Jak robimy to na backendzie

Przemyślenia programisty z przykładami

Robert Podsiadły

test:fest

tieto EVRY

whoami?

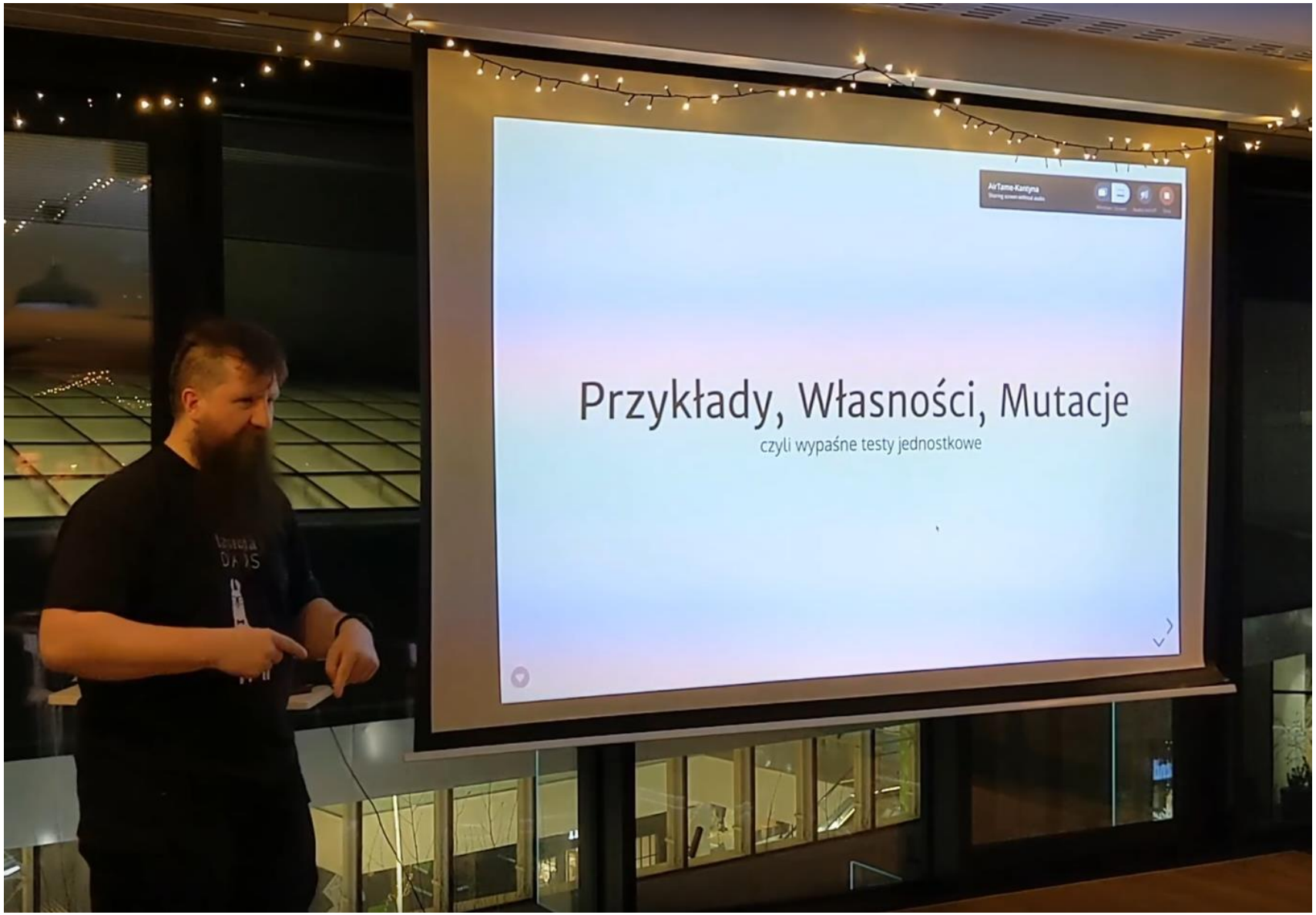
- W branży IT pracuję od 1986, w tym od czterech lat jako Java Software Engineer (Dev)
- Od ponad dwóch lat w TIETO
- Mój konik to backend związany z JVM
- Prywatnie mąż, ojciec, dziadek, miłośnik muzyki (jazzu), literatury, sportów wytrzymałościowych i przewodnik Golden Retrievera

whoareyou?

