

Praktyczne Aspekty Rozwoju Oprogramowania: TDD na przykładzie C++

Hanna Senhadri (hanna.senhadri@nokia.com)
Michał Orynicz (michal.orynicz@nokia.com)

The Nokia logo, consisting of the word "NOKIA" in a white, sans-serif font, centered within a large white circle.

Kwestie organizacyjne



Kwestie organizacyjne...

- Uczestnicy kursu otrzymają dawkę wiedzy z zakresu rozwoju oprogramowania wraz z przykładami zastosowań praktycznych – **część teoretyczna + praktyczna zajęć**
- Pytania w trakcie zajęć – mile widziane
- Czy powinniśmy robić przerwę?
- Na zajęciach przysługują dodatkowe punkty za aktywność i rozwiązanie zadania (+1, +2):
 - Wykonanie przykładu/rozwiązania publicznie
 - Pytania i odpowiedzi
 - Rozwiążanie zadania przesłane na maila w dniu zajęć



Kwestie organizacyjne

...a zdalna forma zajęć

- Sprawdzenie listy obecności – **na podstawie Teams**
- Okno czatu – pomagamy i odpowiadamy na pytania!
- Osoby wypowiadające się: prosimy o oznaczenie się na czacie (punkty za aktywność)
- Możemy współdzielić/pokazywać kod przez dedykowany link:
 - <https://retester.com/>
 - <https://godbolt.org/>
 - ...
- Propozycje dobrych praktyk z poprzednich zajęć?



Czym jest UT?



Testy w życiu codziennym

Czy równo zaparkowałem?

Czy moja marynarka pasuje do butów?

Czy efekt moich działań jest zgodny z oczekiwaniami?

Czy farba mi się gdzieś nie rozpuściła?

Nie przesoliłem?

Czy moja postać brzmi zgodnie z jej charakterem?

Slide 6

HS(0 **[@Michał Oryńicz (Nokia)]** slajd do wypełnienia
Hanna Senhadri (Nokia); 2023-04-04T09:57:39.711

Testy w życiu codziennym

Czy efekt moich działań jest zgodny z oczekiwaniami?

Poziomy testów

Testy end-to-end



Testy jednostkowe

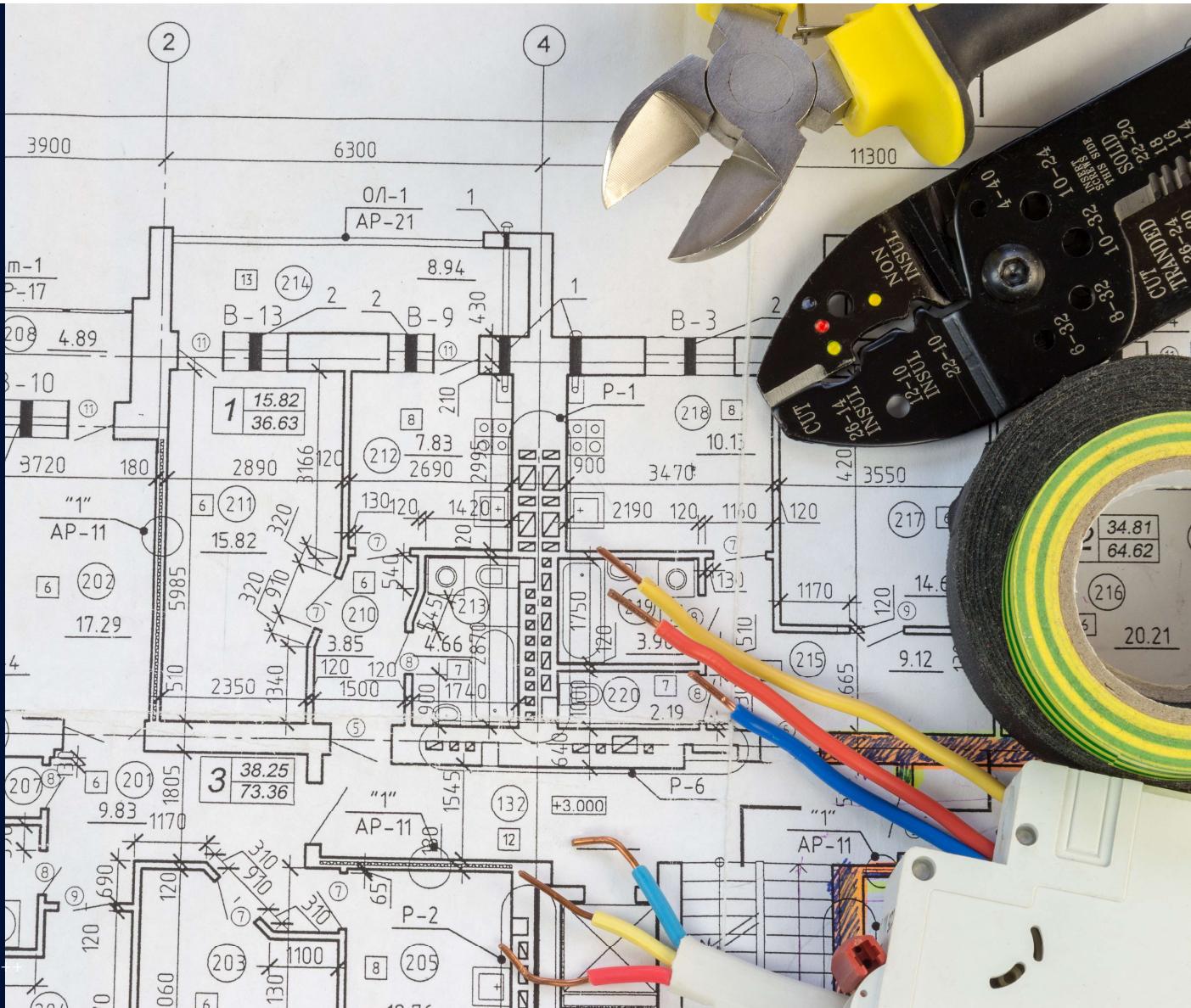


Testy integracyjne



Test jednostkowy

- Co zwróci moja funkcja/metoda dla takich parametrów?
- Jak zachowa się mój obiekt po takiej sekwencji wydarzeń?



