Design in the Red Step

26-01-25

Learning Outcomes

1

Goal 1

Remember to design new classes and functions in the test, before they exist

2

Goal 2

Use the capabilities of the IDE to create classes and functions from the test

3

Goal 3

Focus on what you're solving, not all the minute details.

Agenda

Connect: How to create a class

Concept: Design in TDD - in the red step

Do: Lift Button

Reflect: Keyboard shortcuts

Connect : Ways to create a class

Ways to create a class

"I'd like to create a new class, called LiftButton. What should I do? Please navigate me"

```
7   class LiftButtonTest {  new *
8
9       @Test  new *
10      void should_create_a_new_LifButton() {
11
12          //navigate me to create a new LiftButton class
13
14
15          // then
16          assertThat( actual: false)
17                  .isTrue();
18      }
19  }
```

"Use the IDE to create the new class from the test"

Concept: Design in TDD

"How does TDD help improving the design of your code?"

🚀 How does TDD improve the design of your code?

It helps organising what each method must do

Fran

Better setup for future refactoring when needed

It forces you to think about the tests before writing the production code (it makes the code more testable)

Roberto Gonzalez

it helps designing for testability

promotes modularity

It provides some space for refactoring (with a safety net)

it helps to think about how our classes be used from the consumers of our code

Ivan

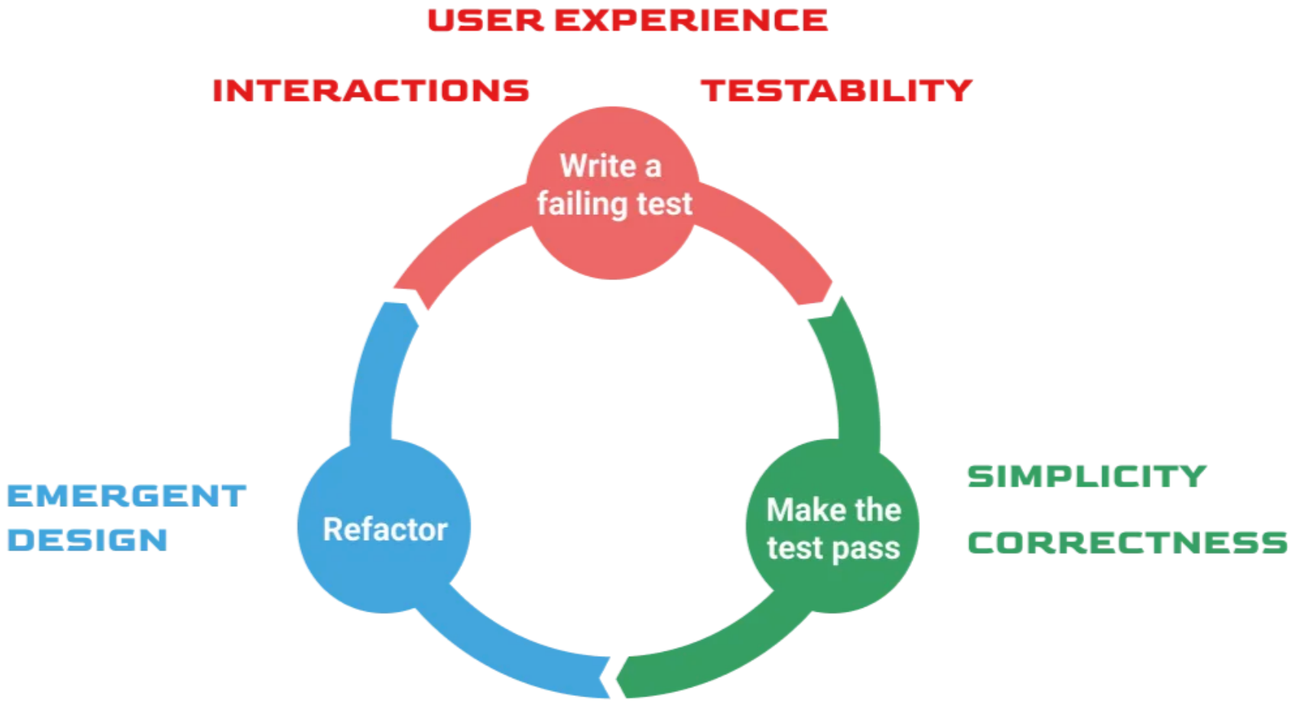
Should help keep methods cleaner

Catherine Grogan

Probably end up with cleaner method interfaces

Add your notes to the board!