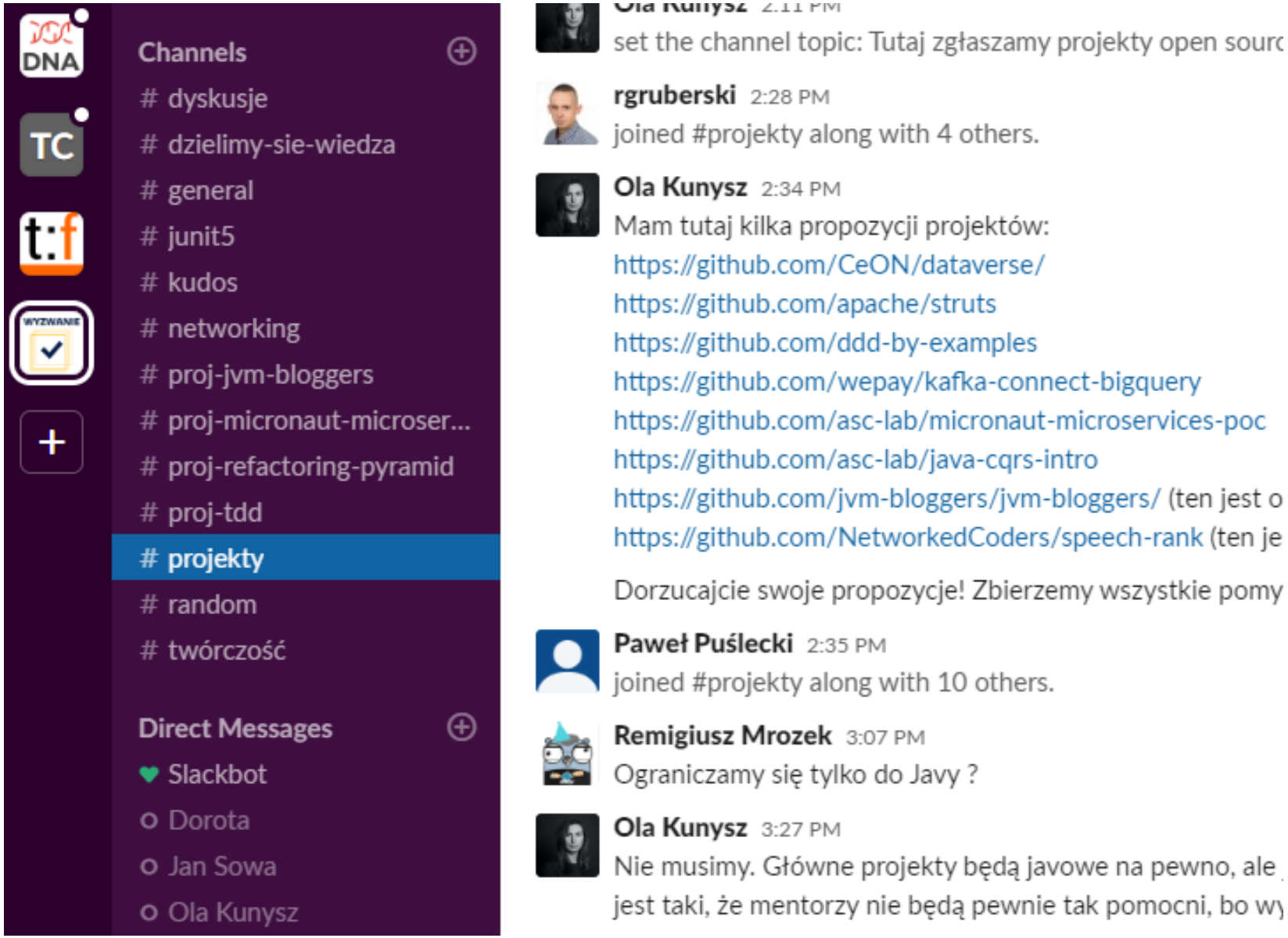
Co ja tutaj robię?

tieto EVRY

Duża część zaangażowania dev-community idzie w jakość (testowanie)


Przykłady, Własności, Mutacje
czyli wypaśne testy jednostkowe

https://szkolatestow.online/wyzwanie



wyzwanie.slack.com

Projekt typu UBER

- Taka rura przetwarzająca zdarzenie od wejścia do wyjścia, z przystankami na poszczególnych warstwach – kolejkach.
- Start projektu to test E2E sprawdzający czy przechodzi zdarzenie – rodzaj smoke testu
- Testy sprawdzające co dzieje się pomiędzy warstwami – testy integracyjne

Projekt typu sterowanie siecią energetyczną

- Niezawodność

- Trzy poziomy testów:
  - stuby, mocki zastępujące hardware na poziomie developera
  - Framework testowy na poziomie merge do mastera
  - Prawdziwy hardware na poziomie przygotowania release.

- Najlepsi developerzy są w zespołach testerskich

Przykładowy kod

Założenia:

- Prosty, funkcjonalny, bez infantylnych: boo, foo itd.
- Można na nim zastosować większość testów, o których chciałbym mówić.
- Napisana przeciętnie z błędami, żeby QA miał gdzie się wyżyć

