TDD
Test-Driven Development

Jan Capasso

Why should we test?

- Spend less time debugging 🕑
- Reduce fear of changing 😱
- Good documentation 📖
- Cyclic feedback -> can help for design decisions ⟳

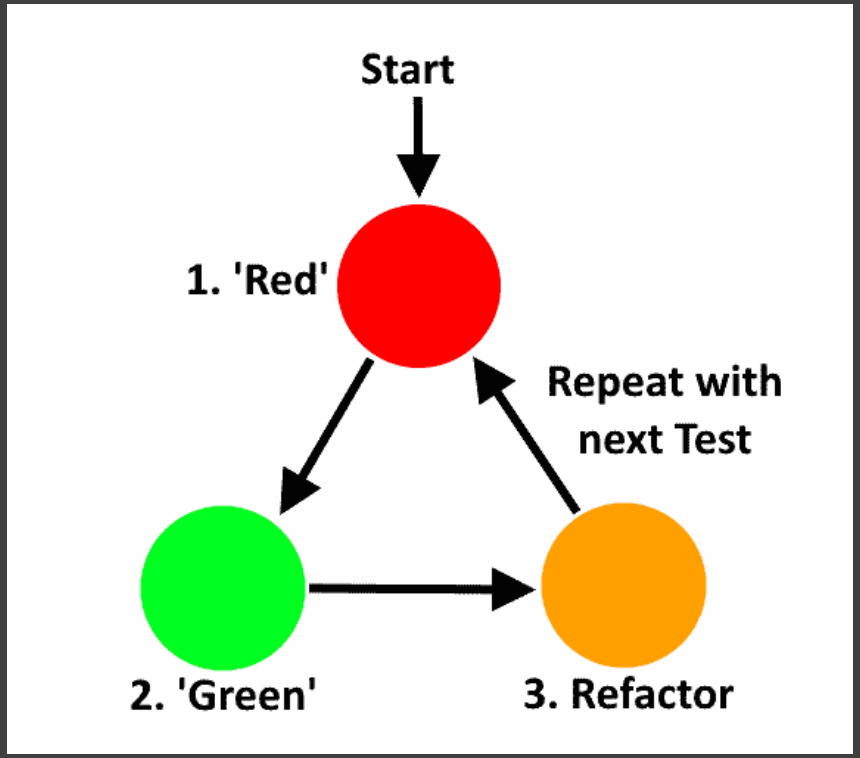
What we test?

- Test behavior
- Give a test a meaningful name
- Avoid
  - using technical names
  - leaking implementation details
  - writing technical tests - test de behavior!

```java
public class ShoppingCardShould {

    @Test
    void haveSize1WhenAddToCardCalled(){
        ShoppingCard shoppingCard = new ShoppingCard();
        Product item = new Product( test: "Test");

        shoppingCard.addToCard(item);

        assertEquals( expected: 1, shoppingCard.getList().size());
    }

    @Test
    void haveOneProductInWhenAdded(){
        ShoppingCard shoppingCard = new ShoppingCard();
        Product item = new Product( test: "Test");

        shoppingCard.addToCard(item);

        assertEquals( expected: 1, shoppingCard.size());
    }

}
```

How we test?

- Follow the red - green - refactor cycle
- Red phase -> Write a failing test
- Green phase -> Write enough to pass the test
- Refactor phase -> Refactor the code

