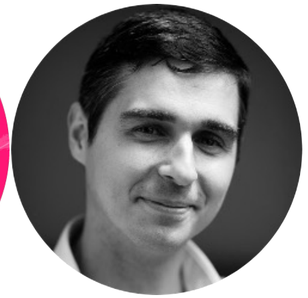
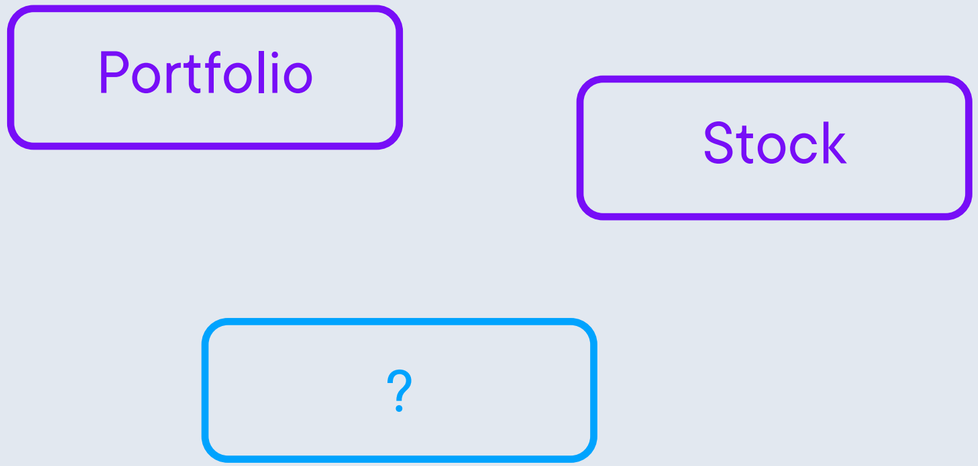
Using TDD to Drive the Design



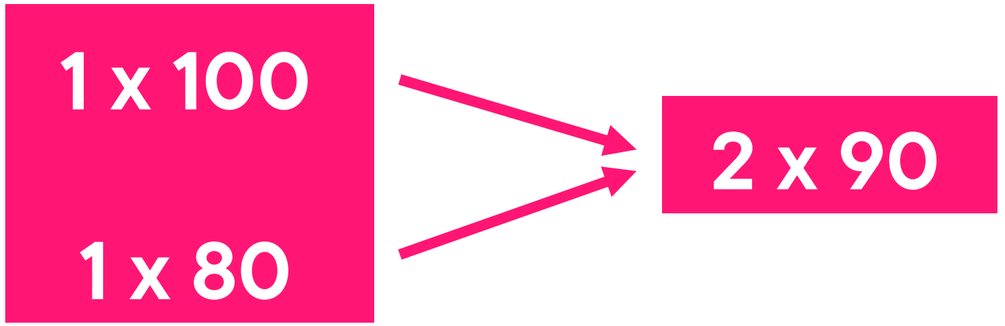
Andrejs Doronins
Software Developer in Test