RESTowa aplikacja alkomatu

- Oparta na SJA 10 g (lub 12,5 ml) czystego alkoholu etylowego

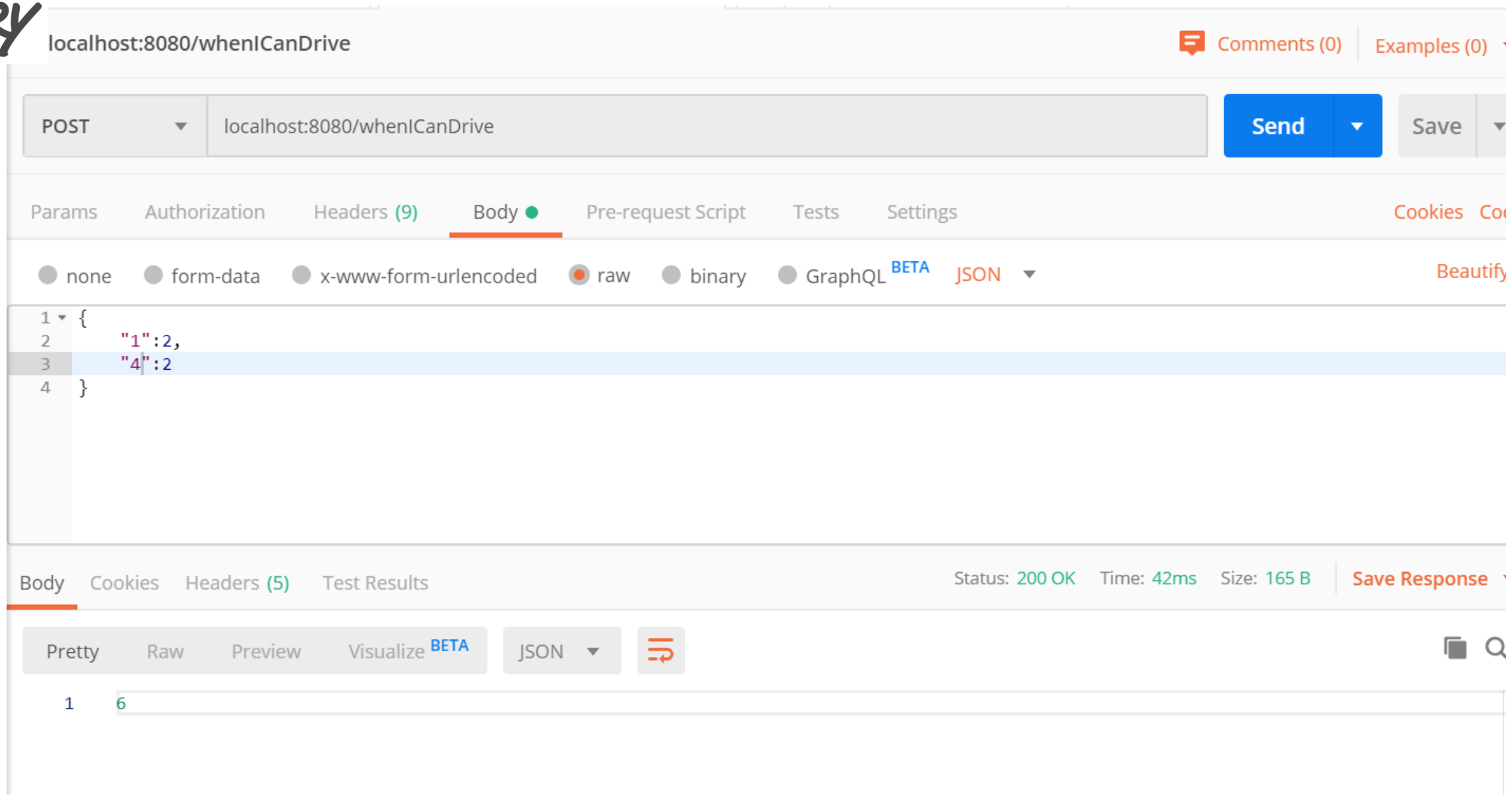
- Małe piwo – 330 ml piwa – 1,19 SJA – spalamy ~1h15m,