2-5 mins



"Effective software design is integral to every phase of the Red-Green-Refactor cycle"

"Effective software design is integral to every phase of the Red-Green-Refactor cycle"

Concept: Design in the Red Step

Design a good user experience

"A failing test is used to design a good user experience for our public APIs before implementing the code"

```java
class LiftButtonTest {  new *

    @Test  new *
    @DisplayName("lights should be ON when button is pressed")
    void lights_should_be_ON_when_button_is_pressed() {

        //given
        LiftButton liftButton = new LiftButton();

        //when
        liftButton.buttonPressed();

        //then
        assertThat( actual: false).as( description: "Not implemented")
                .isTrue();
    }
}
```

Design interactions between system collaborators

" We can use a failing test to design meaningful interactions between our system collaborators"

```java
//system collaborator
@Mock(answer = Answers.RETURNS_DEEP_STUBS)  2 usages
SessionFactory sessionFactory;

//system under test
@InjectMocks  2 usages
BaseDao<FooEntity> baseDao;

@Test  ibanFR *
@DisplayName("should save foo entity")
void shouldSaveFooEntity() {
    //given
    FooEntity foo = new FooEntity();

    //when
    baseDao.save(foo);

    //then
    then(sessionFactory.getCurrentSession())
            .should()
            .save(foo);
}
```

Design for testability

"TDD ensures testability by design. Because of the test-first principle, we are forced to have a design that is testable"

```
class LiftButtonTest {  new *

    @Test  new *
    @DisplayName("lights should be ON when button is pressed")
    void lights_should_be_ON_when_button_is_pressed() {

        //given
        LiftButton liftButton = new LiftButton();

        //when
        liftButton.buttonPressed();

        //then
        assertThat(liftButton.lights())
                .as("lights should be ON")
                .isTrue();
    }
}
```

"A failing test drives code design by defining intuitive APIs, ensuring testability, and enabling interactions between system collaborators"

Concept: Design in the Green step

Design for simplicity

"You are not allowed to write any more production code than is sufficient to pass the one failing unit test"

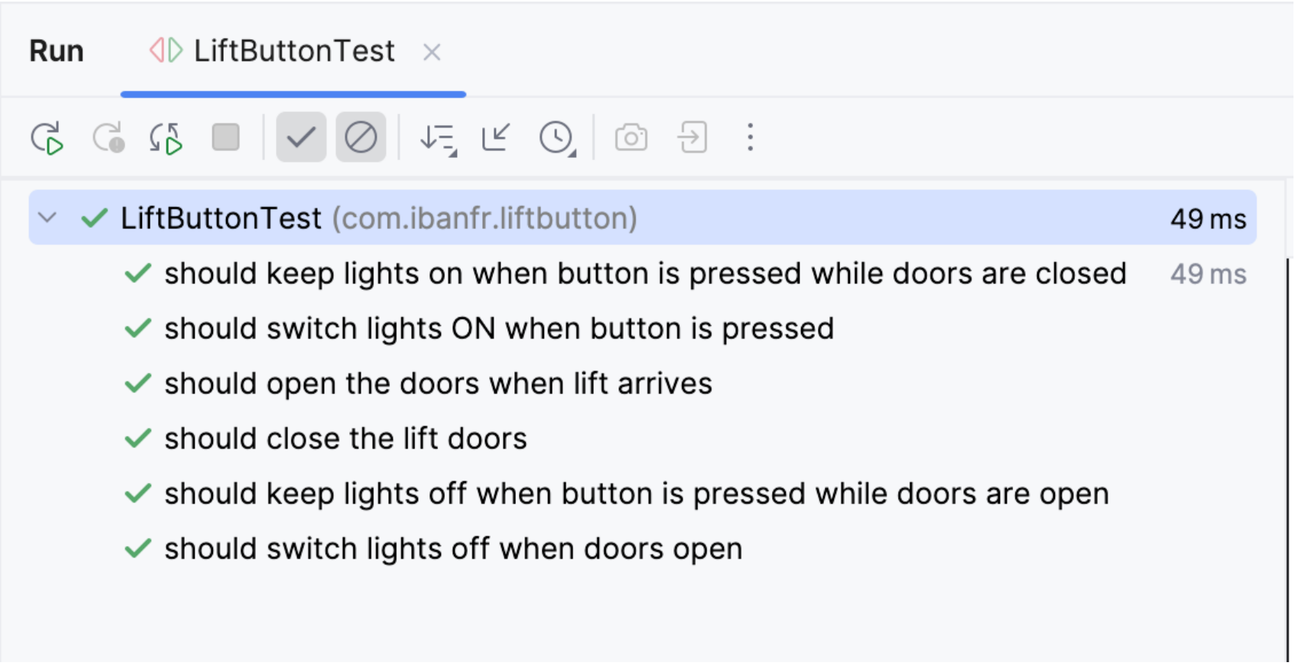
```
package com.ibanfr.leapyears;

public class LeapYearCalculator {  2 usages  ibanFR *

    public static boolean isLeapYear(int year) {  9 usages
        return true;
    }
}
```

