Code retreat + TDD

Wòjcech Makùrôt

JUnit - résumé

- metoda testowa
  - nazwa
  - rodzaje metod
- sekcje
- sut, cut

```
@Test
public void toyotaEngineShouldBeRunningAfterIgnition() {
    // given
    Car sut = new CarFactory().forBrand(Brand.TOYOTA).build();
    // when
    sut.ignite();
    // then
    assertThat(sut.isRunning()).isTrue();
}
```

AssertJ (ew. Hamcrest) - résumé

- do czego?
- dlaczego?
- jak?

```
assertThat(actual).isEqualTo(expected);

assertThat("TEST").containsIgnoringCase("est");

assertThat(bank.getAccountFor("Wòjk")).is(vipAccount);
```

Mockito - résumé

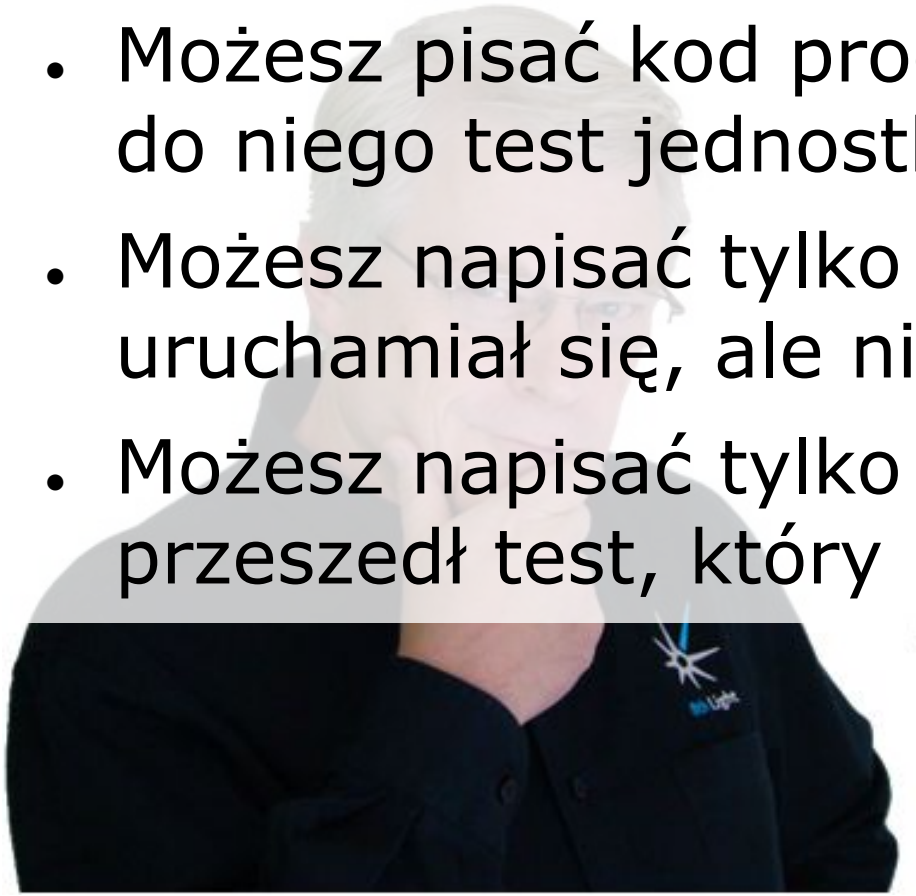
- do czego?
- dlaczego?
- jak?

