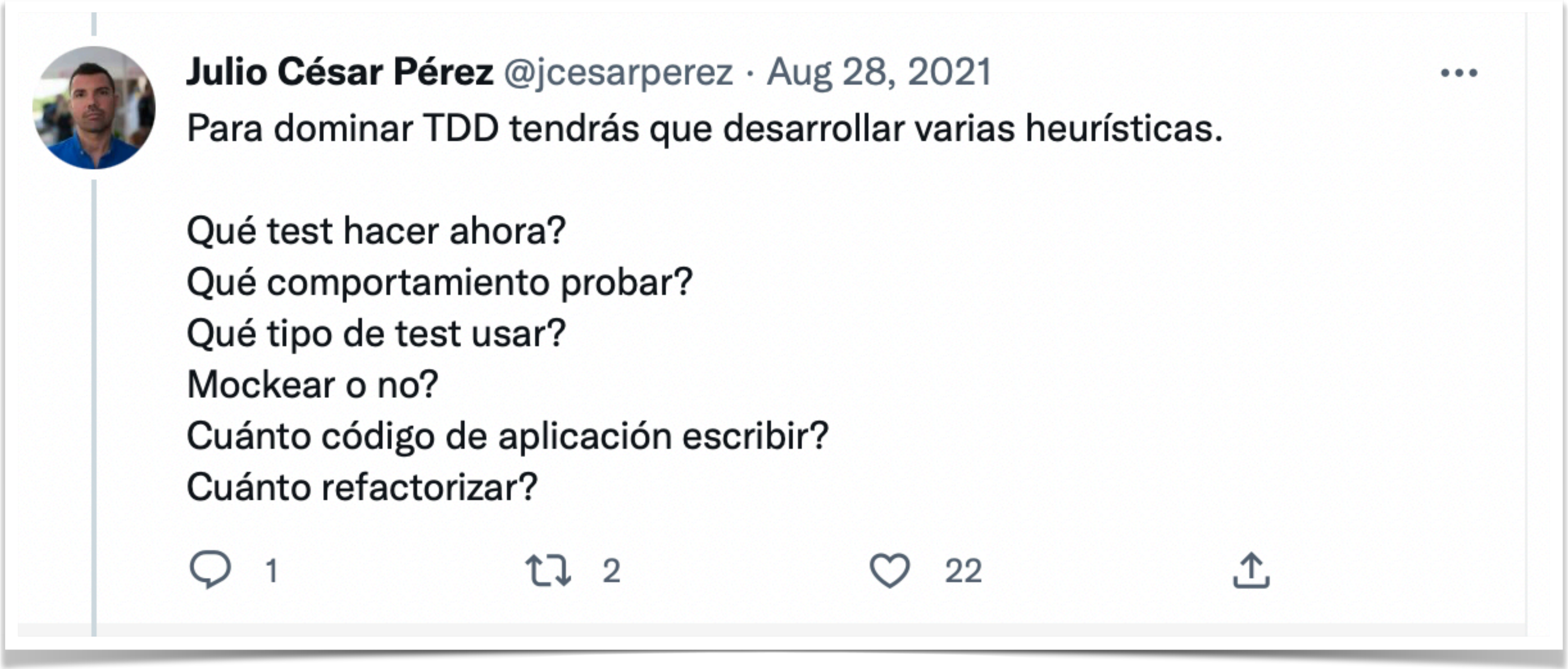
MADS
S7: TDD
(Bloque 2 - XP)