- Duże piwo – 500 ml piwa – 1,8 SJA – spalamy ~2h,

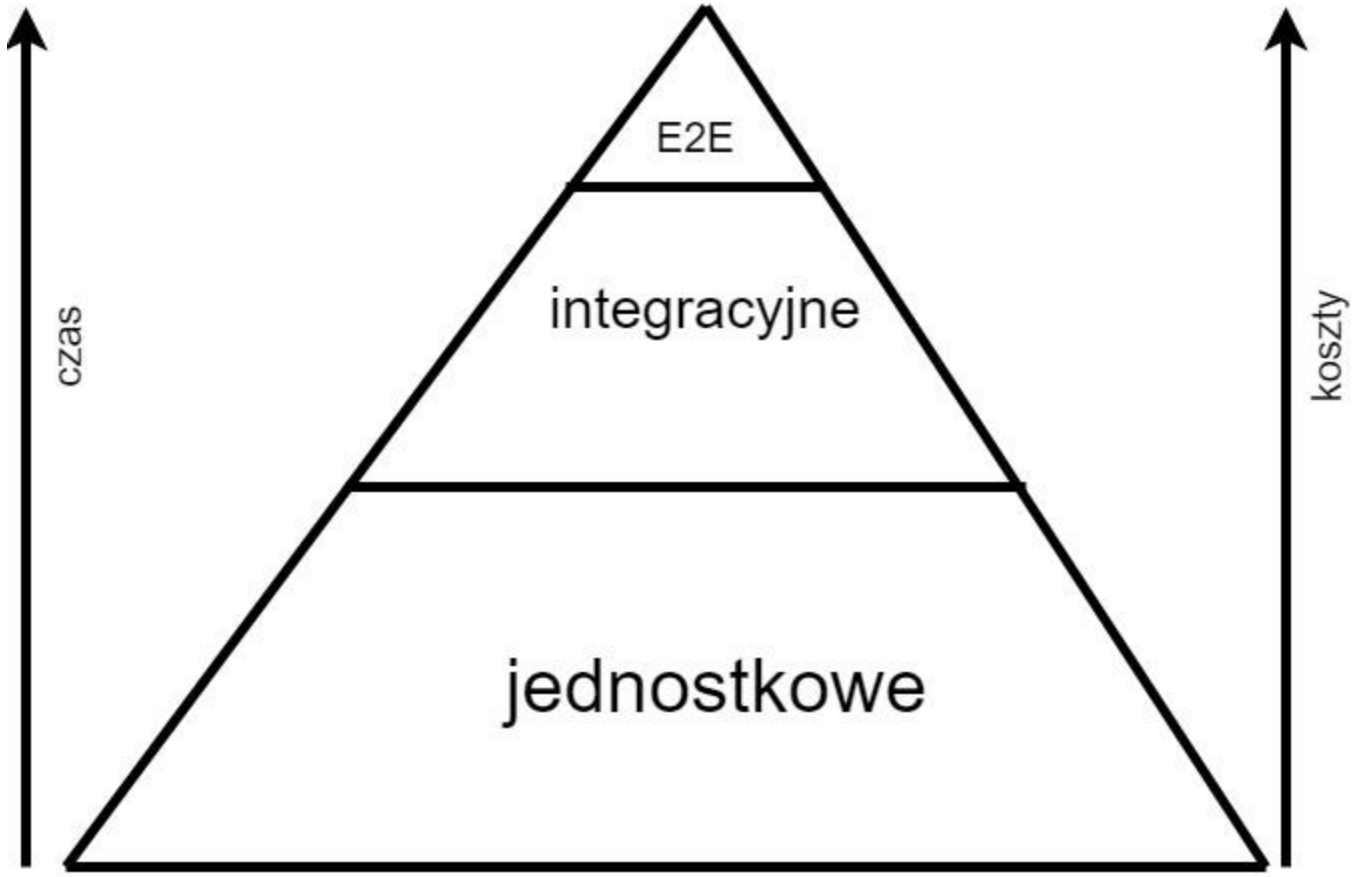
- Kieliszek wina – 175 ml wina – 1,68 SJA – spalamy ~2h,

https://pl.wikipedia.org/wiki/Standardowa_dawka_alkoholu

Stosujecie na własną odpowiedzialność!!!


localhost:8080/whenICanDrive
Comments (0)
Examples (0)
POST
localhost:8080/whenICanDrive
Send
Save
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL BETA
JSON
Beautify
{
"1":2,
"4":2
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 42ms
Size: 165 B
Save Response
Pretty
Raw
Preview
Visualize BETA
JSON
6