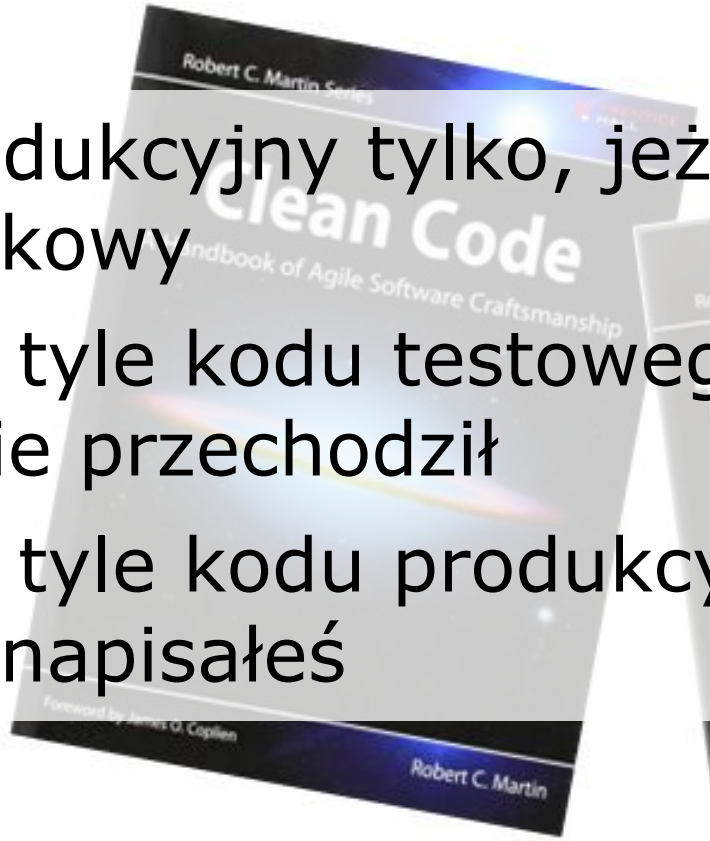
```
class Lover {
   private Flower flower;
   boolean checkLove() {...}
}
```

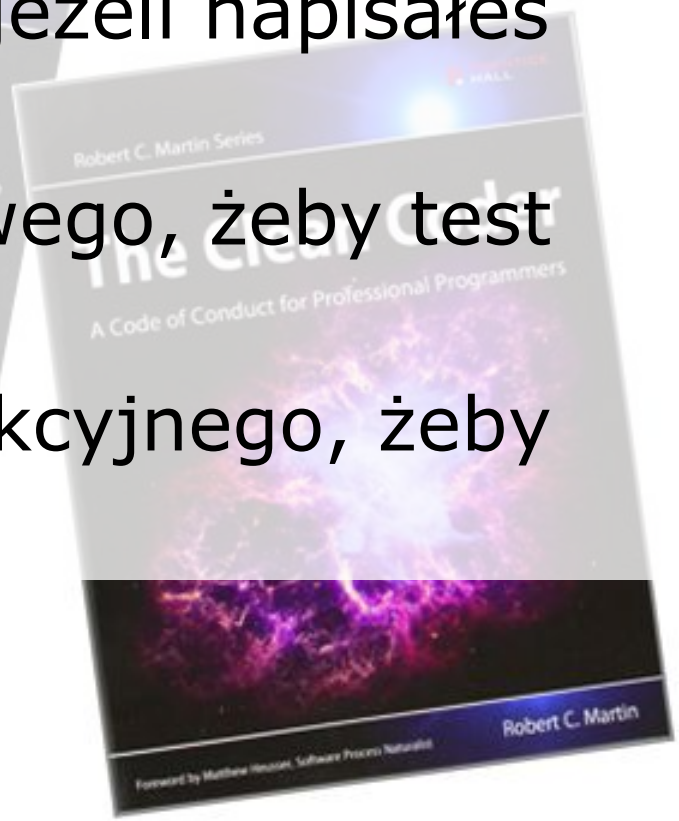
```
@RunWith(MockitoJUnitRunner.class)
public class LoverTest {

    @Mock
    private Flower flower;
    @InjectMocks
    private Lover sut;

    @Test
    public void shouldLoveForThreeLeavesFlower() {
        // given
        when(flower.getLeavesCount()).thenReturn(3);
        // when
        boolean result = sut.checkLove();
        // then
        assertThat(result).isEqualTo(true);
    }
}
```