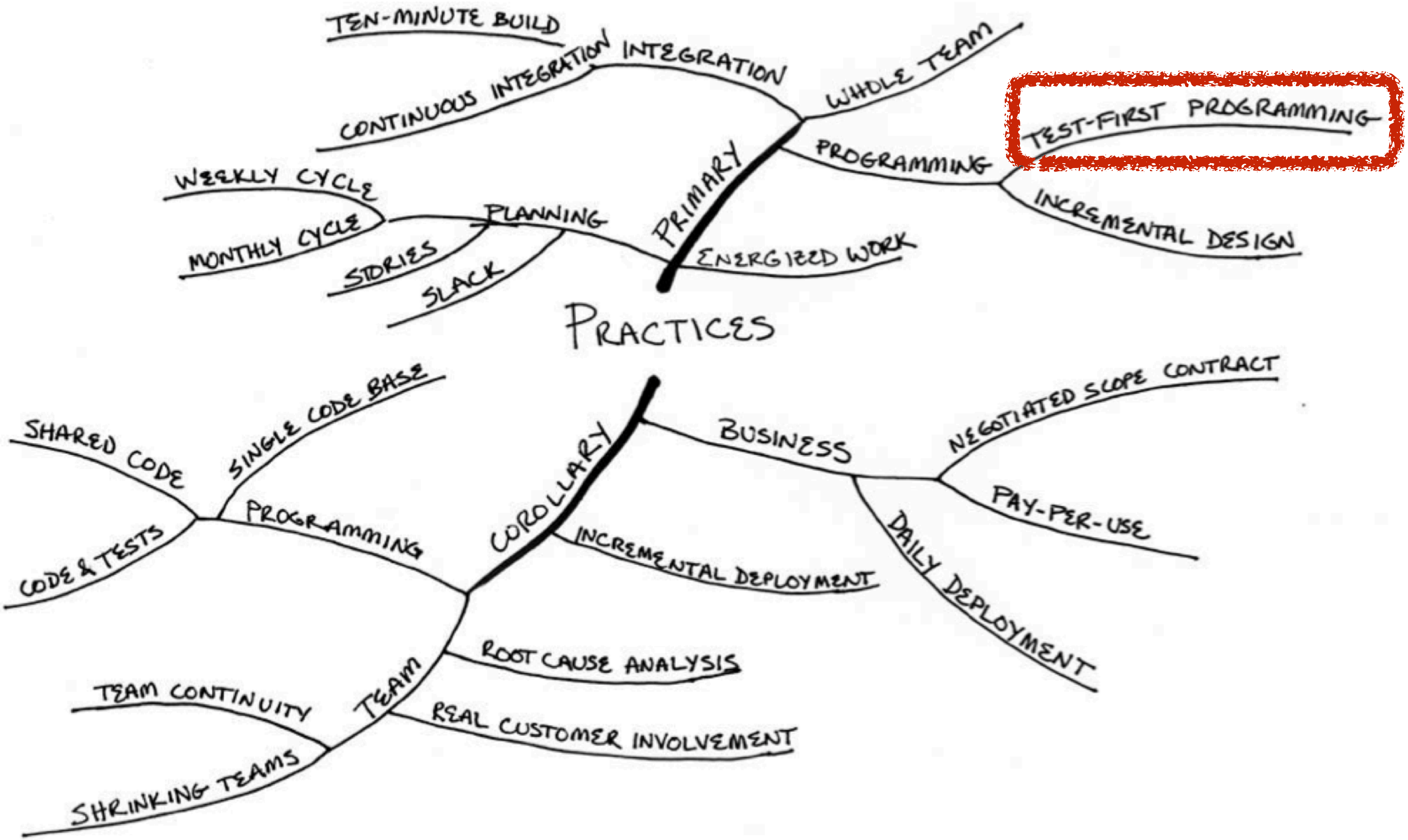
El tweet de la semana

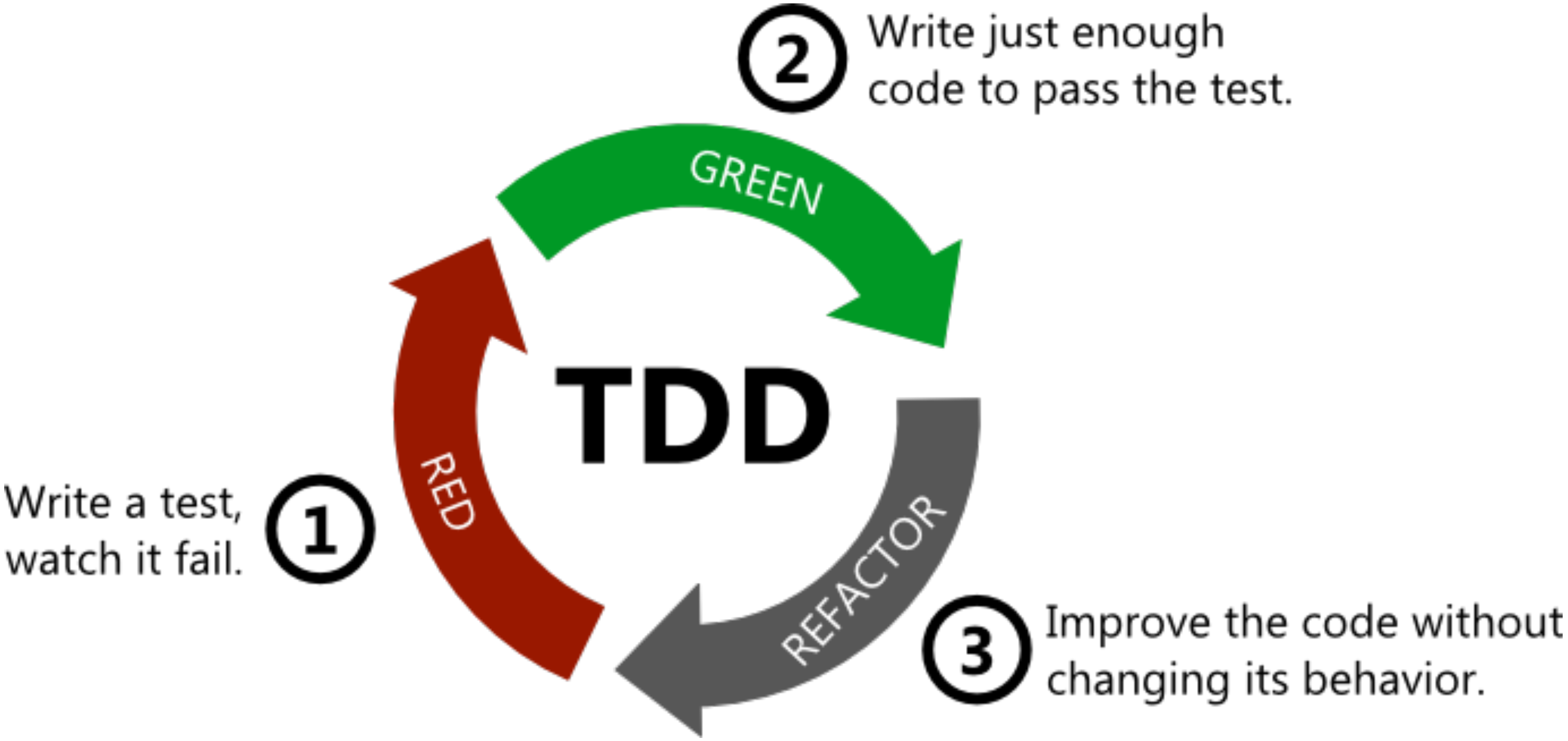


Julio César Pérez @jcesarperez · Aug 28, 2021

Para dominar TDD tendrás que desarrollar varias heurísticas.

Qué test hacer ahora?
Qué comportamiento probar?
Qué tipo de test usar?
Mockear o no?
Cuánto código de aplicación escribir?
Cuánto refactorizar?