Czym jest TDD?



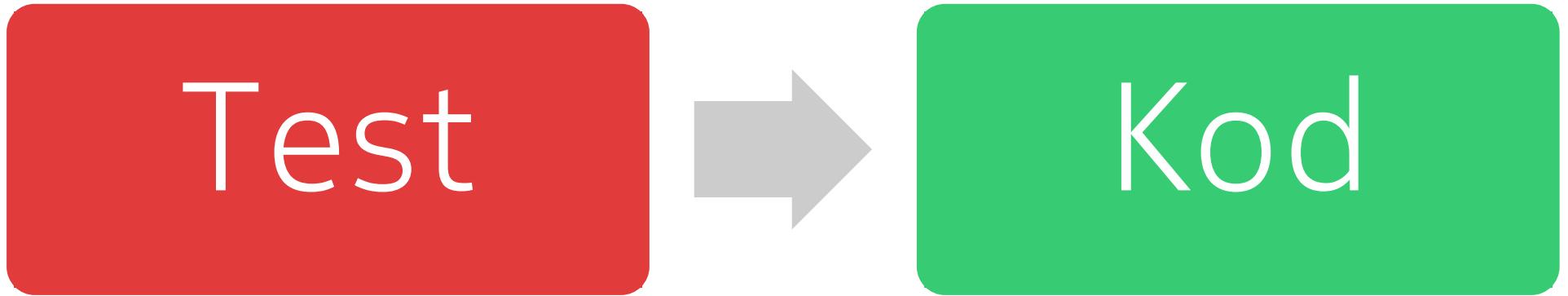
Test Driven Development



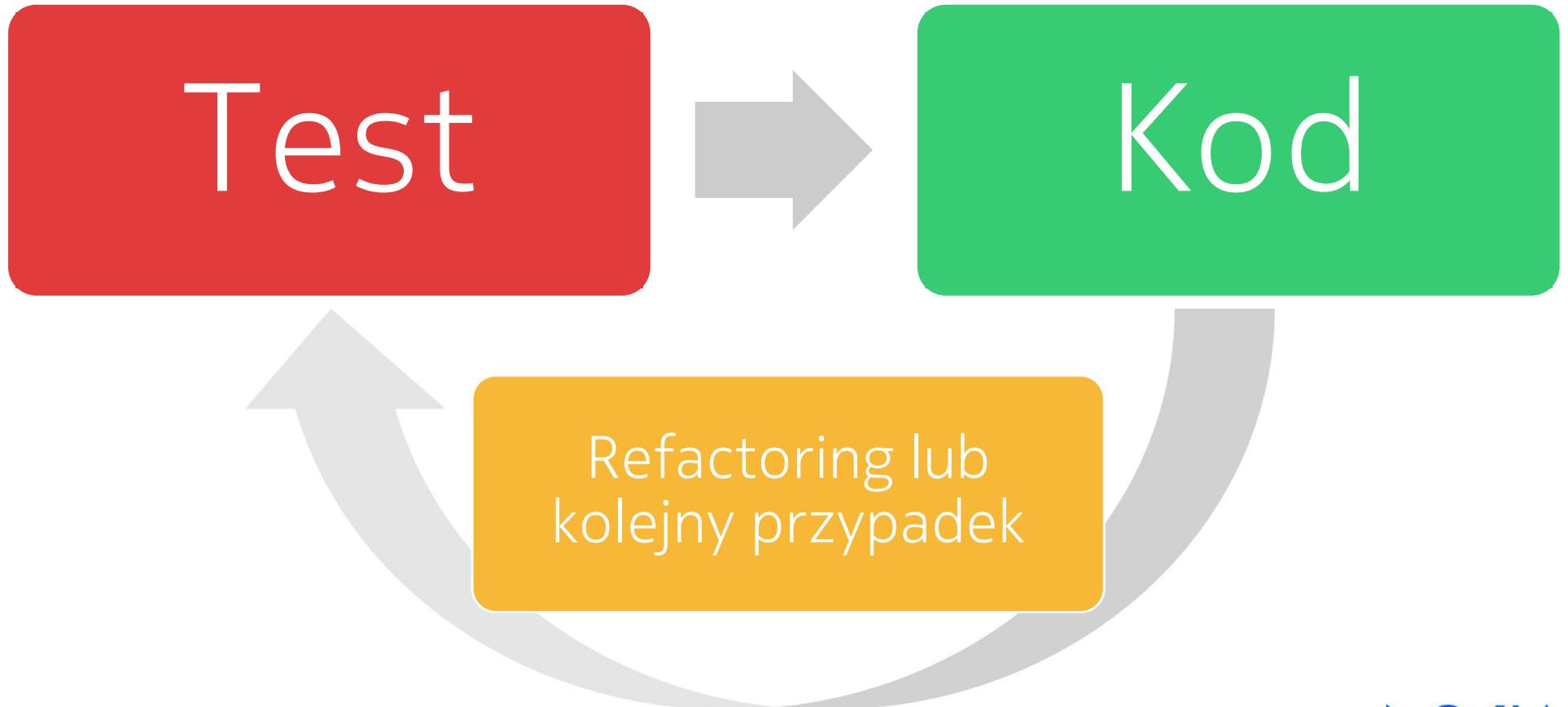
Zasady pracy w TDD

Test

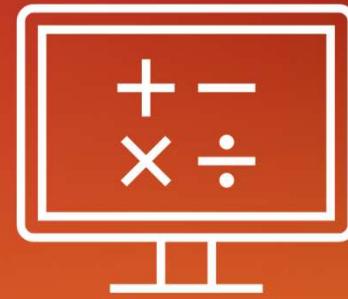
Zasady pracy w TDD



Zasady pracy w TDD



Zadanie na dziś



Zadanie na dziś