Trzy zasady TDD

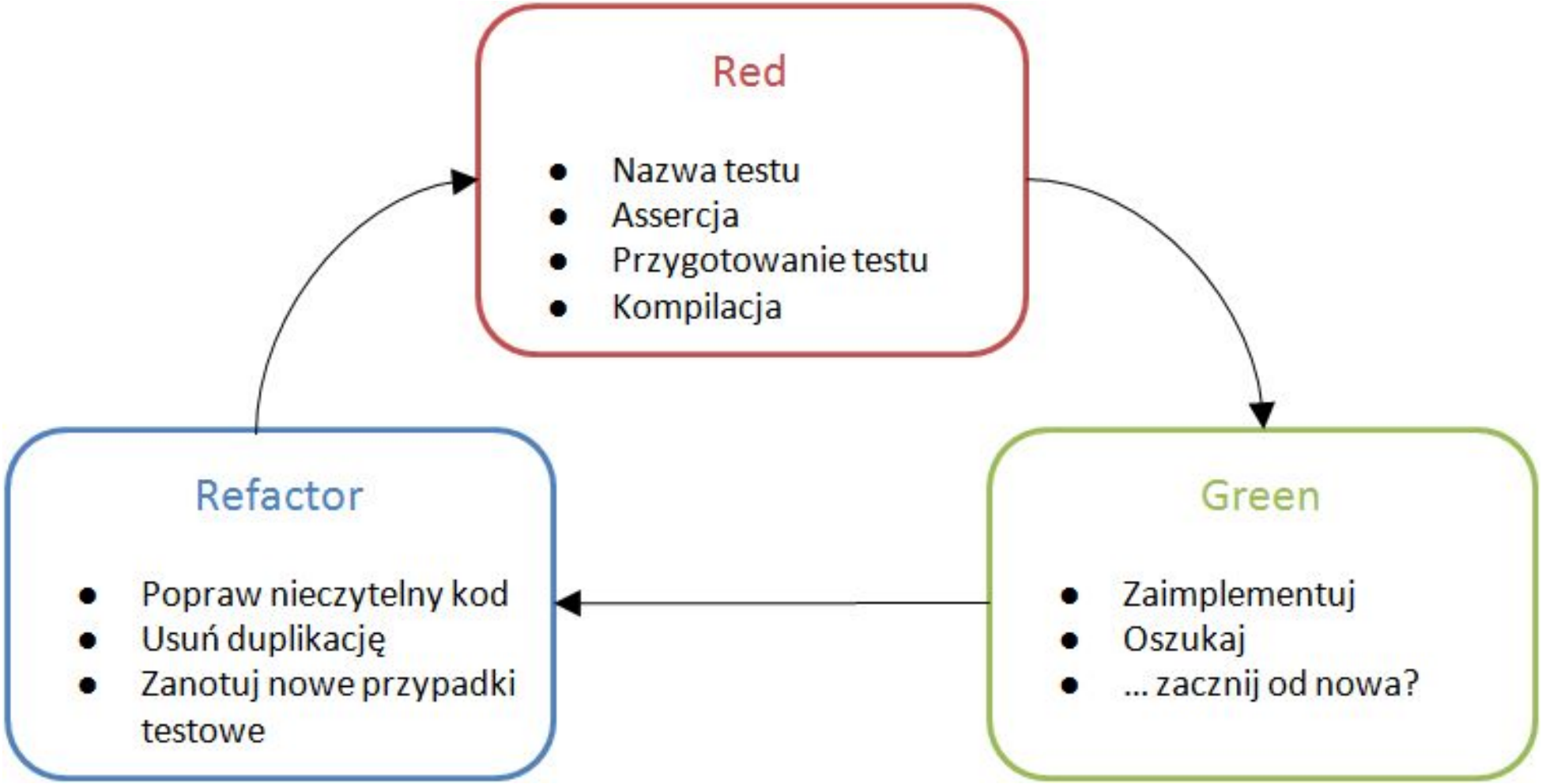
- Możesz pisać kod produkcyjny tylko, jeżeli napisałeś do niego test jednostkowy
- Możesz napisać tylko tyle kodu testowego, żeby test uruchamiał się, ale nie przechodził
- Możesz napisać tylko tyle kodu produkcyjnego, żeby przeszedł test, który napisałeś







Cykl TDD


Red
Nazwa testu
Assercja
Przygotowanie testu
Kompilacja
Green
Zaimplementuj
Oszukaj
... zacznij od nowa?
Refactor
Popraw nieczytelny kod
Usuń duplikację
Zanotuj nowe przypadki testowe

XP Simplicity Rules

- Do The Simplest Thing That Could Possibly Work

XP Simplicity Rules

- Simple code:
  - Runs all the tests
  - Contains no duplication
  - Expresses all the ideas you want to express
  - Minimizes classes, methods and modules

Coderetreat?