Design for correctness

"A passing test means that our code behaves according to the expectations specified in our tests"

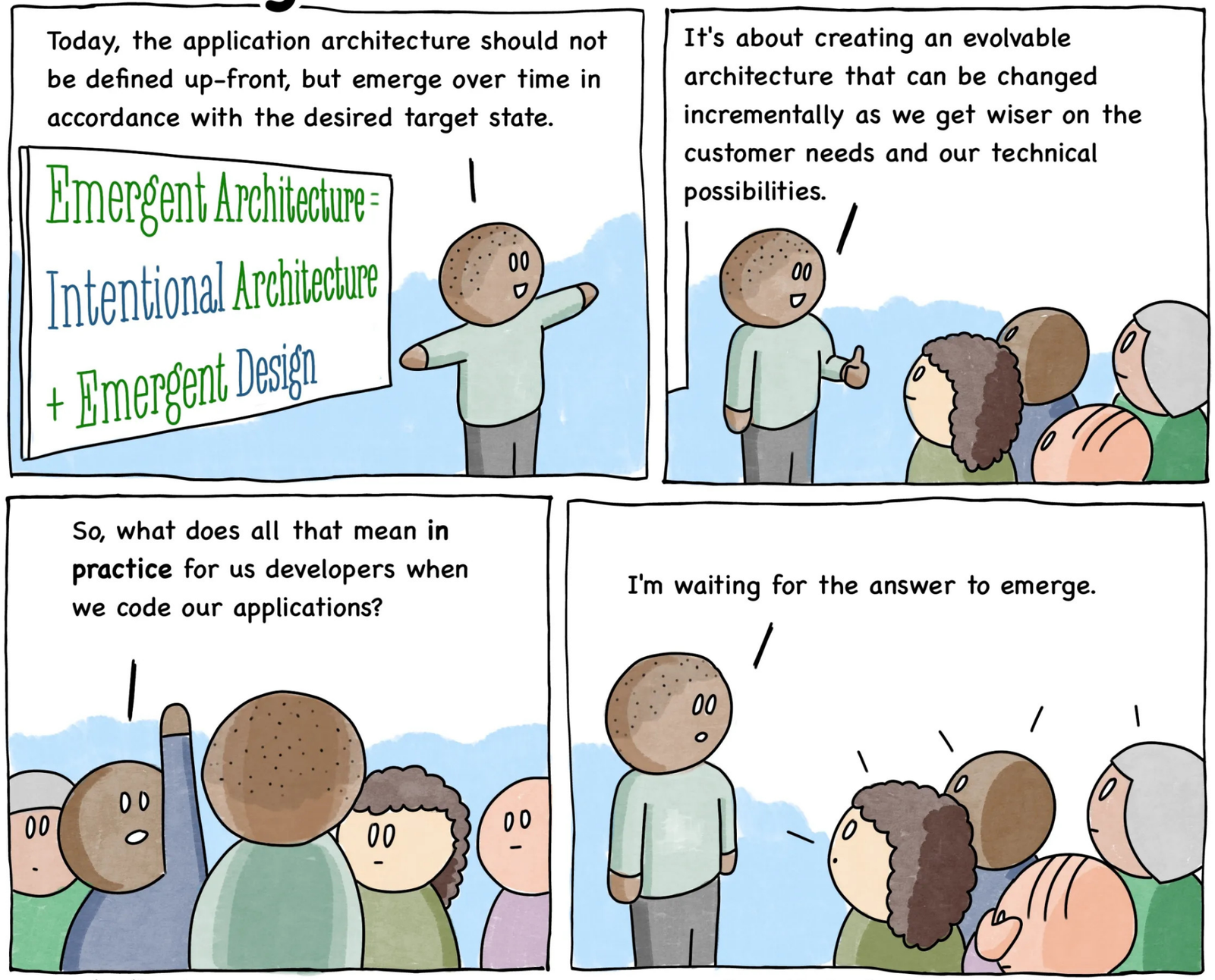


"Design in the Green Step prioritizes simplicity by writing only the minimal code needed to pass a failing test while ensuring correctness by aligning behavior with explicitly defined expectations"

Concept: Design in the Refactor Step

"Emergent Design!"

Comic Agilé



www.comicagile.net

Created by Luxshan Ratnaravi & Mikkel Noe-Nygaard

Emergent Design

"The process of refining our software as a way to improve existing code written in the Green Step"

"Grow the design of our code by continuous refactoring"

```
package ie.etu.leapyears;

public class LeapYearsApp {  1 usage  ibanFR

    public static boolean isLeapYear(int year) {  5 usages  ibanFR

        if (year % 400 == 0) {
            return true;
        }
        if (year % 100 == 0) {
            return false;
        }
        return year % 4 == 0;
    }
}
```

Refactor This

1 Rename... ⇧F6
2 Change Signature... ⌘F6

Extract/Introduce

3 Introduce Parameter Object...
4 Extract Delegate...
5 Extract Interface...
6 Extract Superclass...

7 Inline Method... ⌥⌘N
8 Find Method Duplicates and Replace with Calls...

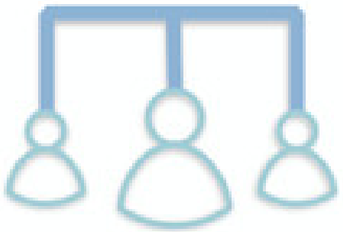
9 Move Members... F6
0 Copy Class... F5
Safe Delete... ⌘⌦

Type Migration... ⇧⌘F6
Convert To Instance Method...

Invert Boolean...