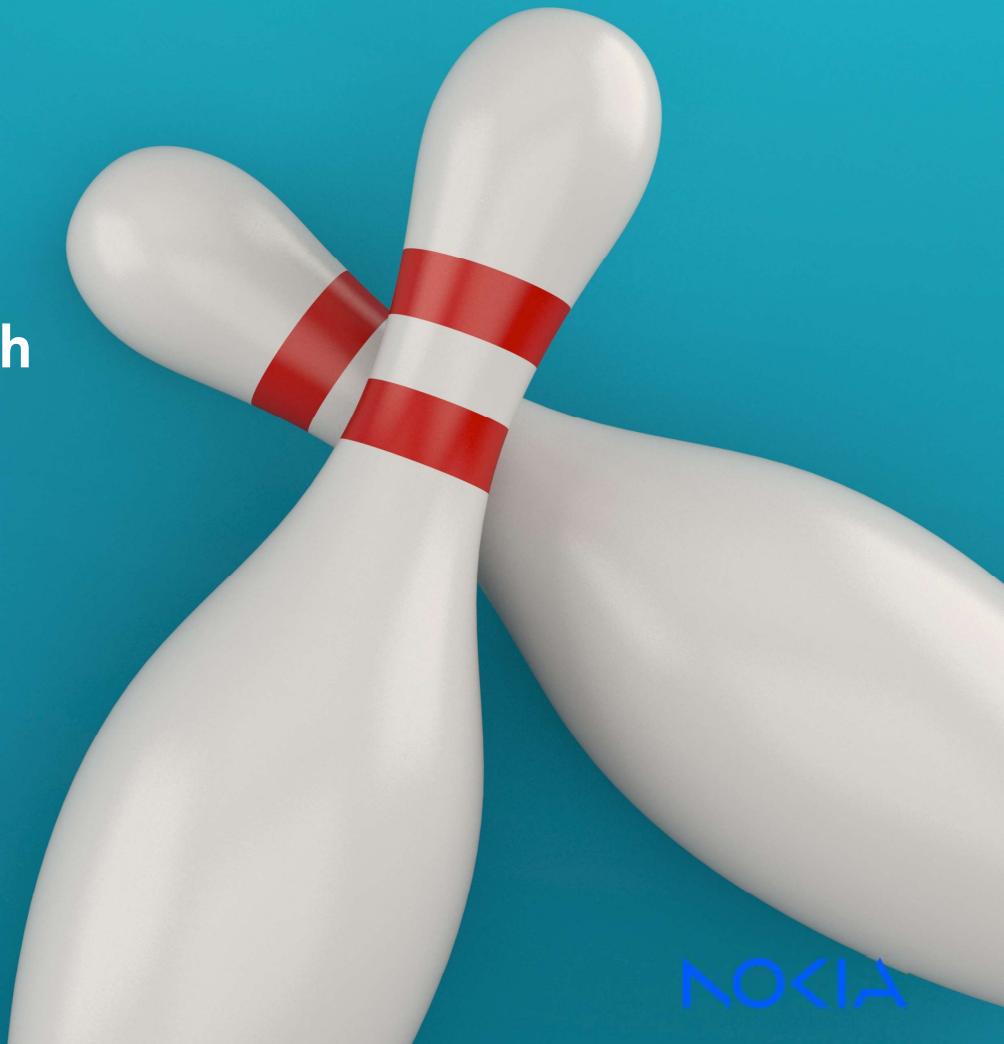
Liczenie punktów podczas gry w kręgle

- 10 rund, maks. po 2 rzuty w każdej
- 10 pinów
- Wynik za rundę: **ilość przewróconych pinów + bonusy**

Wymagania:

Wynik gracza

+ **void** roll(**int** pins)
+ **int** getScore()