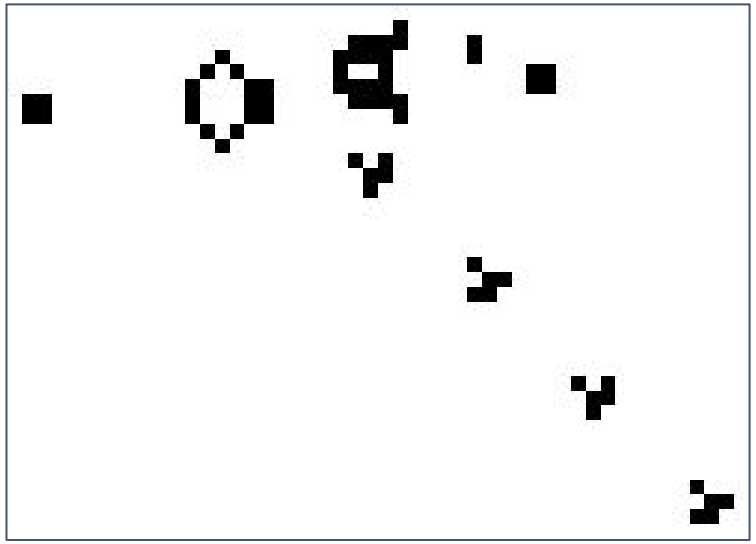
- programowanie w parach
- usuwamy kod
- podsumowanie
- zmiana pary
- zmienne reguły

