Test Driven Design

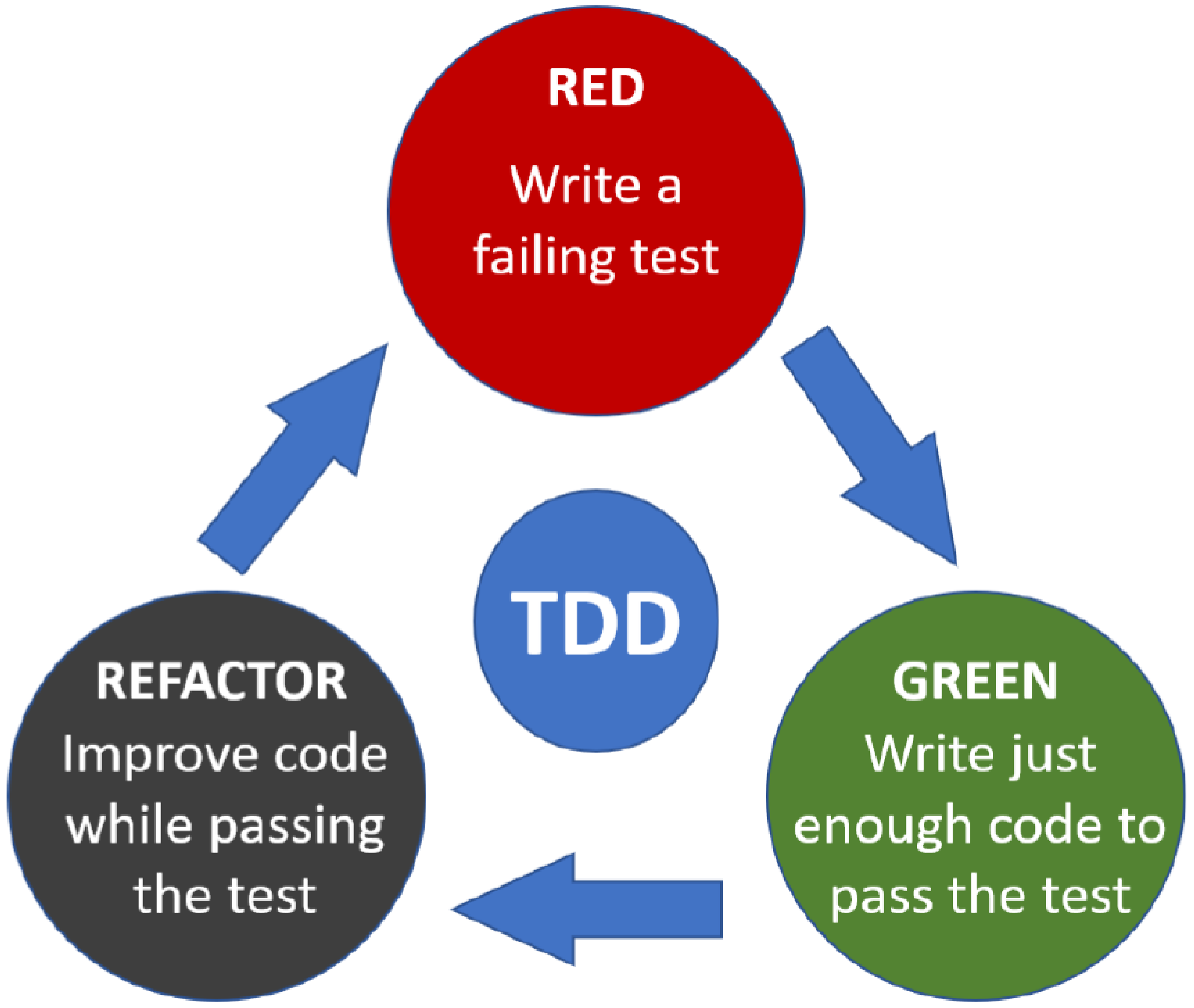
Interactive mob programming

Rune Holen / Bouvet

Idea: One team member is **driver**, while another is the **navigator**.
The rest - the **mob**

Start with a red test case

Refactoring -> use the *Rule of Three*: Extract duplication only when you see it for the third time