NOKIA

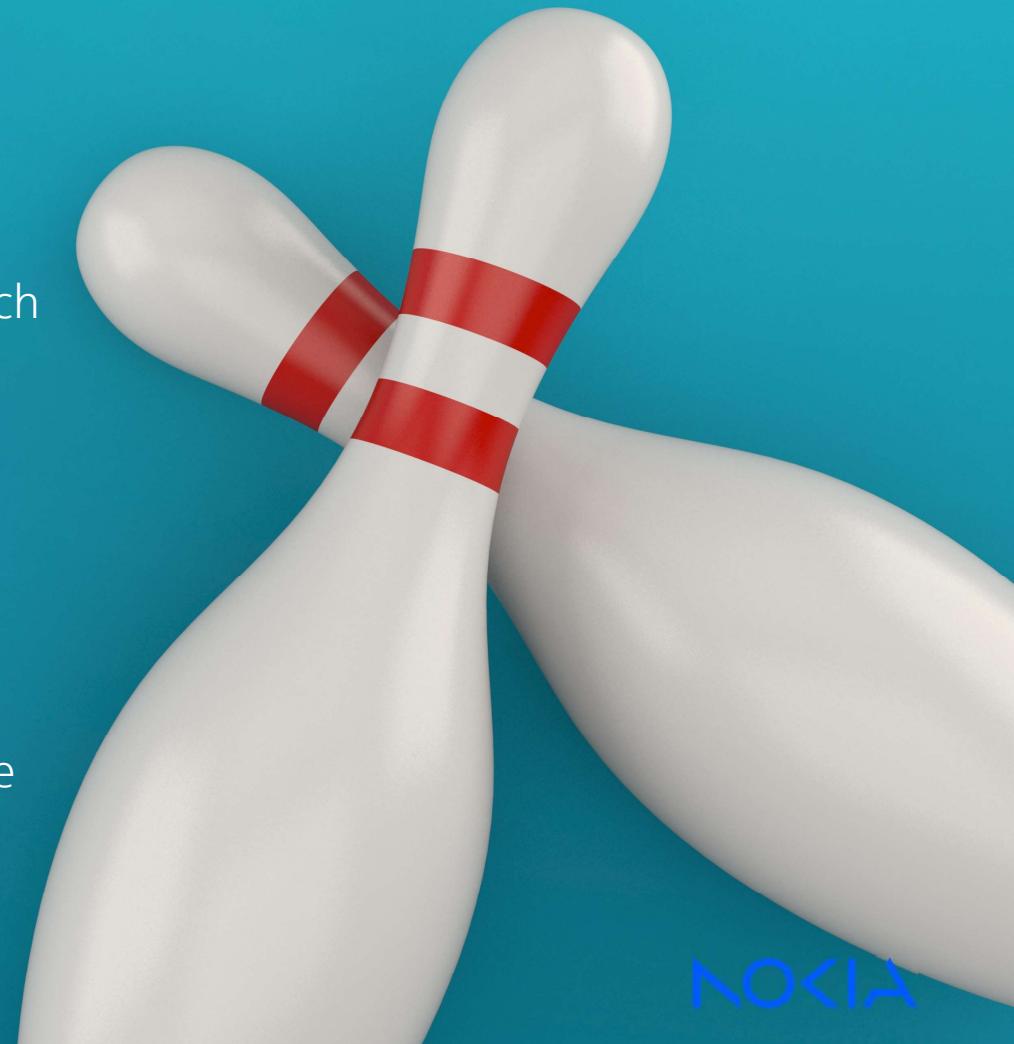
Zadanie na dziś

Liczenie punktów podczas gry w kręgle

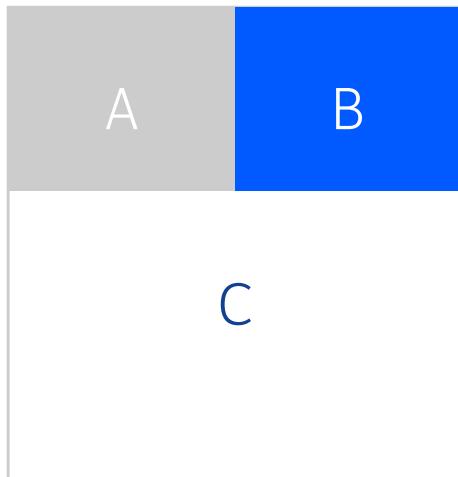
Wynik za rundę: ilość przewróconych pinów +

bonusy za spare i strike:

- **Spare:** przewrócenie 10 pinów podczas dwóch rzutów w jednej rundzie
Bonus: Ilość przewróconych pinów podczas następnego rzutu (kolejna runda)
- **Strike:** przewrócenie 10 pinów podczas pierwszego rzutu w rundzie
Bonus: Punkty zebrane podczas następnych dwóch rzutów (kolejna runda lub dwie kolejne rundy)



Notacja wyników



A – wynik za pierwszy rzut

B – wynik za drugi rzut (opcjonalny)

C – wynik gry w danej rundzie

Prosta Gra

Symulacja

Gracz strąca 1 kręgiel w pierwszym rzucie.



NOKIA

Prosta Gra

Symulacja

Gracz strąca 4 kręgle w następnym rzucie.

Suma za rundę wynosi 5.



Prosta Gra

Symulacja

Gracz strąca 4 kręgle w następnym rzucie.



Prosta Gra

Symulacja

Gracz strąca 5 kręgli w następnym rzucie.



Prosta Gra

Symulacja

Gracz strąca 6 kręgli w następnym rzucie.



Prosta Gra

Symulacja

Gracz strąca 4 kręgle w następnym rzucie.

Jest to **SPARE**, strącenie 10 kręgli w dwóch rzutach podczas jednej rundy.

BONUS: Do wyniku rundy doliczany jest wynik z **JEDNEGO** następnego rzutu.

1	4	4	5	6	/
5		14			



Prosta Gra

Symulacja

Gracz strąca 5 kręgli.