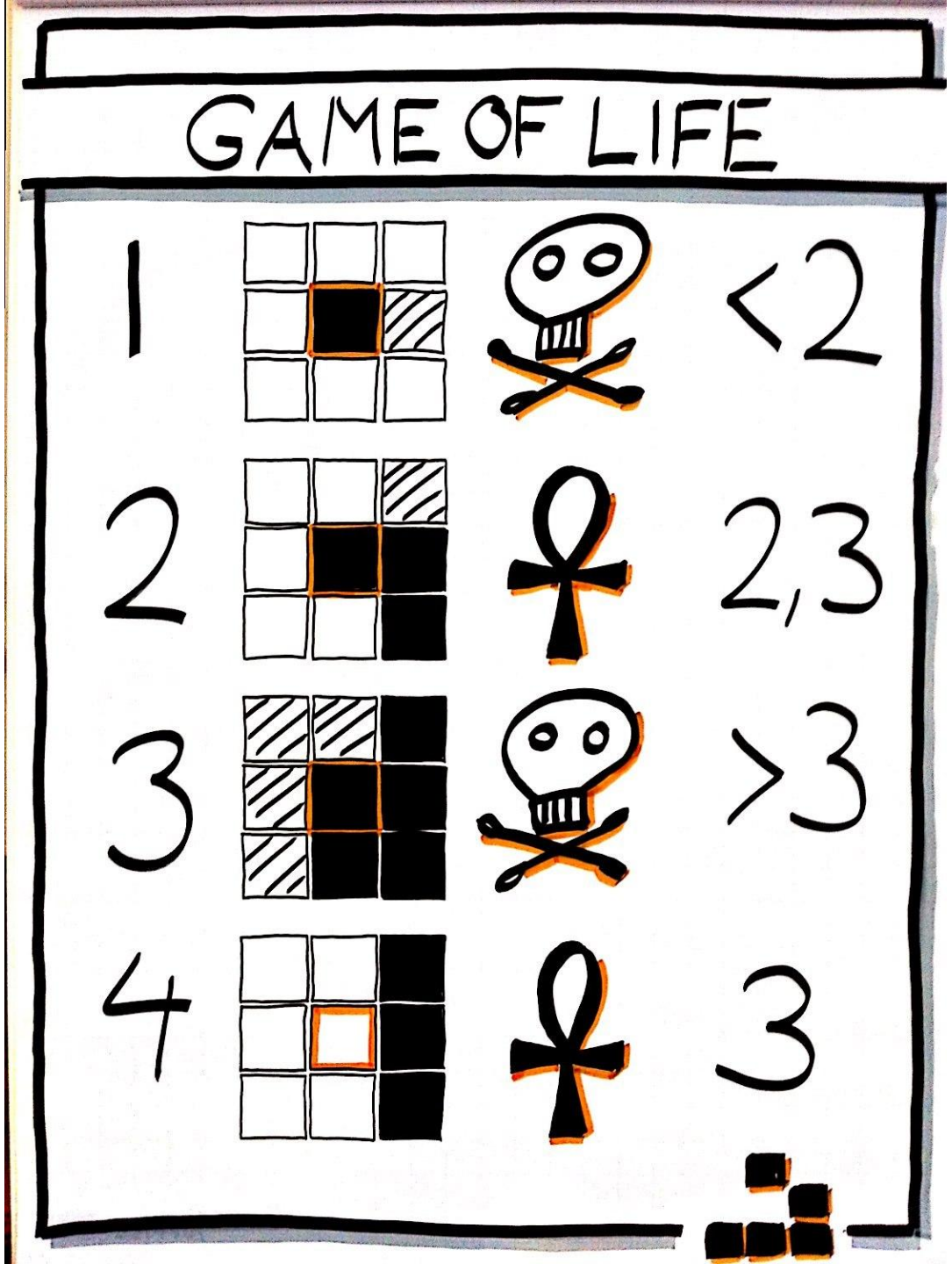
Zasady gry

- pary
  - z innej ławki
- 45 min. kodowania
  - TDD
- podsumowanie
- [przerwa]
- zmiana
  - im więcej zmian partnerów/partnerek, tym lepiej (sic!)

Conway's Game of Life

- plansza 2-wymiarowa [nieskończona]
- komórki żywe lub martwe
- interakcja z sąsiadami (8 szt.)
- gra bez gracza
  - stan inicjalny
  - generacje




GAME OF LIFE
1
<2
2
2,3
3
>3
4
3

Game of Life - runda 1.

	ŻYWA	MARTWA
. **0** żywych sąsiadów ->	umiera	martwa
. **1** żywy sąsiad ->	umiera	martwa
. **2** żywych sąsiadów ->	przeżywa	martwa
. **3** żywych sąsiadów ->	przeżywa	ożywa
. **4** żywych sąsiadów ->	umiera	martwa
. **5** żywych sąsiadów ->	umiera	martwa
. **6** żywych sąsiadów ->	umiera	martwa
. **7** żywych sąsiadów ->	umiera	martwa
. **8** żywych sąsiadów ->	umiera	martwa

Game of Life - runda 1.

Game of Life - runda 2.

- **P**omodoro **P**ing-**P**ong **P**air **P**rogramming
  - Red [A]
  - Green [B]
  - Refactor [A]

Game of Life - runda 2.

Game of Life - runda 3.

Game of Life - runda 3.

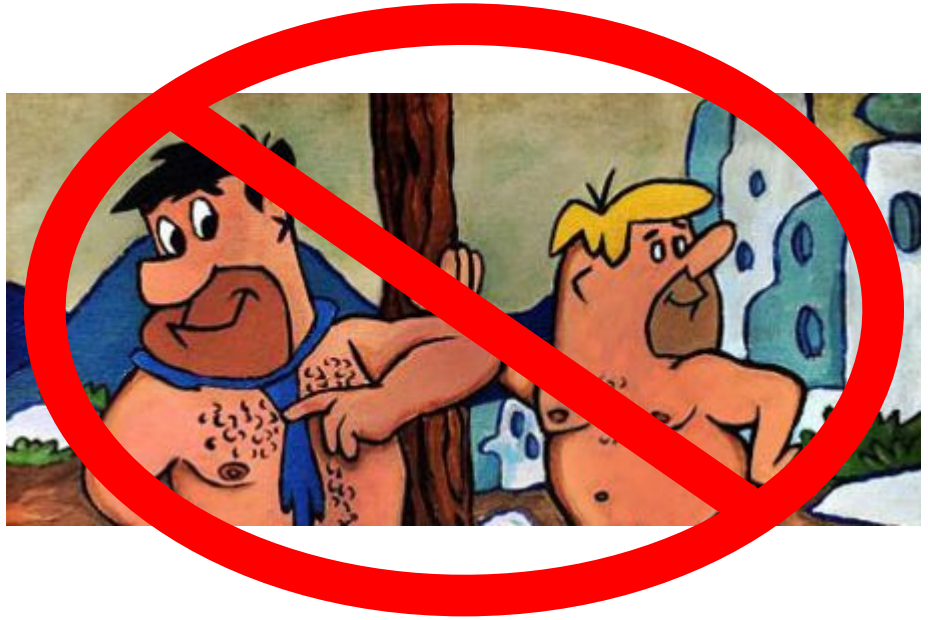
Game of Life - runda 4.

- metoda <5 linii
- niemutowalne obiekty

Game of Life - runda 4.

Game of Life - runda 5.

- bez widocznych nagich prymitywów
  - ~~boolean~~
  - ~~String~~
  - ~~liczby~~
  - ~~kolekcje~~
  - ~~Object~~



- czasowniki zamiast rzeczowników

Game of Life - runda 5.

Wnioski

- Nauczyłem/am się dziś...
- Zaskoczyło mnie dziś...
- W przyszłości będę inaczej...