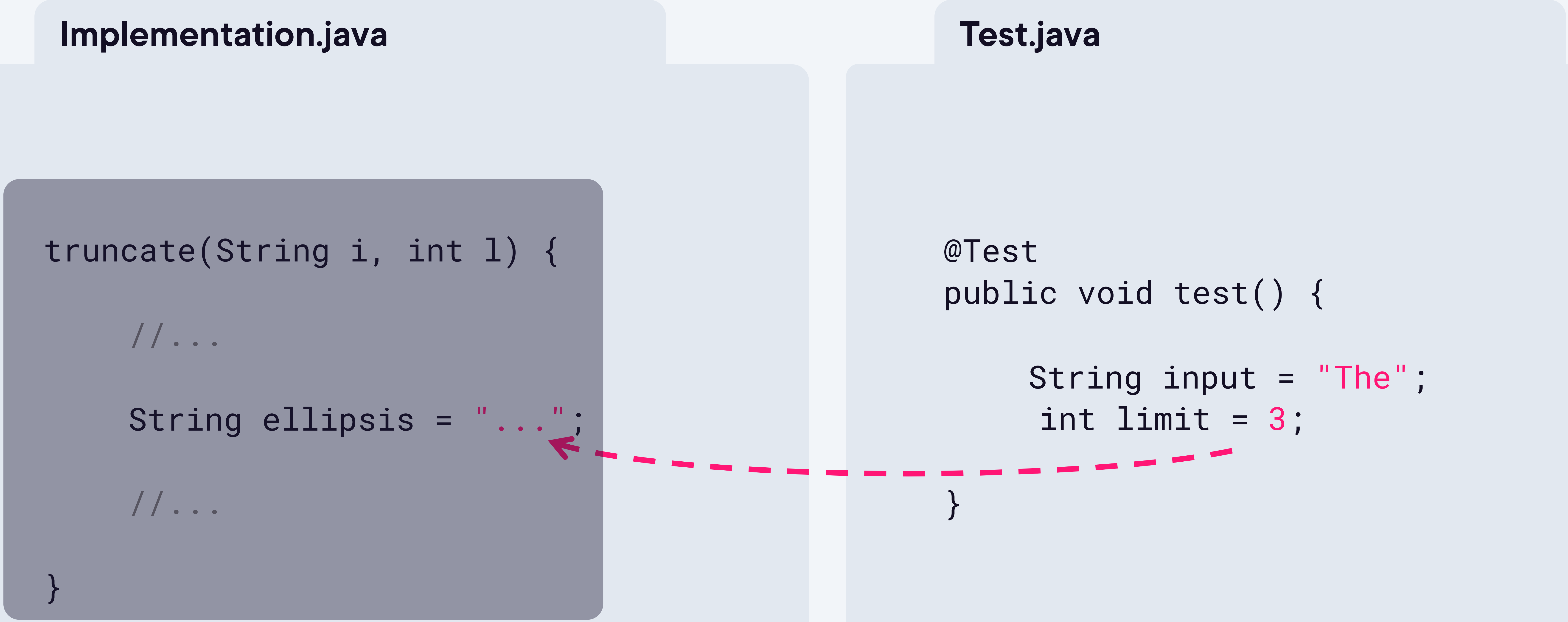
(Credit: James Grenning)

Triangulation



Testing the Internals or the Interface?





Implementation.java

```
truncate(String i, int l) {

    //...

    String ellipsis = "...";

    //...

}
```

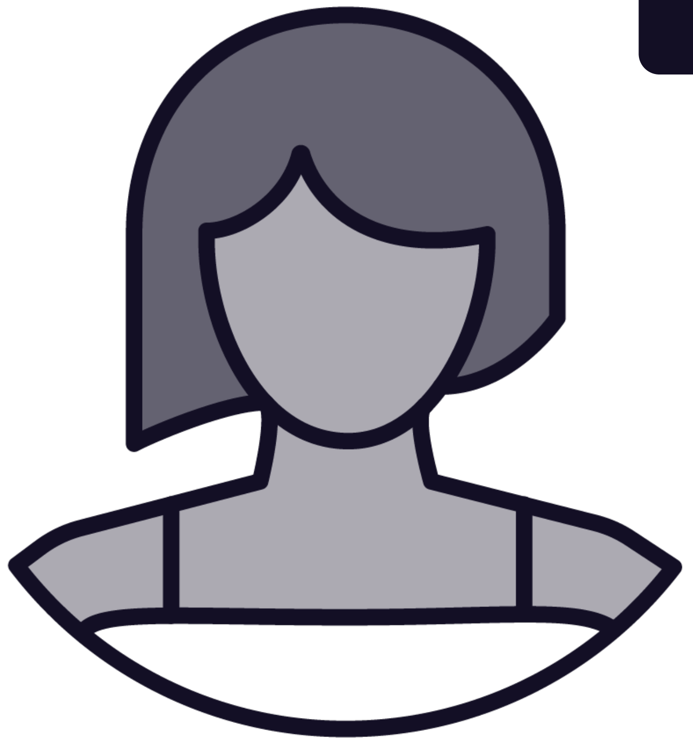
Test.java

```
@Test
public void test() {

    String input = "The";
    int limit = 3;

}
```