Doliczanie bonusu za **SPARE**

1	4	4	5	6	/	5	5
5		14		29			

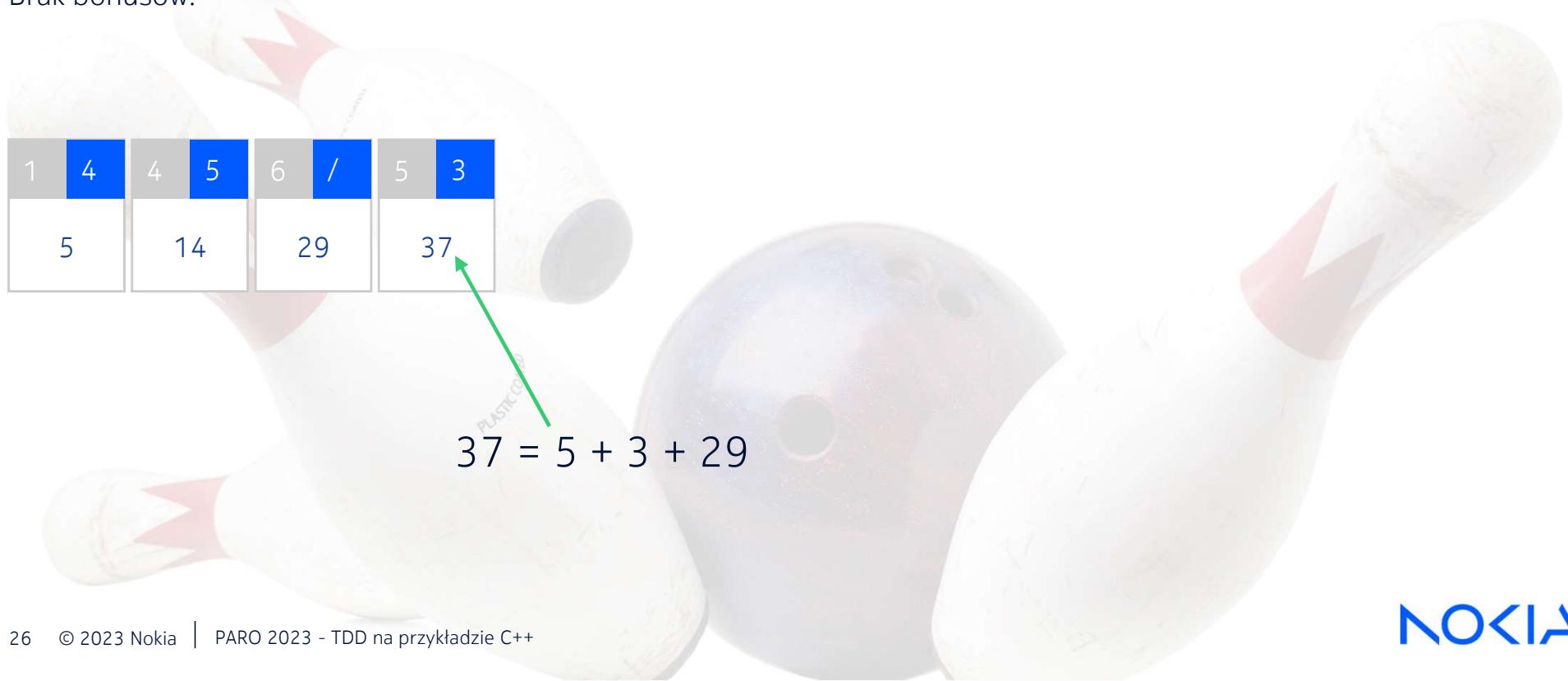
$$29 = 6 + 4 + (5) + 14$$

Prosta Gra

Symulacja

Gracz strąca 3 kręgle w kolejnym rzucie czwartej rundy.

Brak bonusów.



Prosta Gra

Symulacja

Gracz strąca 10 kręgli w jednym rzucie.

Jest to **STRIKE**, strącenie 10 kręgli w pierwszym rzucie rundy.

BONUS: Do wyniku rundy doliczany jest wynik z **DWÓCH** następnych rzutów.