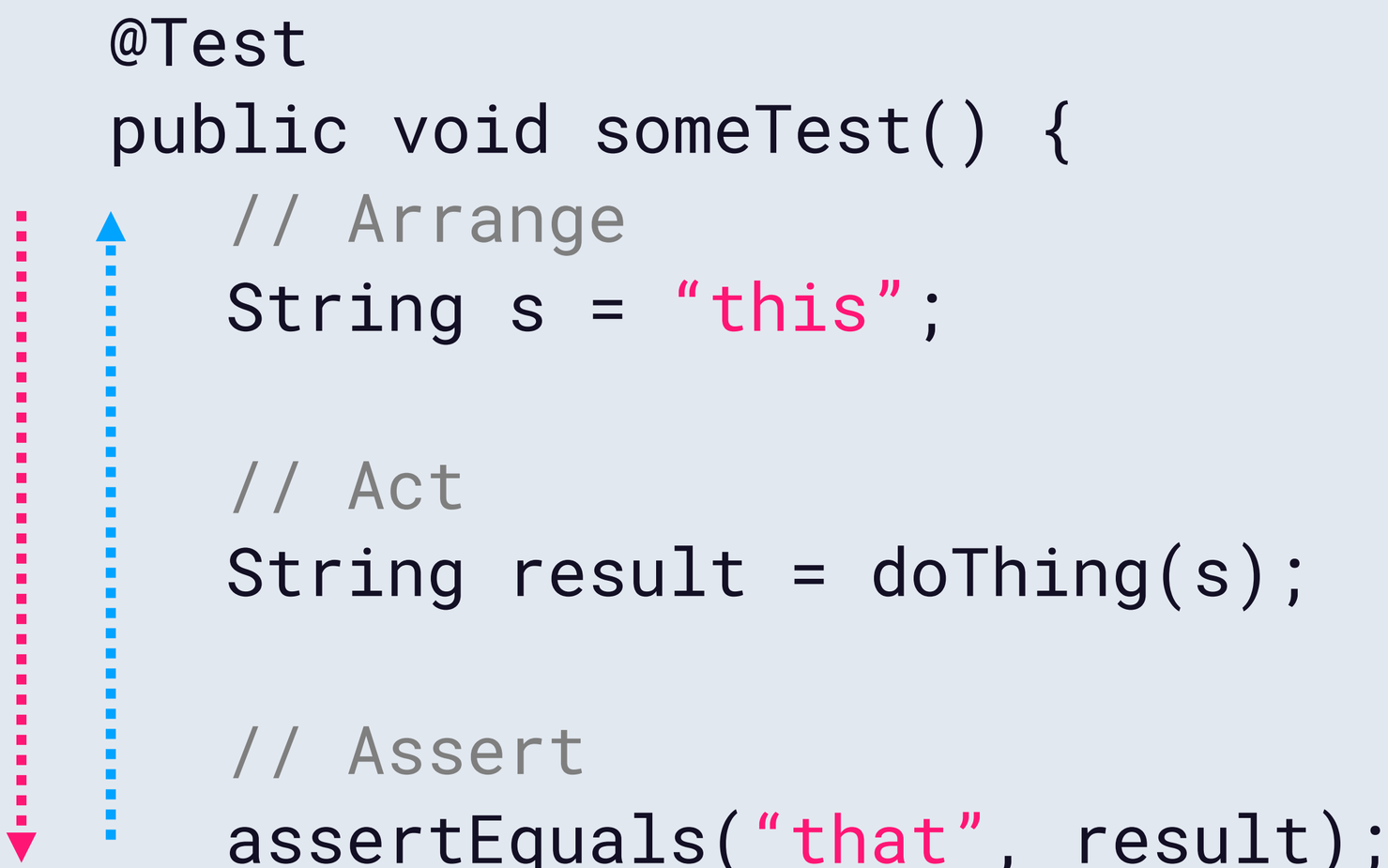
```
String truncate(a, b) {
    // ...
}
```