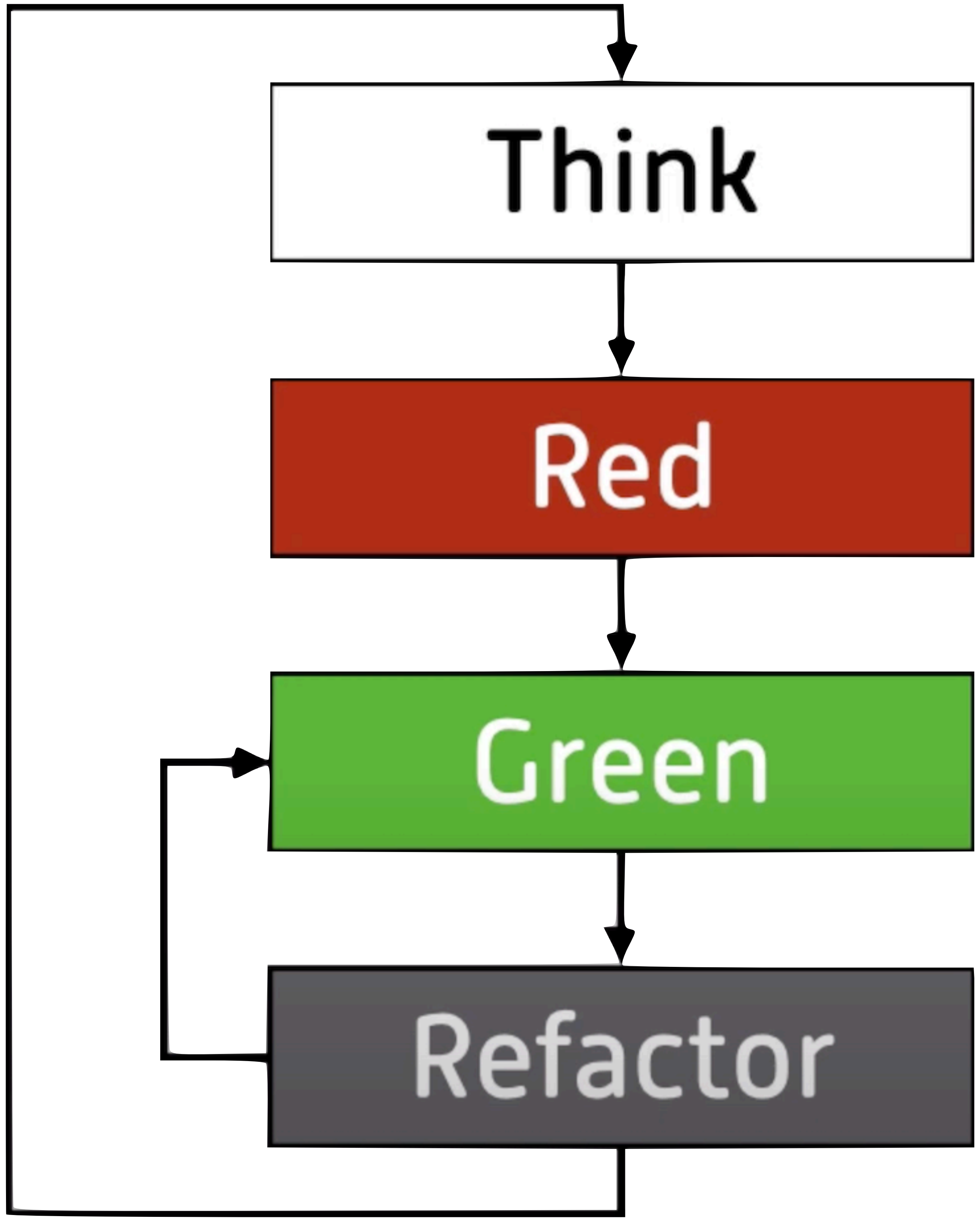

PRACTICES
PRIMARY
INTEGRATION
TEN-MINUTE BUILD
CONTINUOUS INTEGRATION
WHOLE TEAM
PROGRAMMING
TEST-FIRST PROGRAMMING
INCREMENTAL DESIGN
PLANNING
WEEKLY CYCLE
MONTHLY CYCLE
STORIES
SLACK
ENERGIZED WORK
COROLLARY
BUSINESS
NEGOTIATED SCOPE CONTRACT
PAY-PER-USE
DAILY DEPLOYMENT
INCREMENTAL DEPLOYMENT
PROGRAMMING
SINGLE CODE BASE
SHARED CODE
CODE & TESTS
TEAM
ROOT CAUSE ANALYSIS
REAL CUSTOMER INVOLVEMENT
TEAM CONTINUITY
SHRINKING TEAMS



1. Write a test, watch it fail.

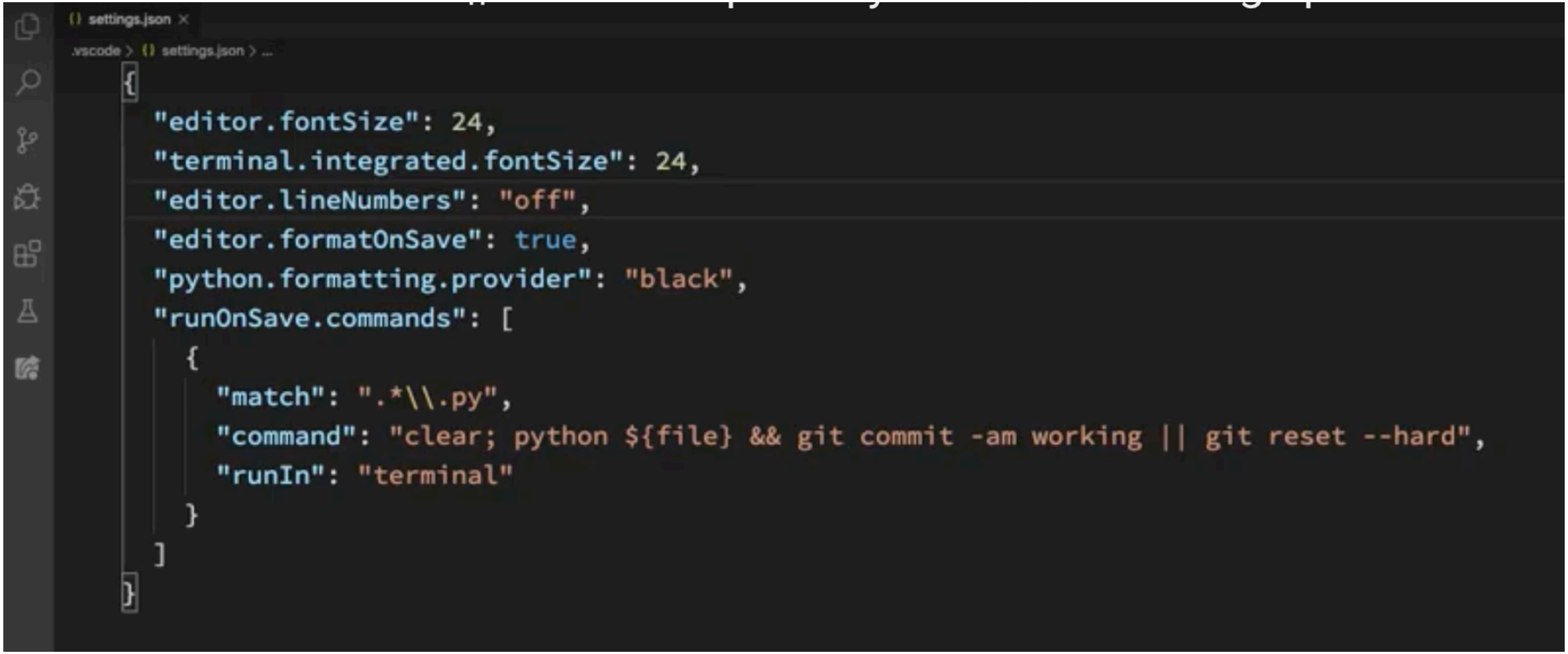
2. Write just enough code to pass the test.