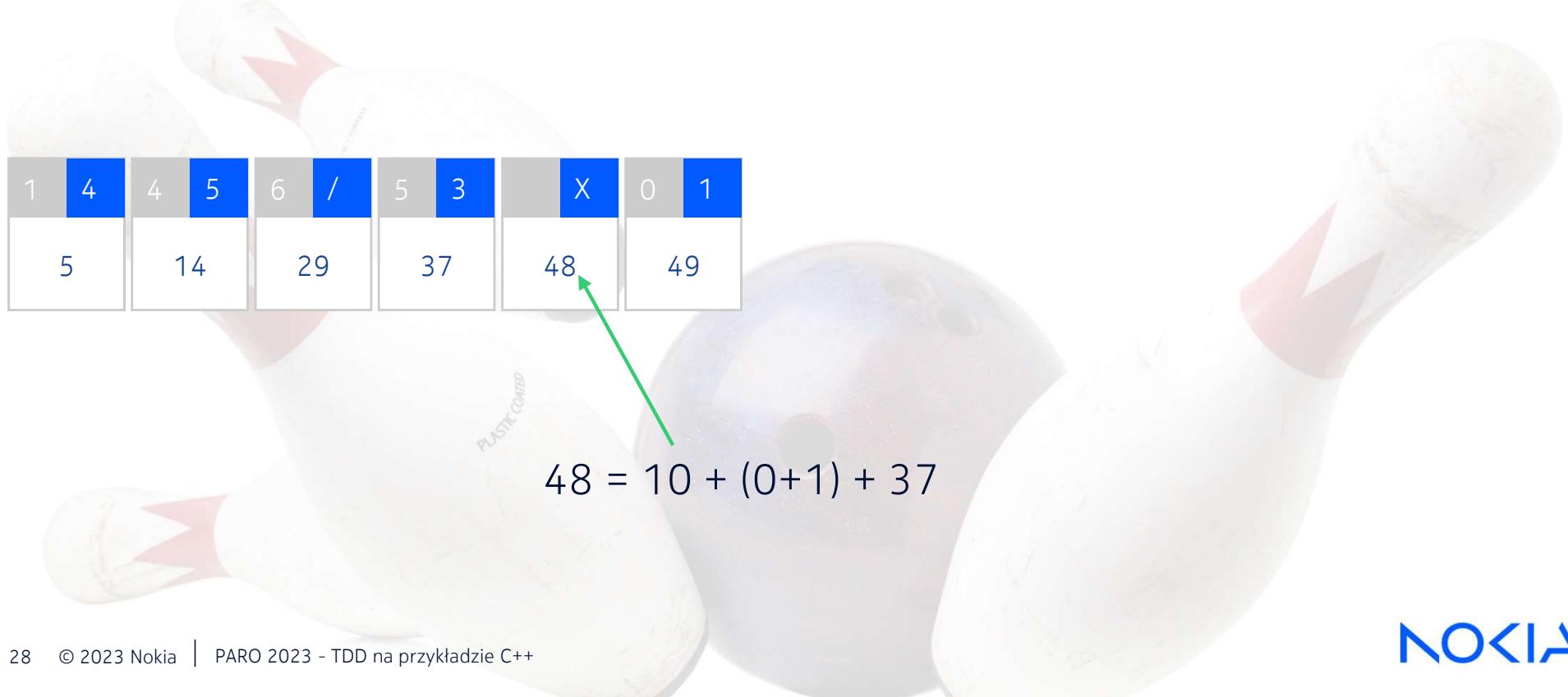
1	4	4	5	6	/	5	3	X
5	14	29	37					



Prosta Gra

Symulacja

Gracz strąca 1 kręgiel, w drugim rzucie kolejnej rundy.



Prosta Gra

Symulacja

Ciąg dalszy gry.

1	4	4	5	6	/	5	3	X	0	1	0	/	7	2	X
5	14	29	37	48		49			66		75				



Prosta Gra

Symulacja

Ostatnia, finałowa runda.

1	4	4	5	6	/	5	3	X	0	1	0	/	7	2	X	2	/	6
5		14		29		37		48		49		66		75		95		111

Runda różni się od typowej rundy, gdyż w przypadku gdy gracz rzuci w ostatniej rundzie:

SPARE (w dwóch rzutach), dostaje jeden dodatkowy rzut.

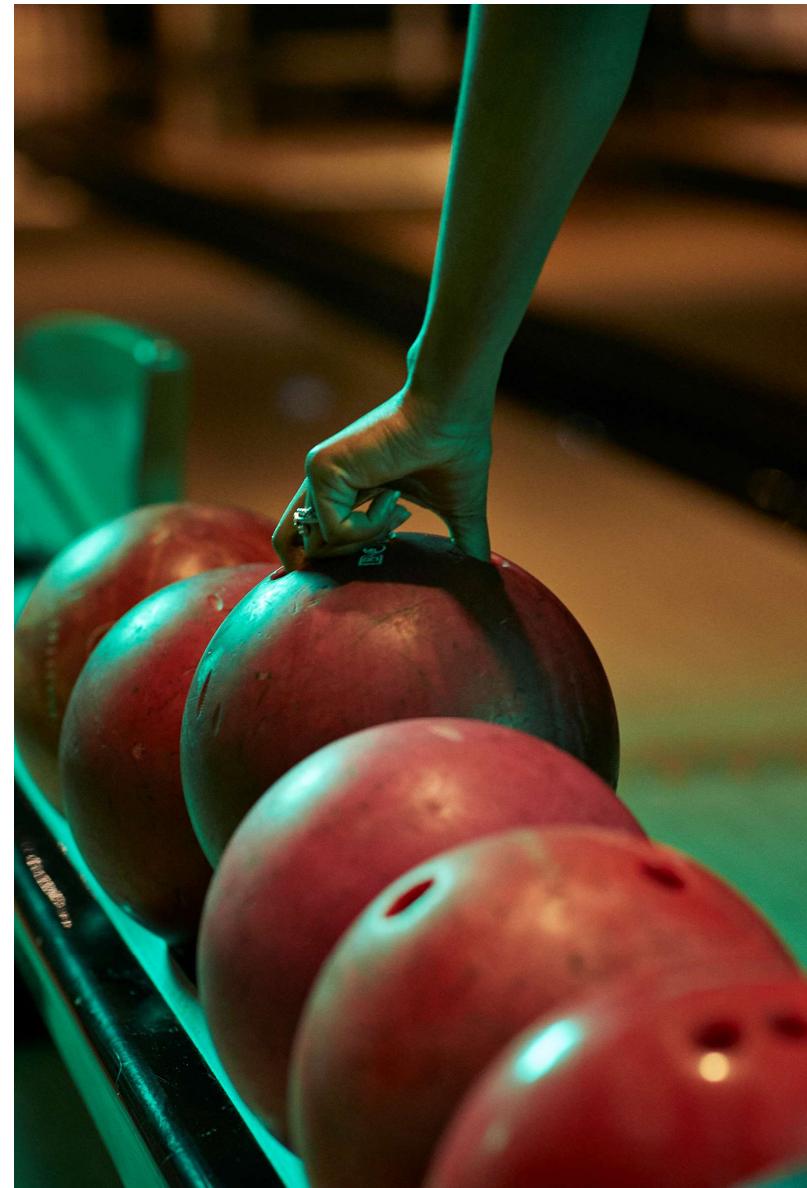
STRIKE (w jednym rzucie), dostaje dwa dodatkowe rzuty.