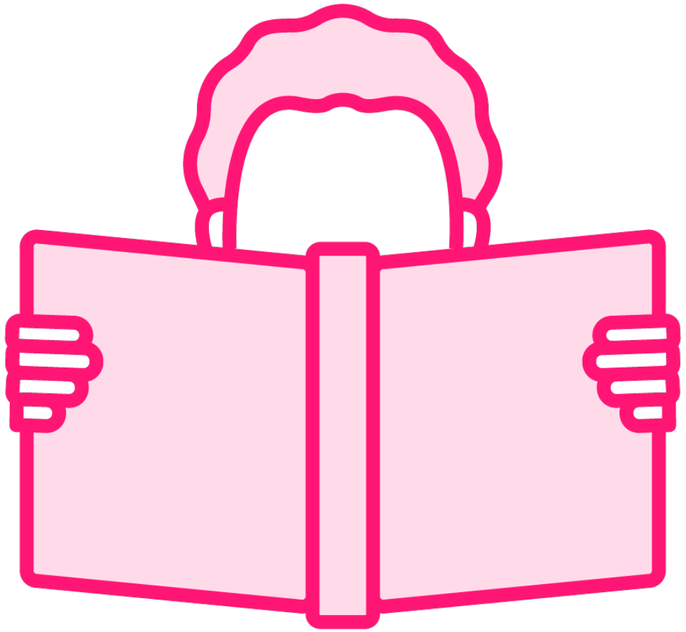
Stock Portfolio

Symbol	Shares (Qty)	Price (Px)	Value
Microsoft	10	250	2500
Apple	2	100	200
			2700




@Test
public void someTest() {
// Arrange
String s = “this”;
// Act
String result = doThing(s);
// Assert
assertEquals(“that”, result);
}

TDD Tutorials

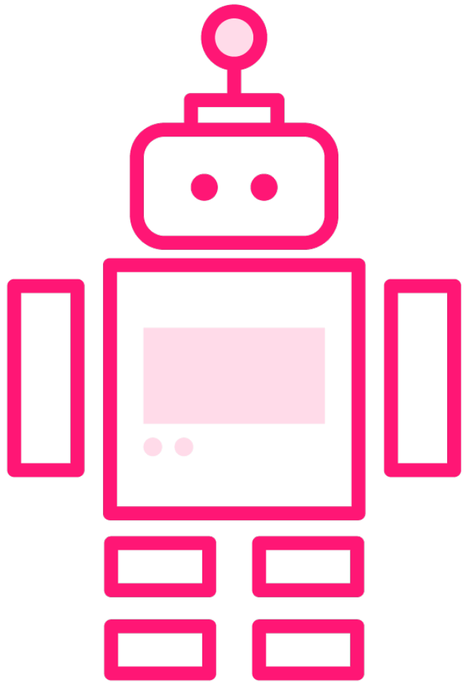


Most have smooth progression

Exception: “Test Driven Development: By Example” book

- Evolving design, things changed a lot

This TDD course



1. Think of the Portfolio example
2. Implement it using TDD
3. Record every detail (incl. mistakes)
4. Present

Stock Portfolio

Symbol	Shares (Qty)	Price (Px)	Value
Microsoft	10	250	2500
Apple	2	100	200
			2700

Public

Private

I'm here to learn TDD, not investing.

Poor domain understanding
=
poor design

Automated Test Best Practices

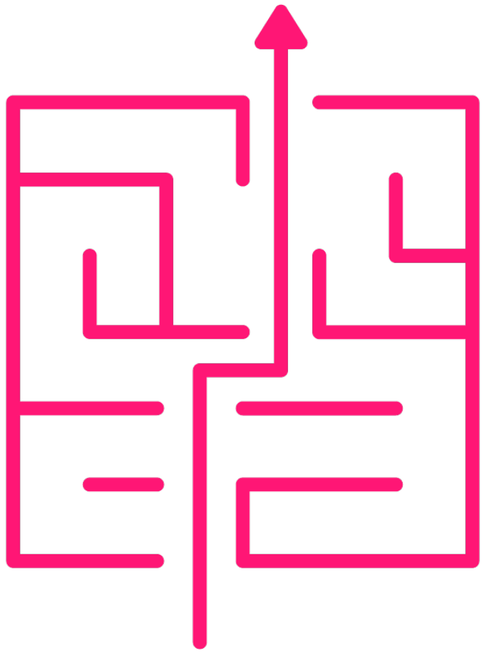
One assertion per test

No need to test trivial getters

But!

- getValue()
- getters will be returning different values

Bigger Challenge



Introduce Money API

Summary

Bottom-up approach when writing tests

Tests drove the design

Small steps weren’t small

Write imperfect code, put things on a TODO list, address later

```
String truncate(a, b) {

    // ...
}
```

Portfolio

Position

Stock

Up Next:

TDD Nuances and Best Practices