Change ... to >>



```
String truncate(String input, int limit) {
     return input.substring(0, limit) + "...";
 }
```

```
String truncate(String input, int limit) {
    if (input == null) { ... }

    if (input.length() <= limit) { ... }

    return input.substring(0, limit) + "...";
}
```

```
   String truncate(String input, int limit) {
        if (input == null) { ... }
        if (limit < 1) { ... }

        String ellipsis = "...";
        if (inputTooShort(input, limit, ellipsis)) {
            return input;
        }

        return input.substring(0, limit) + ellipsis;
}
```

Summary

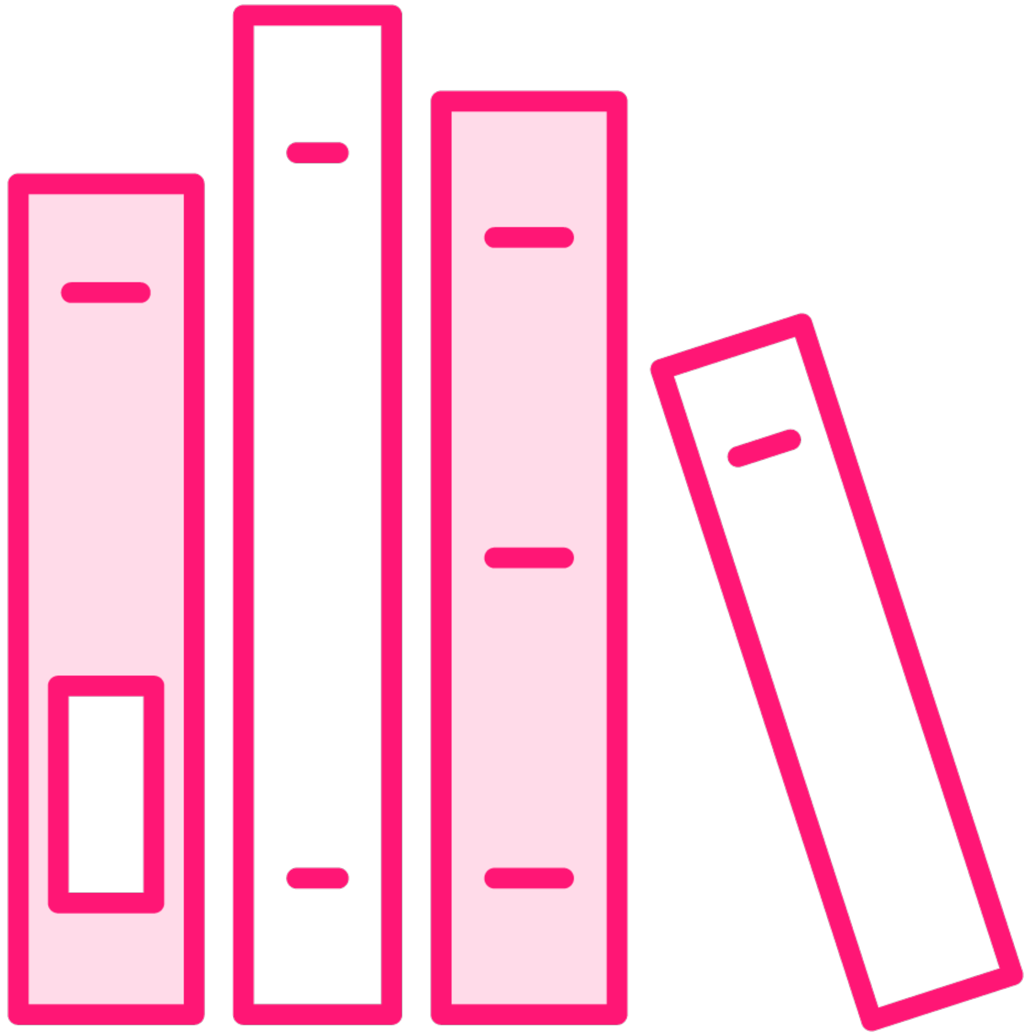
Advantages and criticisms of TDD

RGR:

- Think before jumping in
- Write a failing test - make it pass quickly
- Refactor both the production and test code
- Triangulation - specific tests, general implementation

Small steps!

Testing Mnemonics



ZOMBIES

FIRST

BICEP

CORRECT