RED
Write a failing test
GREEN
Write just enough code to pass the test
REFACTOR
Improve code while passing the test
TDD

Start with a red test case

Implement code accordingly,
starting with e.g. **fake implementation**
→ return hardcoded value

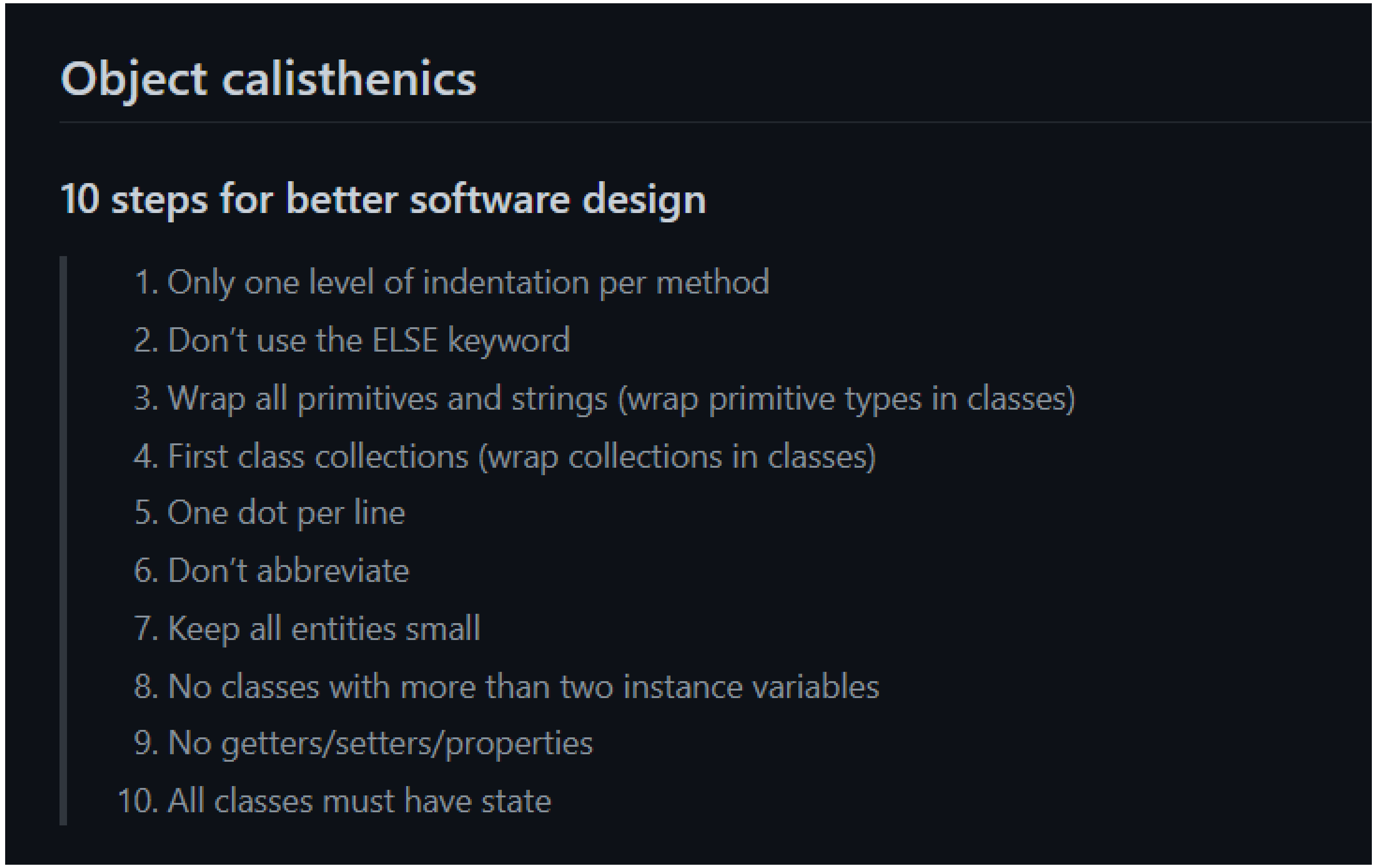
Write a new test case (red)

Continue with **Obvious implementation** (when simple),
otherwise continue with **triangulation**