3. Improve the code without changing its behavior.

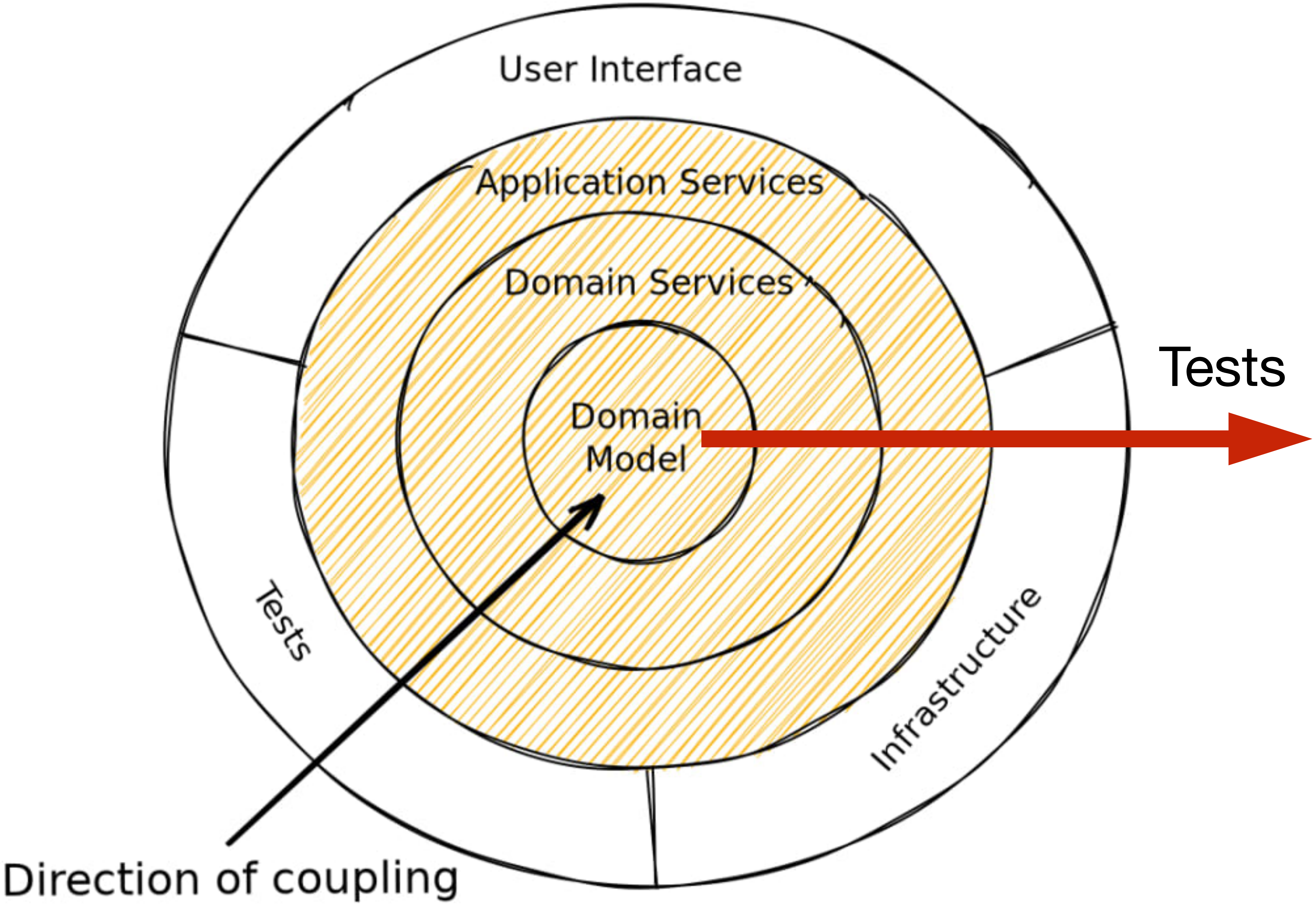

Think
Red
Green
Refactor

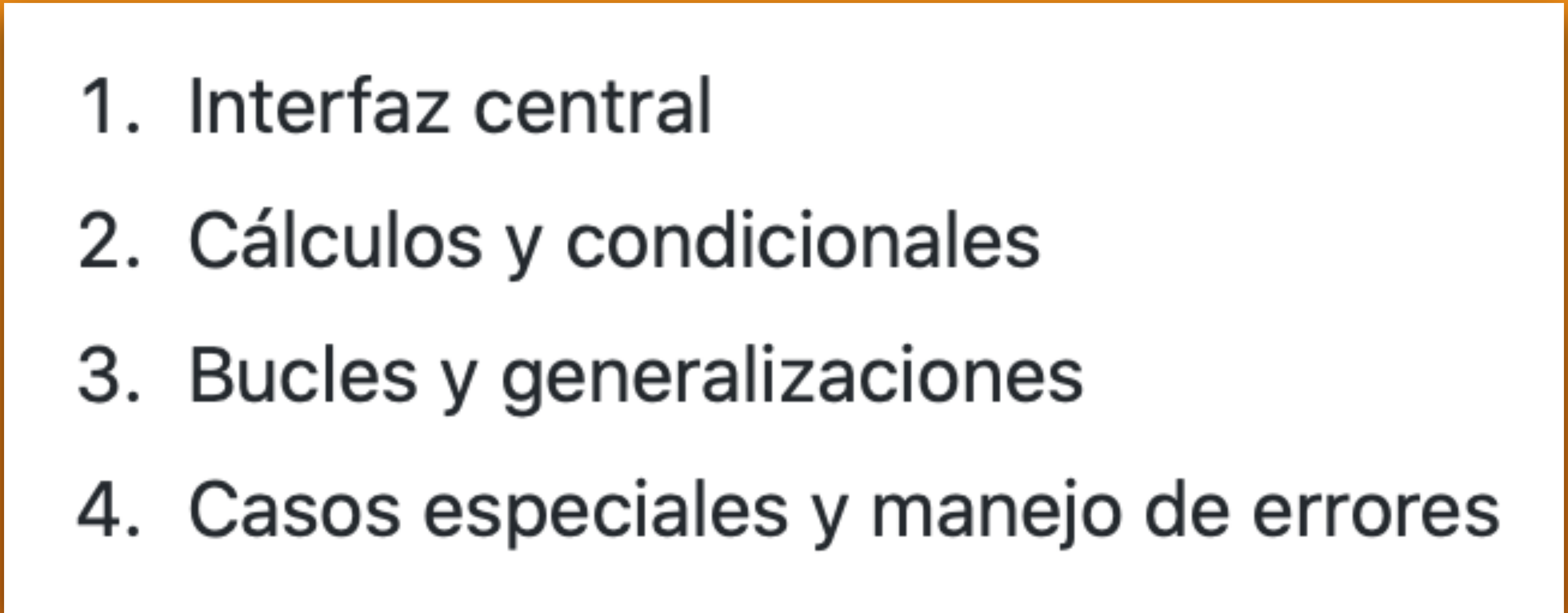
Pequeños incrementos

TCR: Test && Commit || Revert

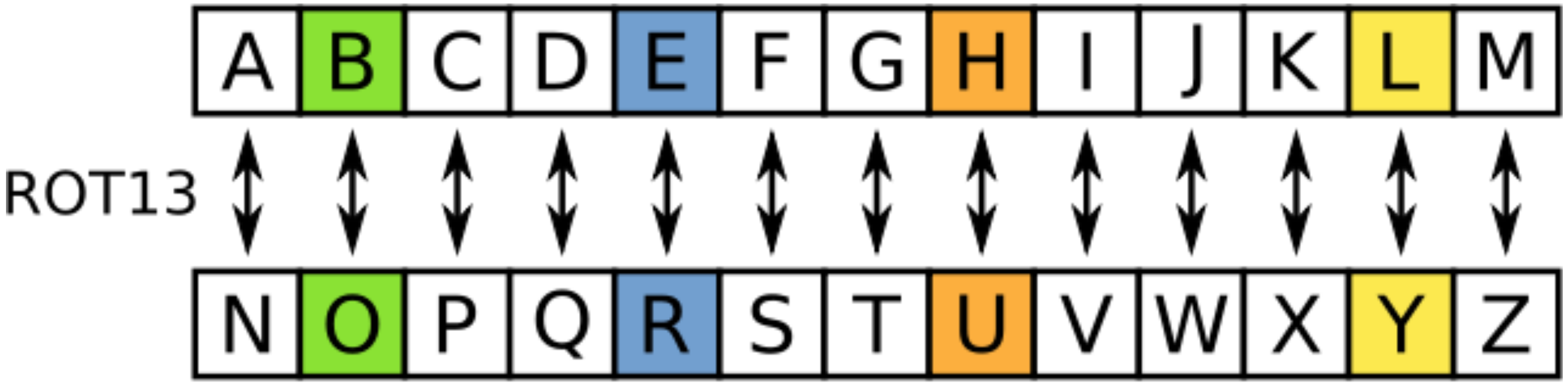


```
{
  "editor.fontSize": 24,
  "terminal.integrated.fontSize": 24,
  "editor.lineNumbers": "off",
  "editor.formatOnSave": true,
  "python.formatting.provider": "black",
  "runOnSave.commands": [
    {
      "match": ".*\\.py",
      "command": "clear; python ${file} && git commit -am working || git reset --hard",
      "runIn": "terminal"
    }
  ]
}
```


User Interface
Application Services
Domain Services
Domain
Model
Tests
Tests
Infrastructure
Direction of coupling



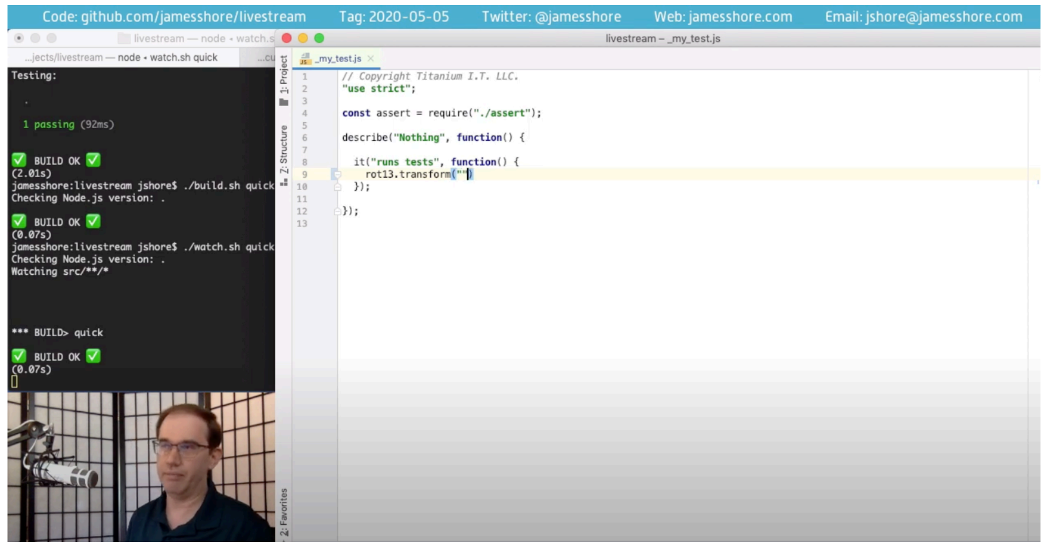
1. Interfaz central
2. Cálculos y condicionales
3. Bucles y generalizaciones
4. Casos especiales y manejo de errores



A B C D E F G H I J K L M

ROT13

N O P Q R S T U V W X Y Z



Code: github.com/jamesshore/livestream   Tag: 2020-05-05   Twitter: @jamesshore   Web: jamesshore.com   Email: jshore@jamesshore.com
livestream — node • watch.s
...jects/livestream — node • watch.sh quick
livestream – _my_test.js
_my_test.js
1: Project
7: Structure
2: Favorites

```
Testing:

  .

  1 passing (92ms)

✅ BUILD OK ✅
(2.01s)
jamesshore:livestream jshore$ ./build.sh quick
Checking Node.js version: .

✅ BUILD OK ✅
(0.07s)
jamesshore:livestream jshore$ ./watch.sh quick
Checking Node.js version: .
Watching src/**/*



*** BUILD> quick

✅ BUILD OK ✅
(0.07s)
```

