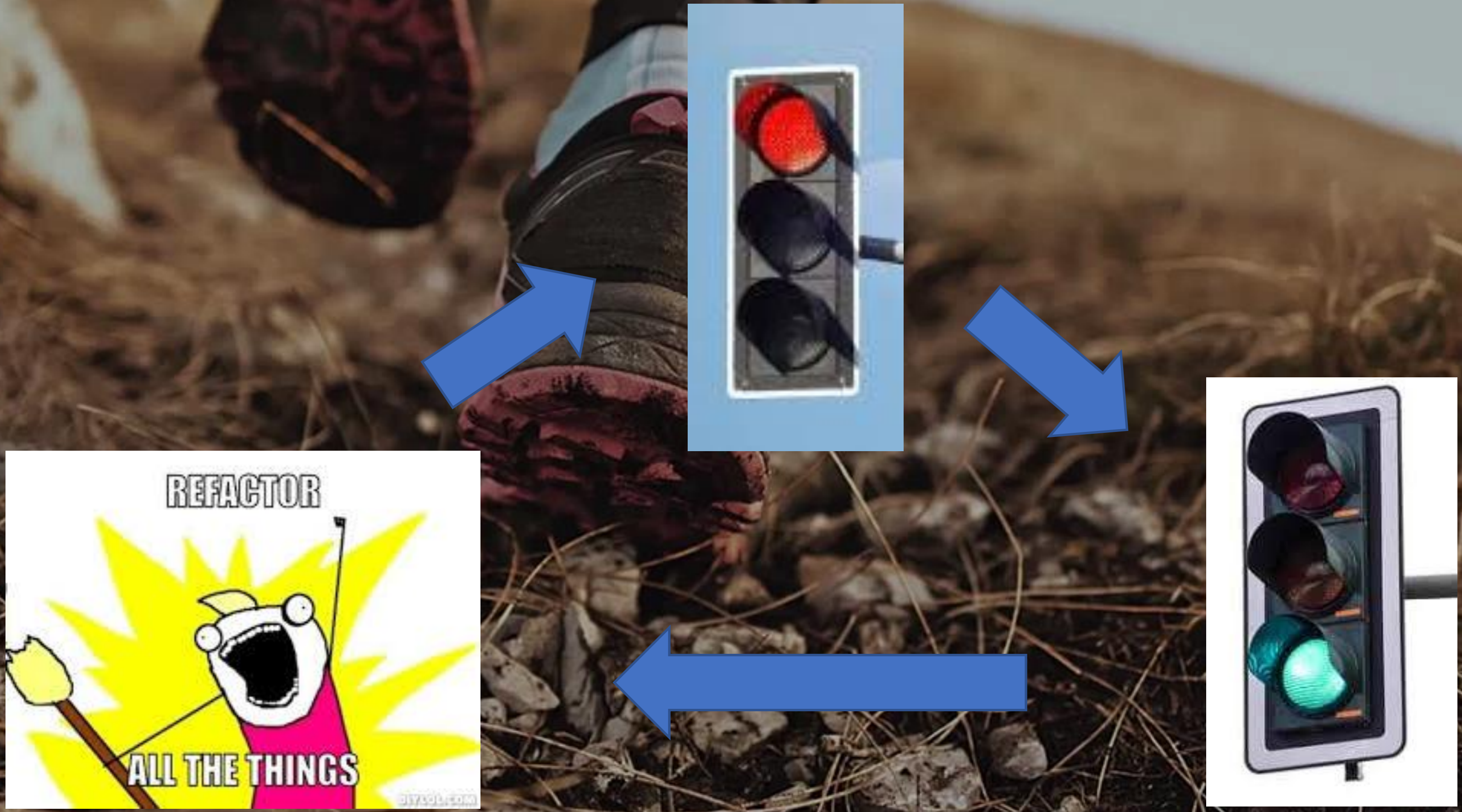
TDD Walking

Sacha Peter

The three states of TDD



Start writing a RED test

- One single behaviour
- One logical assertion
- Mutually independent

- Meaningful names
- Failing for the right reason
- AAA

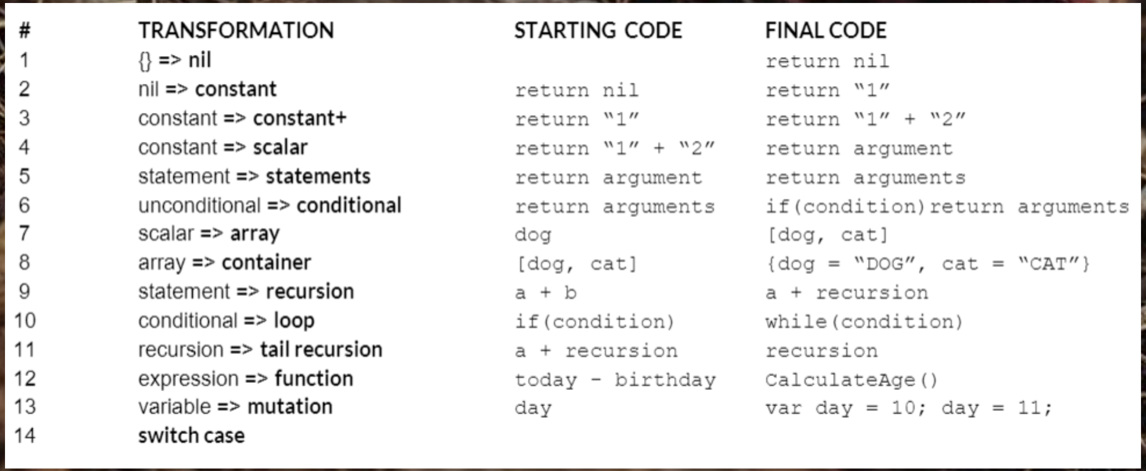
- The tests are the documentation!

Then make it GREEN

- Simplest code to pass the test

Then make it GREEN

- Simplest code to pass the test
- Fake implementation
- Obvious implementation
- Triangulation
- Transformation Priority Premise:



#	TRANSFORMATION	STARTING CODE	FINAL CODE
1	{} => nil		`return nil`
2	nil => constant	`return nil`	`return “1”`
3	constant => constant+	`return “1”`	`return “1” + “2”`
4	constant => scalar	`return “1” + “2”`	`return argument`
5	statement => statements	`return argument`	`return arguments`
6	unconditional => conditional	`return arguments`	`if(condition)return arguments`
7	scalar => array	`dog`	`[dog, cat]`
8	array => container	`[dog, cat]`	`{dog = “DOG”, cat = “CAT”}`
9	statement => recursion	`a + b`	`a + recursion`
10	conditional => loop	`if(condition)`	`while(condition)`
11	recursion => tail recursion	`a + recursion`	`recursion`
12	expression => function	`today - birthday`	`CalculateAge()`
13	variable => mutation	`day`	`var day = 10; day = 11;`
14	switch case		

Refactor!!!

- All test must be green
- Refactor aggressively and constantly
- Rule of three

- Object Calisthenics rules

1. Only one level of indentation per method
2. Don’t use the ELSE keyword
3. Wrap all primitives and strings
4. First class collections (wrap all collections)
5. Only one dot per line `~~dog.Body.Tail.Wag()~~ => dog.ExpressHappiness()`
6. No abbreviations
7. Keep all entities small
   [10 files per package, 50 lines per class, 5 lines per method, 2 arguments per method]
8. No classes with more than two instance variables
9. No public getters/setters/properties
10. All classes must have state

Mob Programming

- Driver and Navigator

Mob Programming

- Driver and Navigator
- Driver types
- Navigator navigates
- Don‘t discuss too early
- Mob brings in ideas
- You can always refactor