Wyniki kilku gier

Sztandarowe przykłady

Rzucając cały czas **1**, wynik gry to **20**.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2		4		6		8		10		12		14		16		18		20	

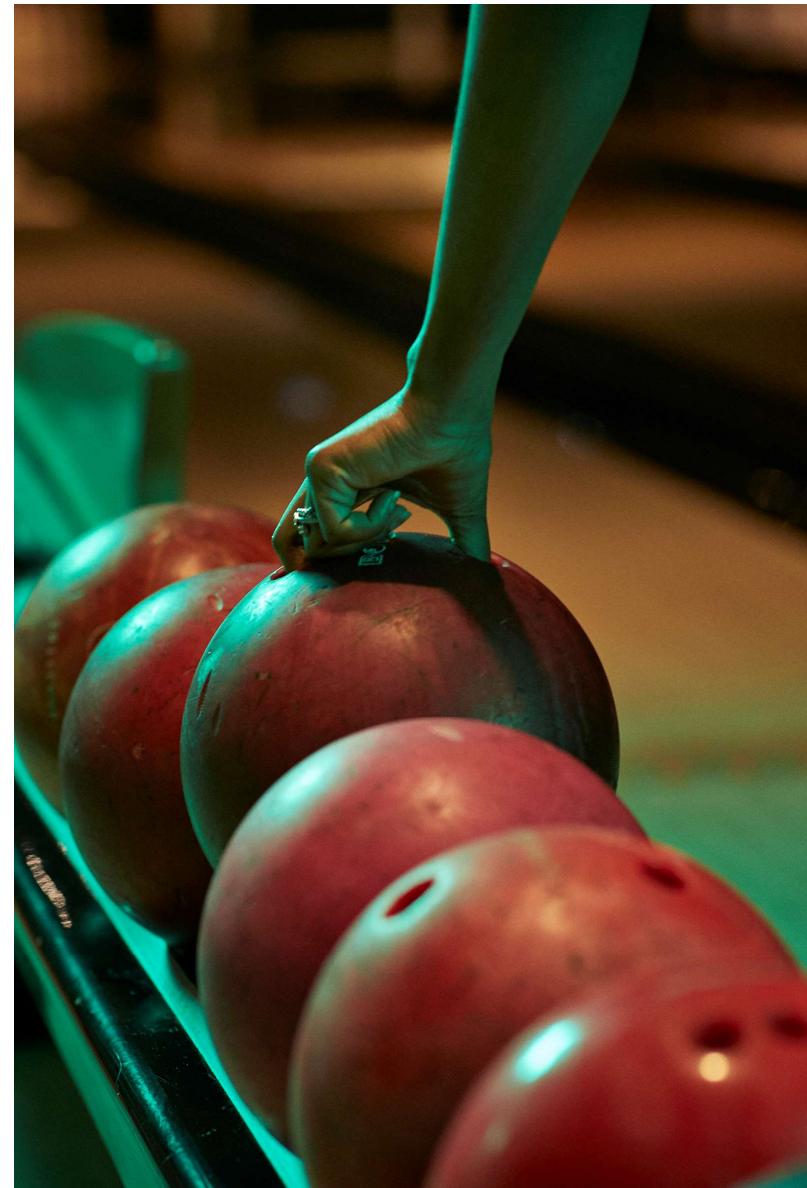


Wyniki kilku gier

Sztandarowe przykłady

Rzucając cały czas **9 i 1 (spare)**, wynik gry to **190**.