Mit 1: piramida testów



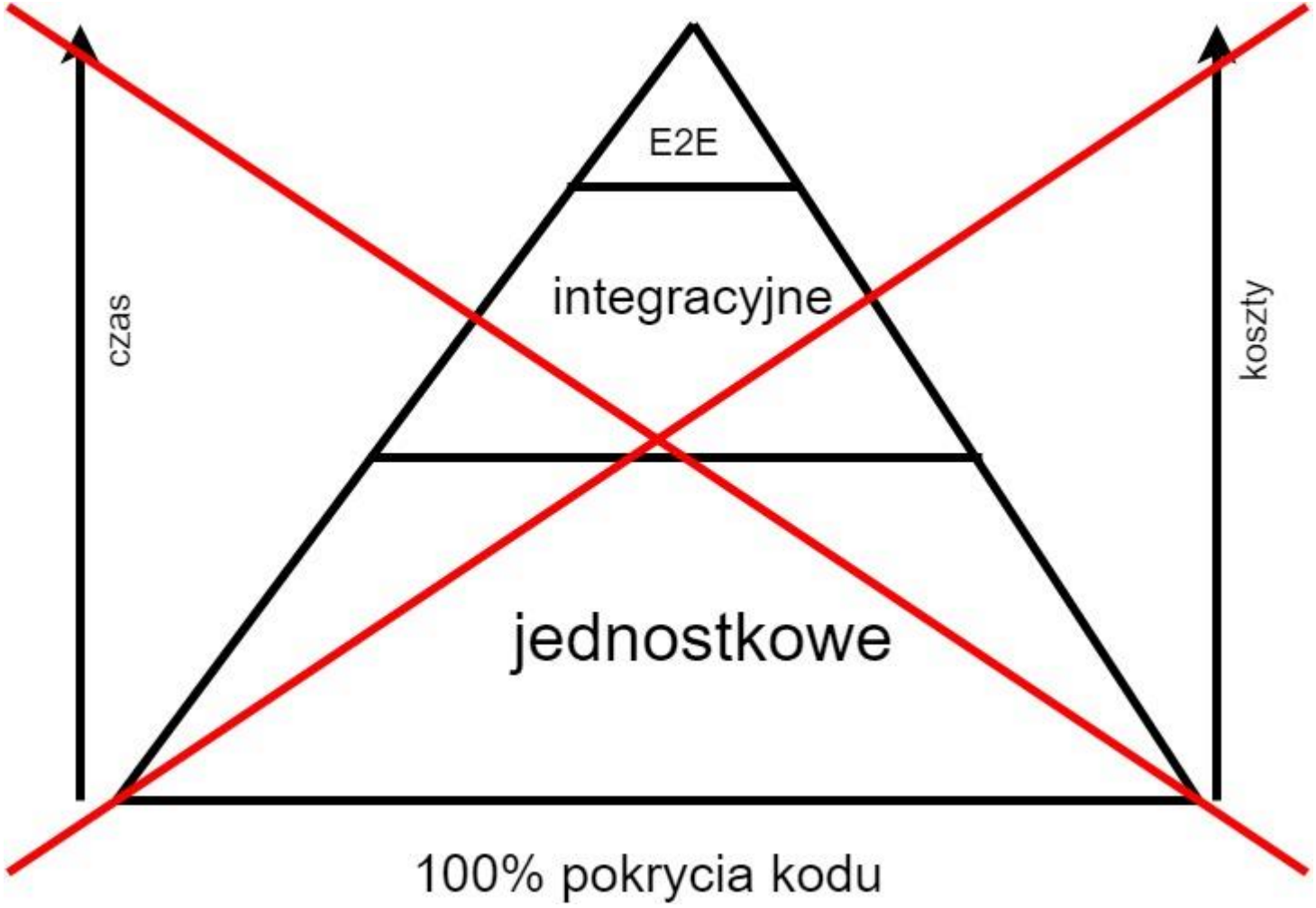
Mit 2: pokrycie kodu

Dobry programista pokrywa testami 100% kodu!

NIE

!!!!!!

Mitom mówimy NIE



tieto EVRY

Dlaczego testujemy (jednostkowo)?

- Uruchomienie funkcjonalności (metody, modułu) jest najszybsze poprzez test, dlatego warto napisać test nawet na chwilę
- Jako sprawdzenie działania kodu, szczególnie algorytmów, wyrażeń regularnych, złożonej logiki
- Dokumentacja działania metod danej klasy. Doc się zawsze rozjedzie, testy nie powinny

Uwaga: pokrywamy testami wszystkie znalezione błędy!

Testy jednostkowe – czego unikamy

- Największym zagrożeniem są testy „cementujące” kod,
- Unikamy testowania geterów/seterów i innych ceremonii,
- Piszemy interfejsy i testujemy interfejsy, nie metody implementacji,
- W miarę możliwości testujemy zachowania a nie klasy

Dlaczego testujemy (integracyjnie)?

- Czy to powinno współpracować, ma ochotę współpracować (np. czy działa serializacja / deserializacja)
- Testy obejmujące wiele warstw abstrakcji
- Refaktoring!!!!!!!!!!!!!!!
- Testy regresji

Dlaczego testujemy (E2E)?

- Zapewnienie realizacji DoD, pokazanie kontraktu,
- Jako element przygotowania do CI/CD,
- Jako smoke tests czy health check tests,
- Im lepsze testy wysokiego poziomu, tym lepiej się współpracuje z innymi zespołami.

Inne podejście a może przerzucić testowania na klienta

- Wrzucamy na produkcję małe zmiany i szybko revertujemy jak coś idzie nie tak
- Testy A/B
- Testy kanarkowe itd.

Co jeszcze testujemy

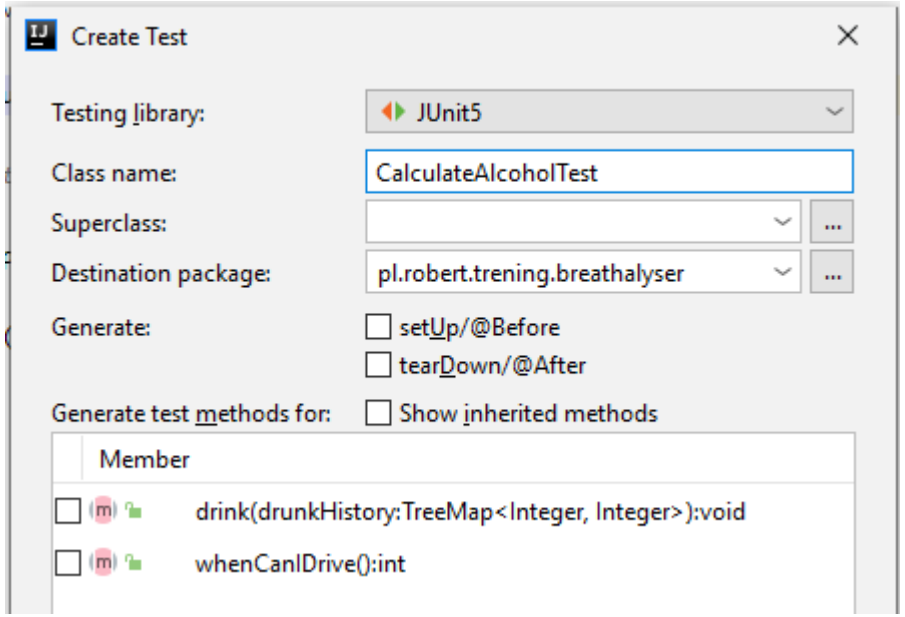
- Wydajność
- Zapotrzebowanie na zasoby
- Wycieki pamięci (zasobów)
- I wiele innych