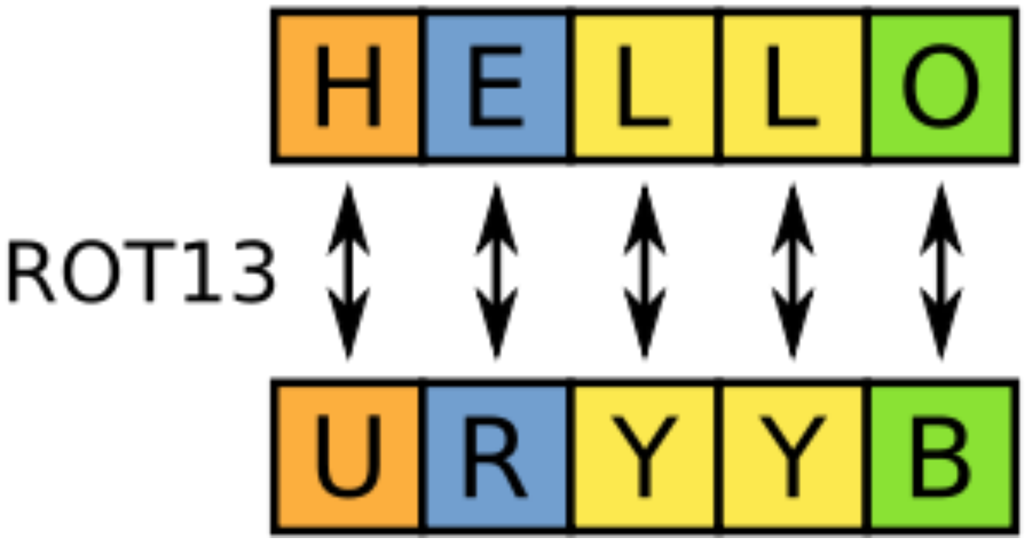
```
// Copyright Titanium I.T. LLC.
"use strict";

const assert = require("./assert");

describe("Nothing", function() {

  it("runs tests", function() {
    rot13.transform("")
  });

});
```



H E L L O

ROT13

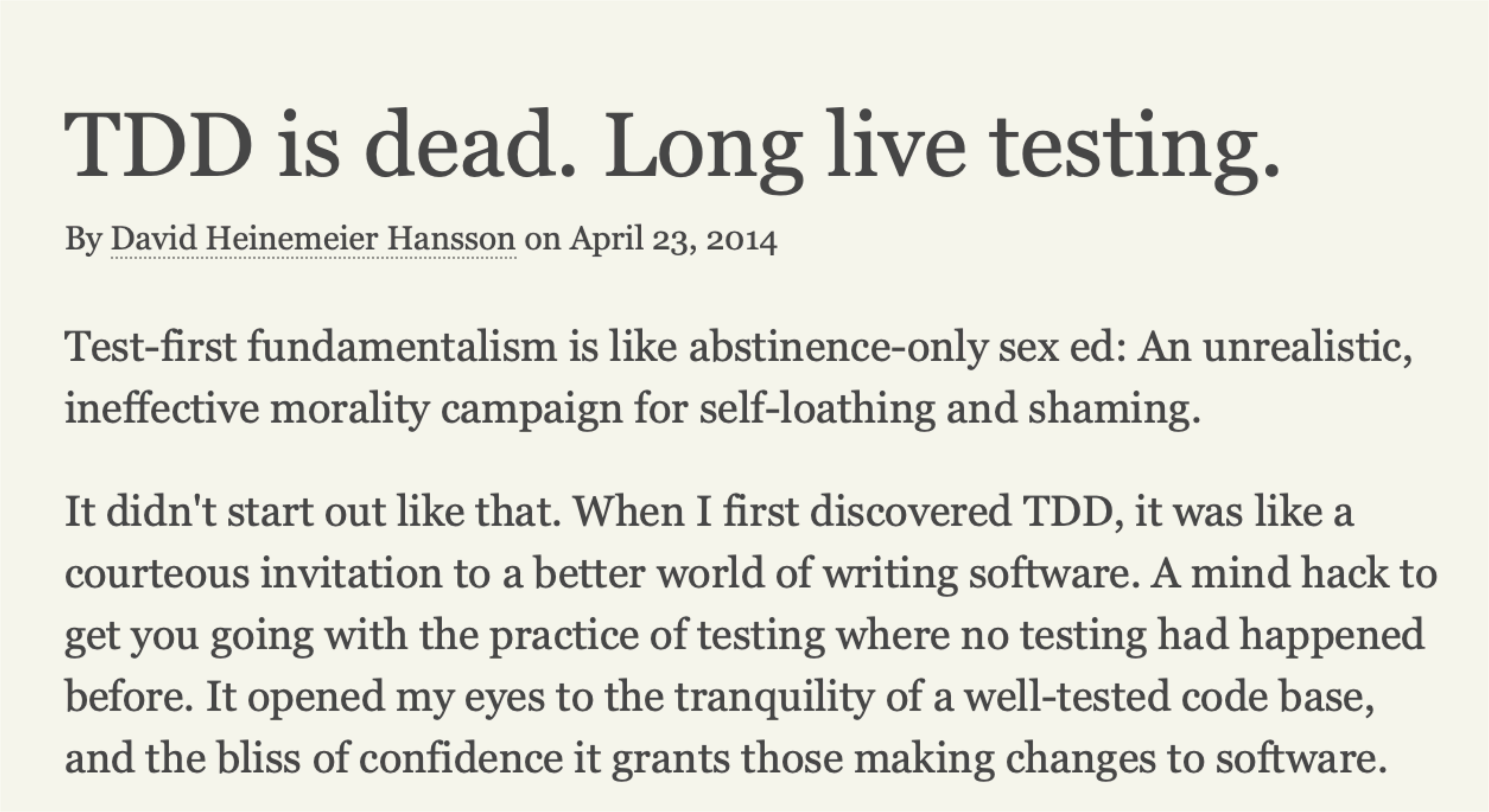
U R Y Y B

Ventajas del uso de TDD

- Especificación clara en todo momento. No perdemos el foco ni el objetivo.
- Mejor diseño.
- Confianza, robustez y documentación.
- Ritmo, flujo.