Transformation **P**riority **P**remise

1. **Fake implementation**
2. **Obvious implementation**
3. **Triangulation**

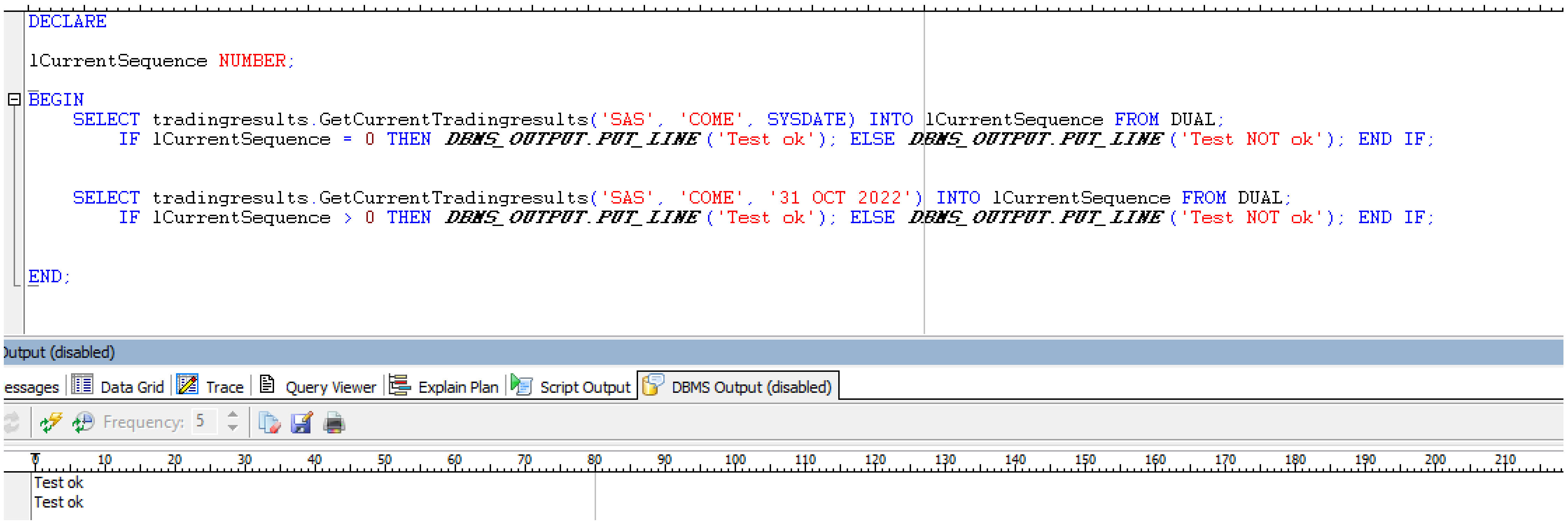
Object Calesthenics



Object calisthenics

10 steps for better software design

> 1. Only one level of indentation per method
> 2. Don't use the ELSE keyword
> 3. Wrap all primitives and strings (wrap primitive types in classes)
> 4. First class collections (wrap collections in classes)
> 5. One dot per line
> 6. Don't abbreviate
> 7. Keep all entities small
> 8. No classes with more than two instance variables
> 9. No getters/setters/properties
> 10. All classes must have state



```
DECLARE

lCurrentSequence NUMBER;

BEGIN
    SELECT tradingresults.GetCurrentTradingresults('SAS', 'COME', SYSDATE) INTO lCurrentSequence FROM DUAL;
        IF lCurrentSequence = 0 THEN DBMS_OUTPUT.PUT_LINE('Test ok'); ELSE DBMS_OUTPUT.PUT_LINE('Test NOT ok'); END IF;


    SELECT tradingresults.GetCurrentTradingresults('SAS', 'COME', '31 OCT 2022') INTO lCurrentSequence FROM DUAL;
        IF lCurrentSequence > 0 THEN DBMS_OUTPUT.PUT_LINE('Test ok'); ELSE DBMS_OUTPUT.PUT_LINE('Test NOT ok'); END IF;


END;
```

```
Test ok
Test ok
```

Simple scenario: Write (automatic) unit test for existing PL/SQL code