Skąd się biorą ~~taski w Jirze?~~

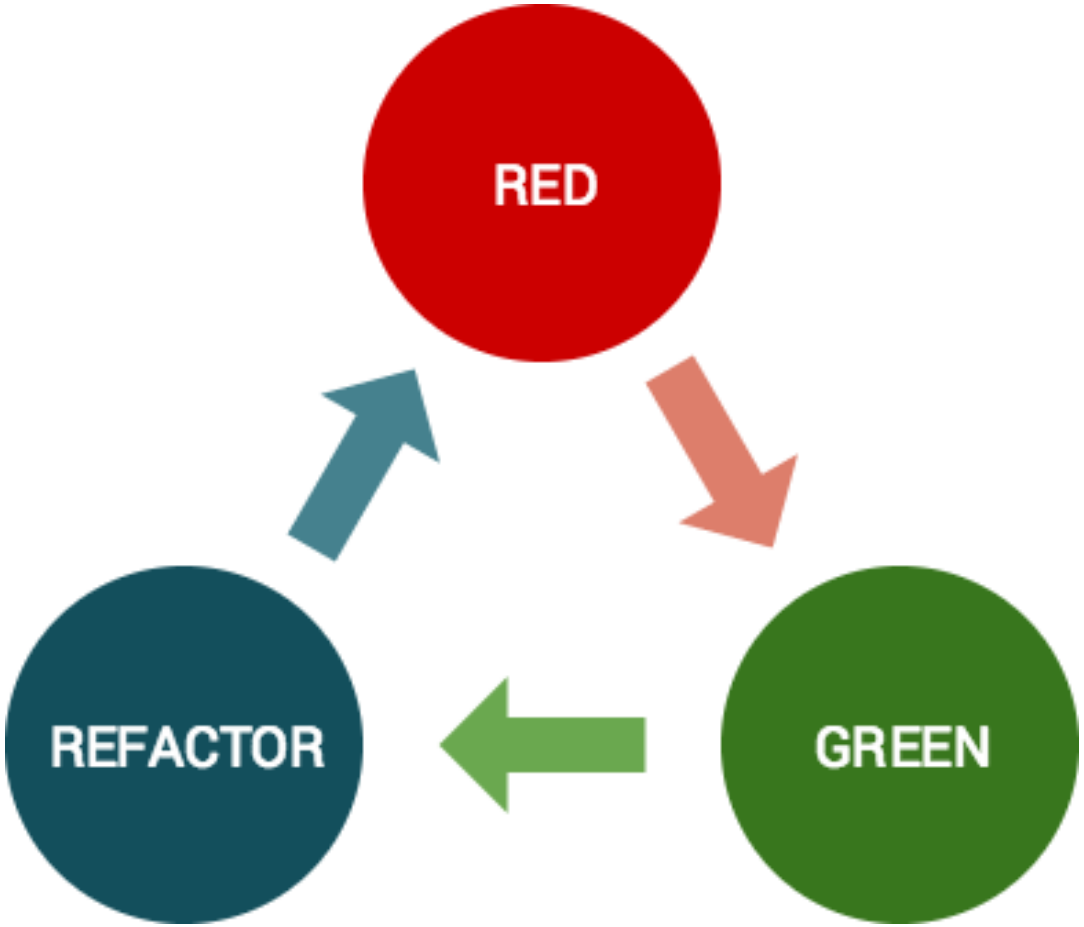
TESTY?

Testy jednostkowe, przez bezrefleksyjne klikanie po IDE

```java
public interface CalculateAlcohol {

//    void drink(int time, StandardDrink alco);

    void drink(TreeMap<Integer, Integer> drunkHistory);

    int whenCanIDrive();
}
```