Críticas

- Lentitud -> ¿comparado con qué?
- Exceso de énfasis en tests unitarios y mocks (Hanson)

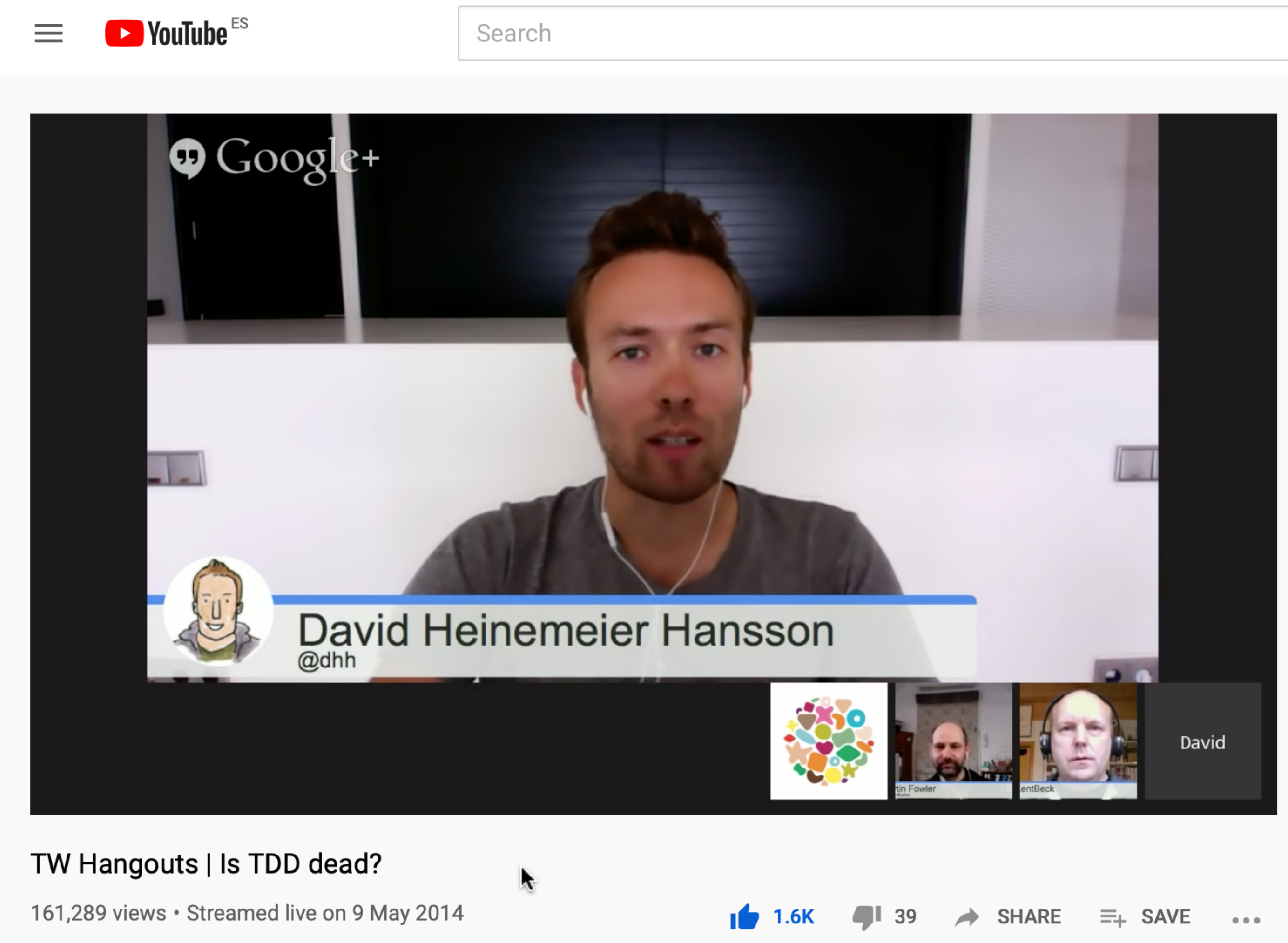


TDD is dead. Long live testing.

By David Heinemeier Hansson on April 23, 2014

Test-first fundamentalism is like abstinence-only sex ed: An unrealistic, ineffective morality campaign for self-loathing and shaming.

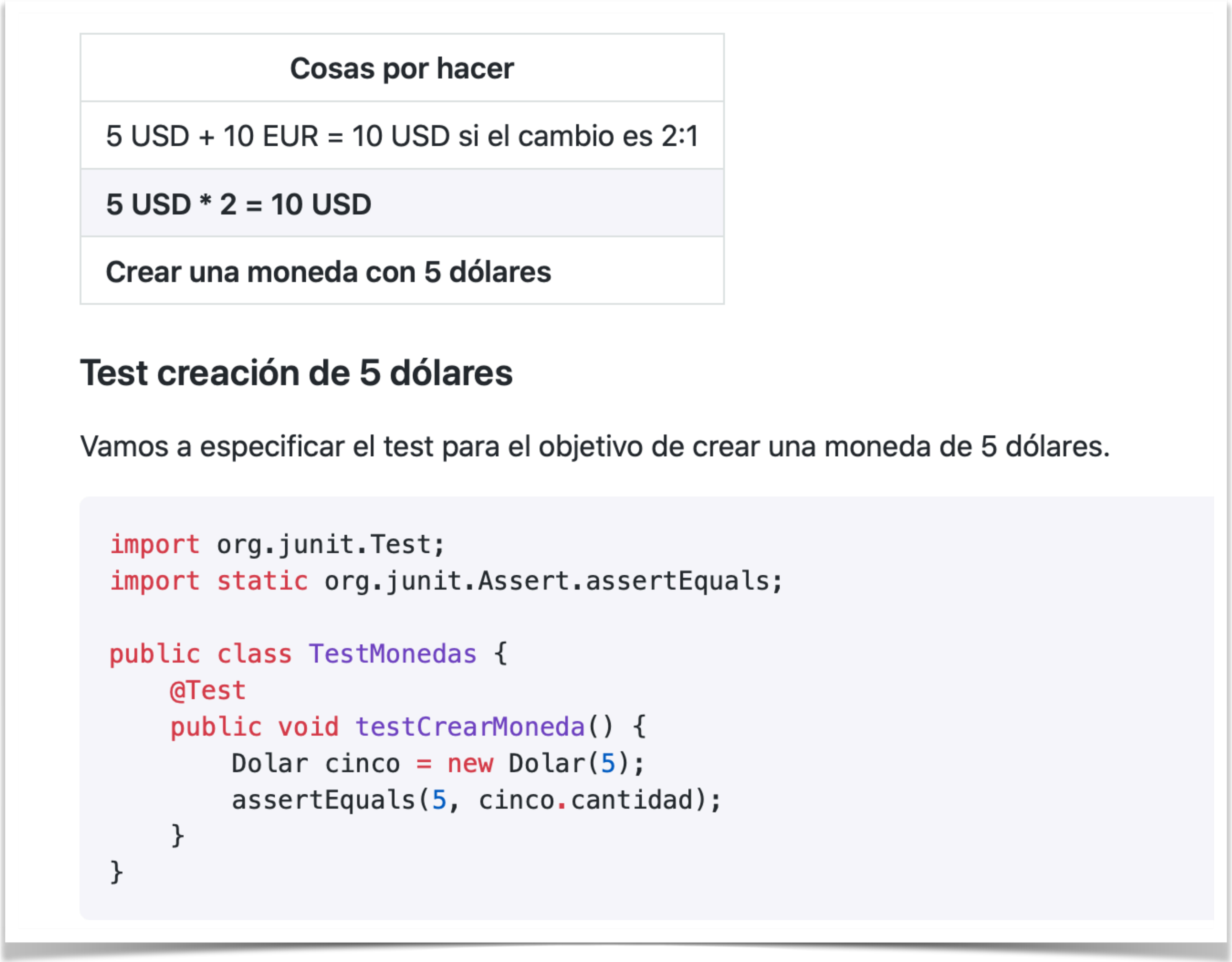
It didn't start out like that. When I first discovered TDD, it was like a courteous invitation to a better world of writing software. A mind hack to get you going with the practice of testing where no testing had happened before. It opened my eyes to the tranquility of a well-tested code base, and the bliss of confidence it grants those making changes to software.