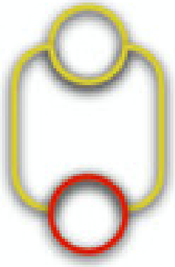
Apply design principles

MODULAR



COHESIVE

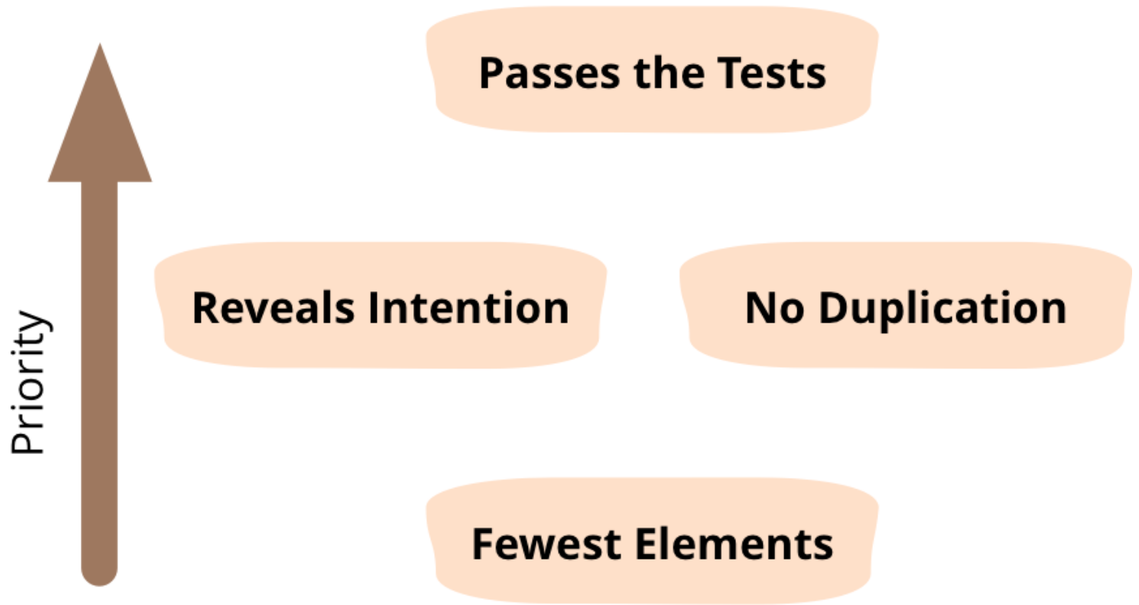
SEPARATION OF CONCERNS



LOOSELY COUPLED

ABSTRACTION/INFORMATION HIDING

Remove duplication & Reveal intention



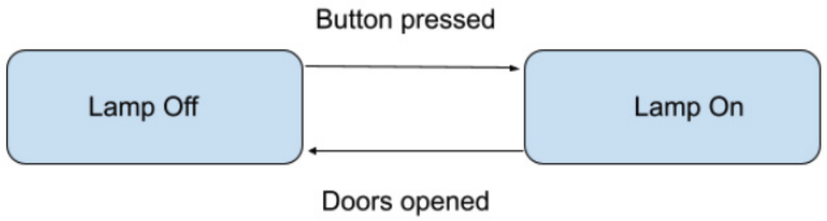
"Emergent Design is an iterative process that focuses on improving software through continuous refactoring"

Do: Lift Button

Lift Button Kata

Write the brains of a lift button. When you press the button, the light comes on. When the lift arrives and the doors open, the light goes out. Pressing the button again while the light is on but the lift doors are closed has no effect. While the doors are open, pressing the button does not switch the light on.

This is a state diagram:



<title>code ninja</title>

🚀 Create a Test List for Lift Button kata

- should turn the light on when button is pressed and the light is off and doors closed
- should do nothing when button is pressed and the light is on but the doors are closed
- should not turn the light on when button is pressed and doors are opened
- should turn the light off when the doors are opened

press button when lift doors CLOSED: light should turn ON

press button when light already ON and doors CLOSED : light should stay ON

press button when lift doors open: light should be OFF

when doors OPEN light should turn OFF

- press the lift button: lights switch on and doors open

- press the lift button while the doors are close and lights are on: don't switch lights on

- press the lift button while the doors are open: don't switch lights on

Add your notes to the board!

2–5 mins

Do the Lift Button Kata

Overview

Practice with the Lift Button Kata in your IDE:

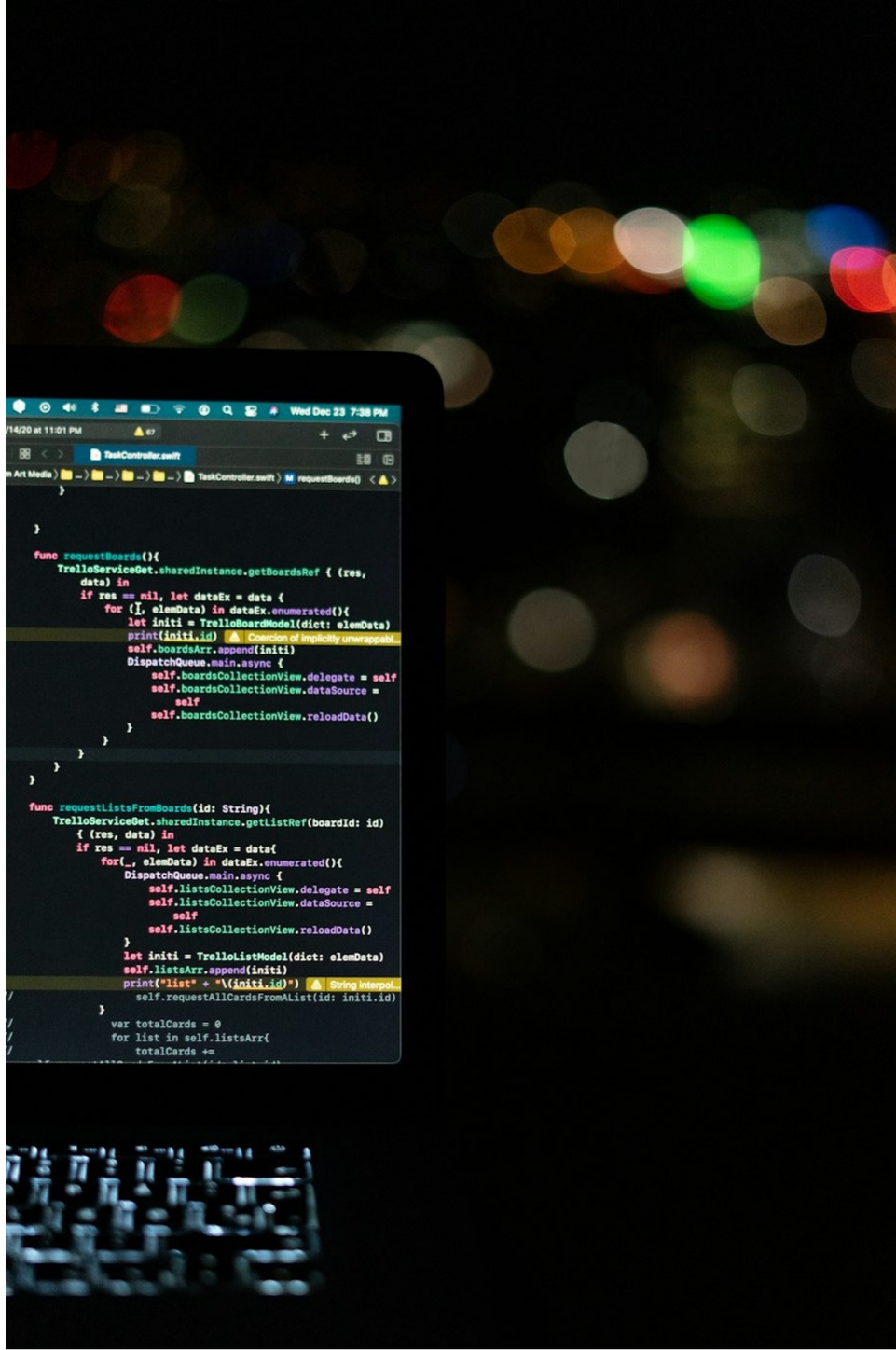
- Practice writing the tests *before* creating the classes and functions they describe.
- Practice using things in the test *before* they exist in the production code.

Work ensemble:

- Add every team member to the Mob timer.
- Switch roles every 5 mins.
- Typist commits the team progress before rotating roles.
- New typist presents the screen.
- Everyone calls out their roles.
- Next person starts the timer.

Github repo

30 mins



Work ensemble on the IDE

Reflect: Keyboard shortcuts

Keyboard shortcuts

"Which keyboard shortcuts did we use to *create* things?"

🚀 Keyboard shortcuts we use to create things

change font size:
ctrl + shift+ ,/.

CMD+C / CMD + X / CMD+V

CMD + N : create new test method (beforeEach)

type "test" on a test class

copy paste description with 'test' setup auto populates both

ctrl + R (Run current test)

CTRL+T (Refactor This)

option + enter gives available options to implement methods, variables, ...

right mouse click > show context actions

ctrl + shift+ R (run all tests from class)

Add your notes to the board!

2-5 mins

References

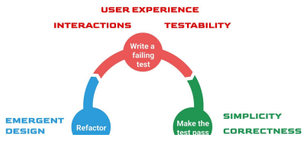
Samman Coaching

sammancoaching.org

Design in the Red step

Design in the Red step

Craft Better Software



craftbettersoftware.com

The TDD Debate: Testing, Design or Development tool?

Test-Driven Development controversy: which role does it serve best?

Thank you.