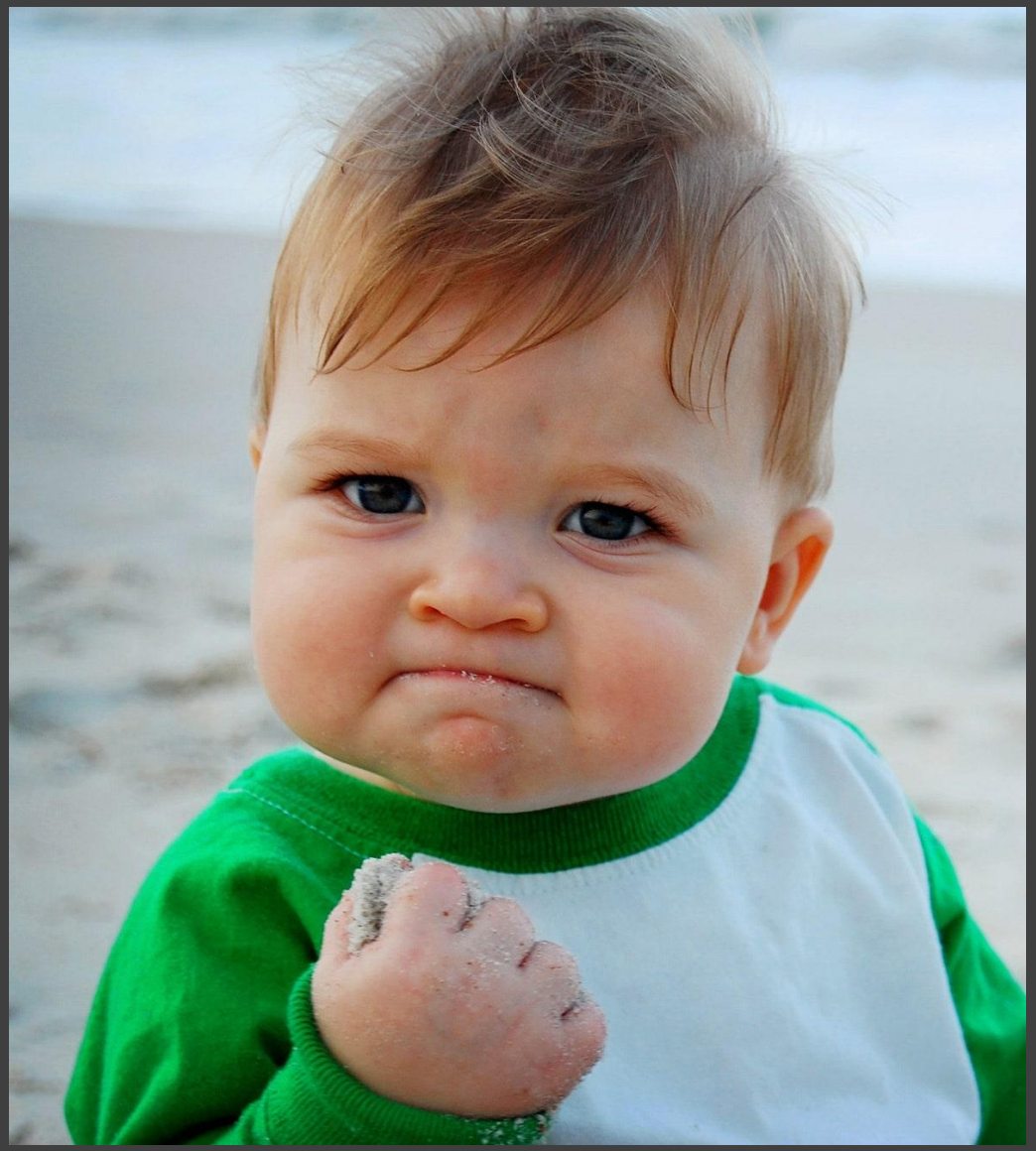


Red phase

- Write a failing test
- See the test fail for the right reason
- Write the assertations first a work backwards
- Ensure the feedback of the failing test is meaningful

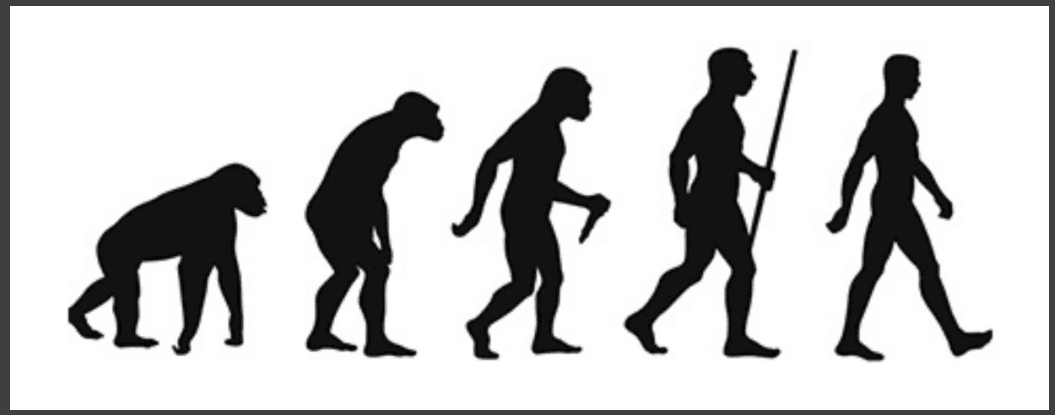
Green phase

- Write the simplest code to pass
  - Any code
  - Improvements on next stage
- Red -> green
  - Fake it
  - Obvious implementation
  - Triangulation



Refactor phase

- Refactor aggressively □□
- Refactor with the IDE
- Refactor production and test code independently
- Use the rule of three



Every TDD expert has small feet

- Make progress in small steps
- Not the fastest but the safest way
- Speed up the TDD process a lot
- Skip steps -> missing benefits

TDD in real life