TDD



Testy jednostkowe a testy integracyjne

```java
@Test
void whenCanIDriveDrink1l2p2h() {
    // Given
    CalculateAlcohol calculateAlcohol = new CalculateAlcoholImpl();
    TreeMap<Integer, Integer> drinkingHistory = new TreeMap<>();
    drinkingHistory.put(1, 10);
    drinkingHistory.put(2, 10);
    // When
    calculateAlcohol.drink(drinkingHistory);
    // Then
    assertEquals( expected: 21, calculateAlcohol.whenCanIDrive(), message: "1l2p2h");
}
```

```java
@Test
public void whenCanIDriveDrink1l2p2h() throws Exception {
    // Given
    TreeMap<Integer, Integer> drinkingHistory = new TreeMap<>();
    drinkingHistory.put(1, 10);
    drinkingHistory.put(2, 10);
    // When
    int iCanDrive = callEndpoint(drinkingHistory);
    // Then
    assertEquals( expected: 21, iCanDrive, message: "1l2p2h");
}
```

```
    @Override
    public int whenCanIDrive() {
        TreeMap<Integer, Integer> timeTable = drinkingHistory.getTimeTable();
        if (timeTable.size() == 0) {
            return 0;
        }

        int maxKey = timeTable.lastKey();
        while (!canIDrive(maxKey)) {
            ++maxKey;
        }
        return maxKey;
    }

    private boolean canIDrive(int time) {
        TreeMap<Integer, Integer> timeTable = drinkingHistory.getTimeTable();
        if (timeTable.size() == 0) {
            return true;
        }

        return Stream
                .iterate(new int[]{0, timeTable.get(0) == null ? 0 : timeTable.get(0)},
                        p -> new int[]{p[0] + 1,
                                p[1] + (p[1] > 0 ? -1 : 0) + (timeTable.get(p[0] + 1) == null ? 0
                                        : timeTable.get(p[0] + 1))
                        })
                .limit((long) time + 1)
                .skip(time)
                .findFirst()
                .get()[1] == 0;
    }

    private void inputDataValidation(TreeMap<Integer, Integer> drunkHistory) {
        for (Map.Entry<Integer, Integer> entry : drunkHistory.entrySet()) {
            if (entry.getKey() < 1 || entry.getKey() > 24)
                throw new IllegalArgumentException("We can drink 24 hours, start in 1");
            if (entry.getValue() < 1)
                throw new IllegalArgumentException("Cola is not alcohol");
            if (entry.getValue() > 20)
                throw new IllegalArgumentException("You've drunk too much, you're dead");
        }
    }
}
```

Testy parametryczne

```
@ParameterizedTest
@MethodSource("createTreeMaps")
void whenCanIDriveDrink1I2p2h(TreeMap<Integer, Integer> drinkingHistory, int hour) {
    // Given
    CalculateAlcohol calculateAlcohol = new CalculateAlcoholImpl();
    // When
    // Then
    assertThrows(IllegalArgumentException.class, () -> {
        calculateAlcohol.drink(drinkingHistory);
    });
}
```

Property testing, testy sterowane właściwościami

- Właściwość (dla x(dodatnie) + y(dodatnie))

  suma jest WIĘKSZA od każdego ze składników
- Jeżeli wypiję w momencie t jednego drinka o mocy a

  → 0% ma po t + a godzinach
- Jeżeli wypiję dużo drinków

  → wynik pozytywny zawiera się w granicach

  min: t(ostatniego drinka) + a(jego moc)

  max: suma SJA wszystkich wypitych drinków

Property testing, testy sterowane właściwościami

```java
@Property
void simplePositiveProperties(@ForAll @IntRange(min = 1, max = 24) int key,
                              @ForAll @IntRange(min = 1, max = 20) int value) {
    // Given
    CalculateAlcohol calculateAlcohol = new CalculateAlcoholImpl();
    // When
    TreeMap<Integer, Integer> drinkingHistory = new TreeMap<>();
    drinkingHistory.put(key, value);
    calculateAlcohol.drink(drinkingHistory);
    // Then
    int hoursForSobriety = calculateAlcohol.whenCanIDrive();
    System.out.println("t = " + key + "\tunit = " + value + "\thours = " + hoursForSobriety);
    assertEquals( expected: key + value, calculateAlcohol.whenCanIDrive());
}
```

```
t = 9   unit = 14   hours = 23
t = 13  unit = 14   hours = 27
t = 10  unit = 19   hours = 29
t = 10  unit = 19   hours = 29
t = 4   unit = 8    hours = 12
t = 4   unit = 15   hours = 19
t = 24  unit = 8    hours = 32
t = 24  unit = 3    hours = 27
t = 5   unit = 10   hours = 15
t = 4   unit = 7    hours = 11
```

Testy mutacyjne

- Skończony kod
- Skończone testy
- Wprowadzamy mikrozmiany (mutacje) do kodu produkcyjnego
- Dla każdej mikrozmiany wywołujemy zbiór testów (do pierwszego błędu)

Testy mutacyjne

- X + Y => X – Y
- If (y > x) {…} => If (y >= x)
- List<String []> listaStr = new ArrayList<>();

  => List<String []> listaStr = null;