TW Hangouts | Is TDD dead?

161,289 views • Streamed live on 9 May 2014

Ejemplo: dólares y euros



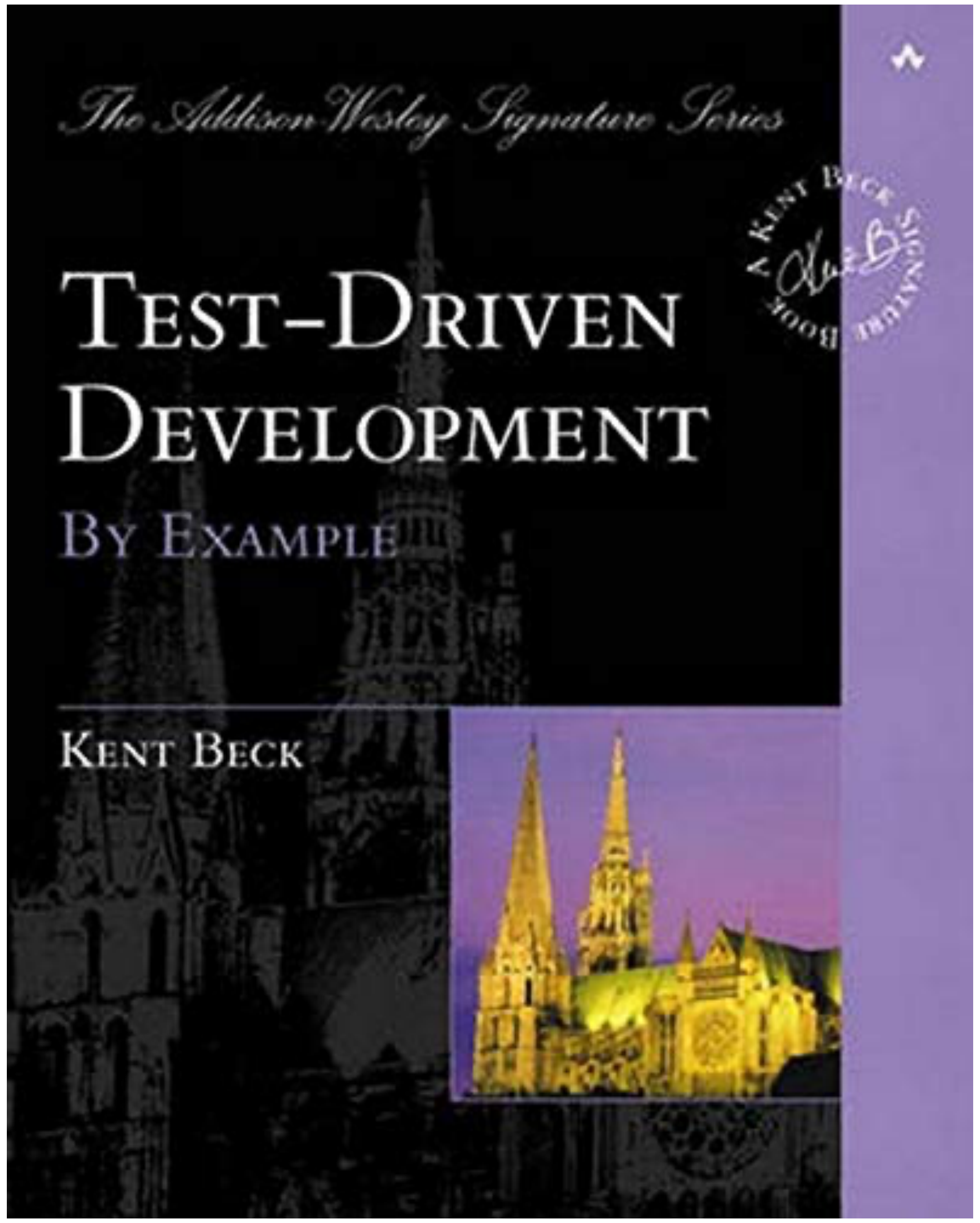
Cosas por hacer
5 USD + 10 EUR = 10 USD si el cambio es 2:1
5 USD * 2 = 10 USD
Crear una moneda con 5 dólares

Test creación de 5 dólares

Vamos a especificar el test para el objetivo de crear una moneda de 5 dólares.

```java
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class TestMonedas {
    @Test
    public void testCrearMoneda() {
        Dolar cinco = new Dolar(5);
        assertEquals(5, cinco.cantidad);
    }
}
```


The Addison Wesley Signature Series
A Kent Beck Signature Book
Test-Driven Development
By Example
Kent Beck

https://github.com/domingogallardo/tdd-monedas