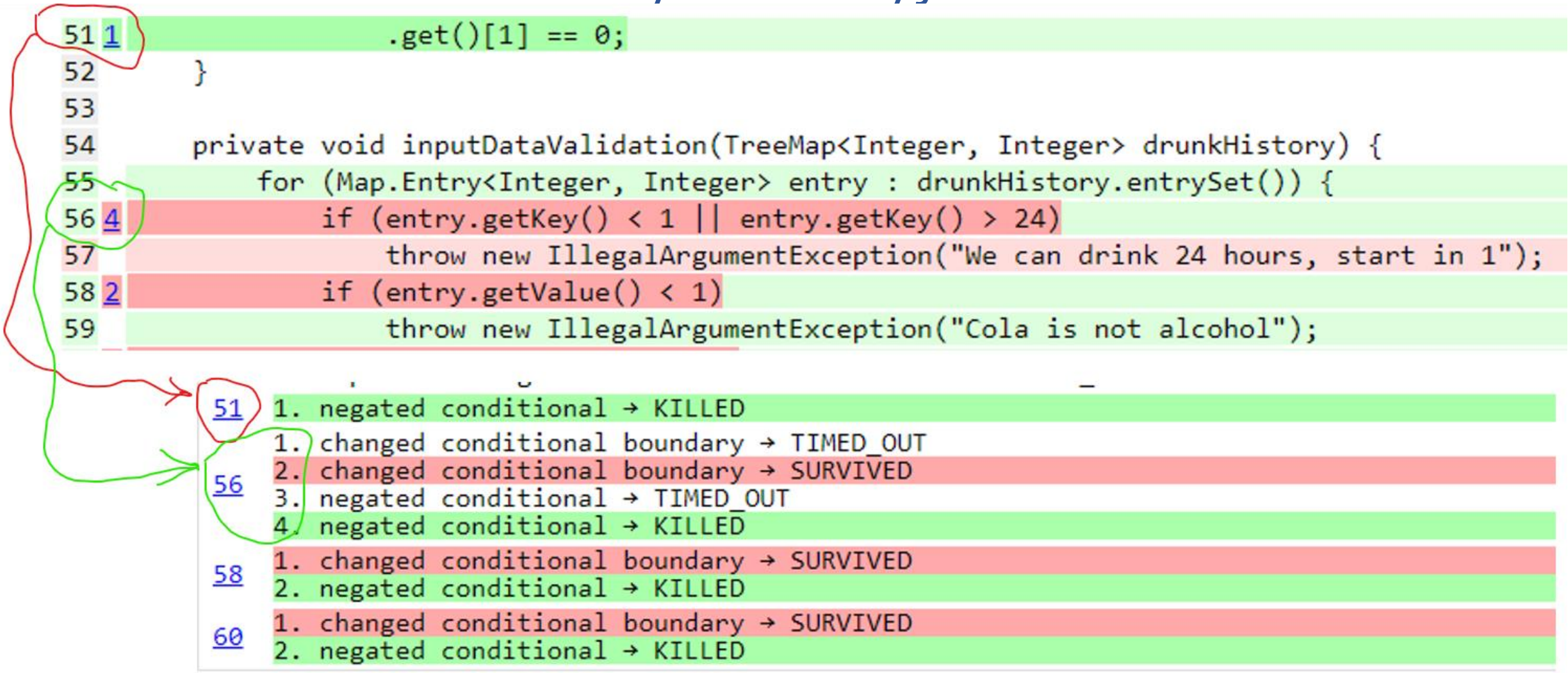
Active mutators

- CONDITIONALS_BOUNDARY_MUTATOR
- INCREMENTS_MUTATOR
- INVERT_NEGS_MUTATOR
- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR
- RETURN_VALS_MUTATOR
- VOID_METHOD_CALL_MUTATOR

Testy mutacyjne

```
================================================================
- Statistics
================================================================
-> Generated 34 mutations Killed 30 (88%)
-> Ran 36 tests (1.06 tests per mutation)
```

Testy mutacyjne



Proces tworzenia projektu

DDD +

=> BDD + Gherkin => DoD + TESTS

Take away!

- Nie konkurujmy tylko uzupełniajmy się!
  Strzelamy do tej samej bramki!

- Mamy różne narzędzia ale razem odpowiadamy za jakość!

- Nie tolerujcie developerów, którzy nie piszą testów

Dziękuję!

Robert Podsiadły

- Email: rob.podsiadly@gmail.com
- Twitter: @robert59p
- LinkedIn: https://www.linkedin.com/in/robertpodsiadly/
- Kod do prezentacji: https://github.com/RobertPod/breathalyser

Materiały – dokumentacja:

- https://junit.org/junit5/
- https://koziolekweb.pl/junit-5/
- https://jqwik.net/
- https://pitest.org/
- https://cucumber.io/docs/gherkin/
- https://www.eventstorming.com/

Materiały – nagrania, blogi:

Wroclaw JUG: https://wroclaw.jug.pl/ (dwa osatanie - dostępne)

- WrocławJUG & DataArt: "Testy jednostkowe w Javie", Ola Kunysz https://youtu.be/43Ox4OaNmJ8
- 150. WrocławJUG [EASY, PL] - Testy jednostkowe na pełnym wypasie - Bartek Kuczyński https://youtu.be/DEDpjYCqTyo

Ola Kunysz:

- https://olaqnysz.blogspot.com/
- https://blog.szkolatestow.online/
- https://szkolatestow.online/wyzwanie