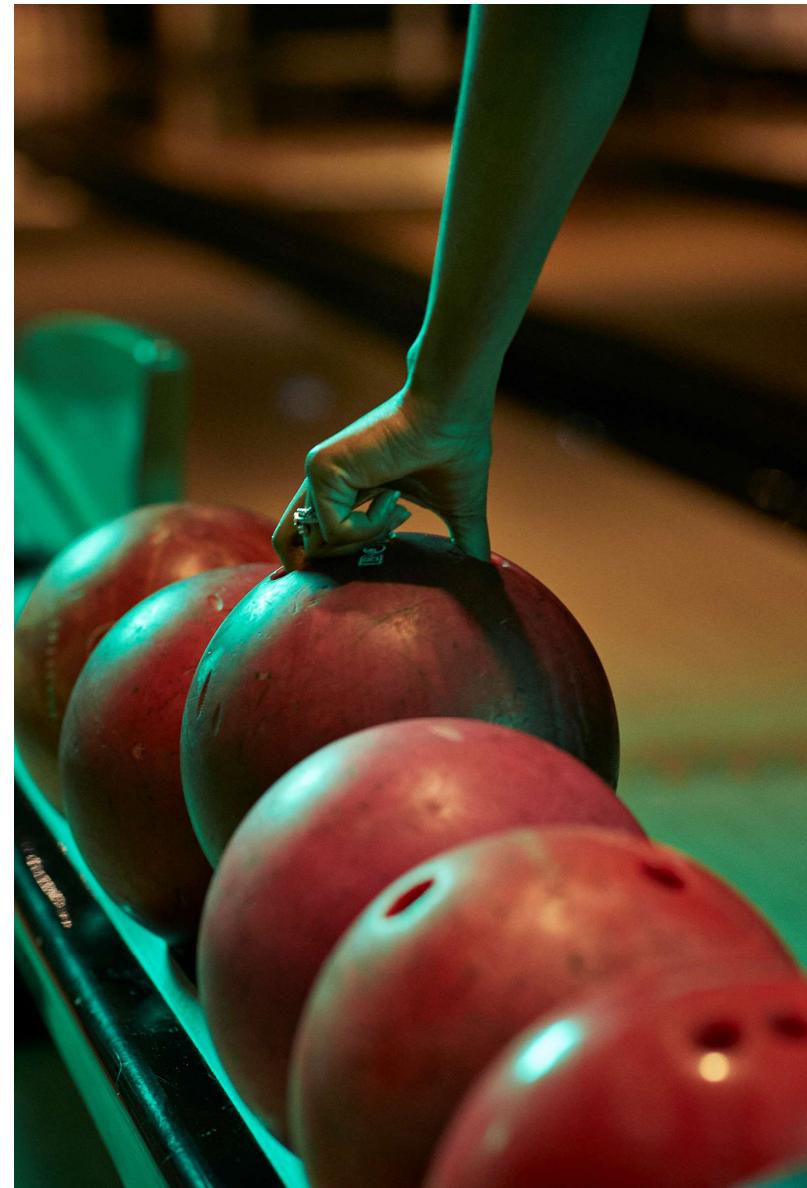
/	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9	/	9
19	38	57	76	95	114	133	152	171	190										



Wyniki kilku gier

Sztandarowe przykłady

Rzucając cały czas **10 (strike)**, wynik to....

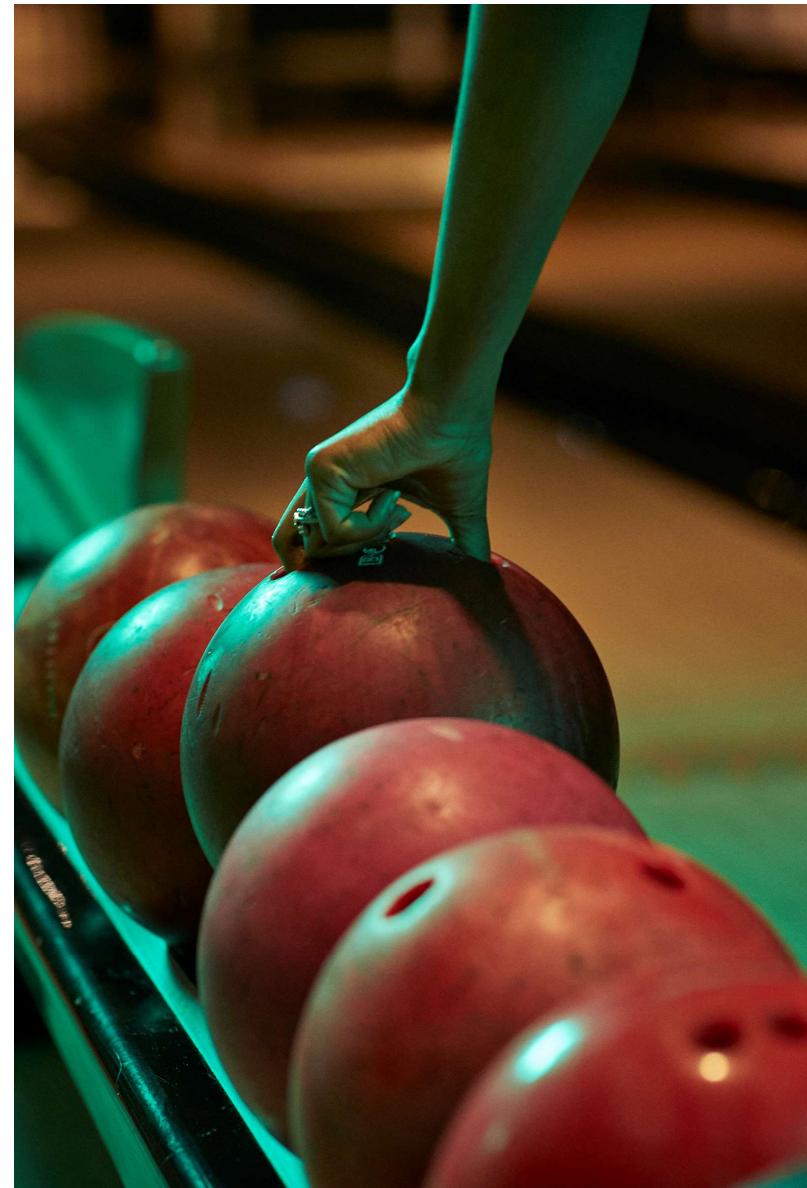


Wyniki kilku gier

Sztandarowe przykłady

Rzucając cały czas **10 (strike)**, wynik to **300**:
tzw. **Perfect Game**

X	X	X	X	X	X	X	X	X	X	X	X	X
30	60	90	120	150	180	210	240	270	300			



Praktyka



Ściągawka

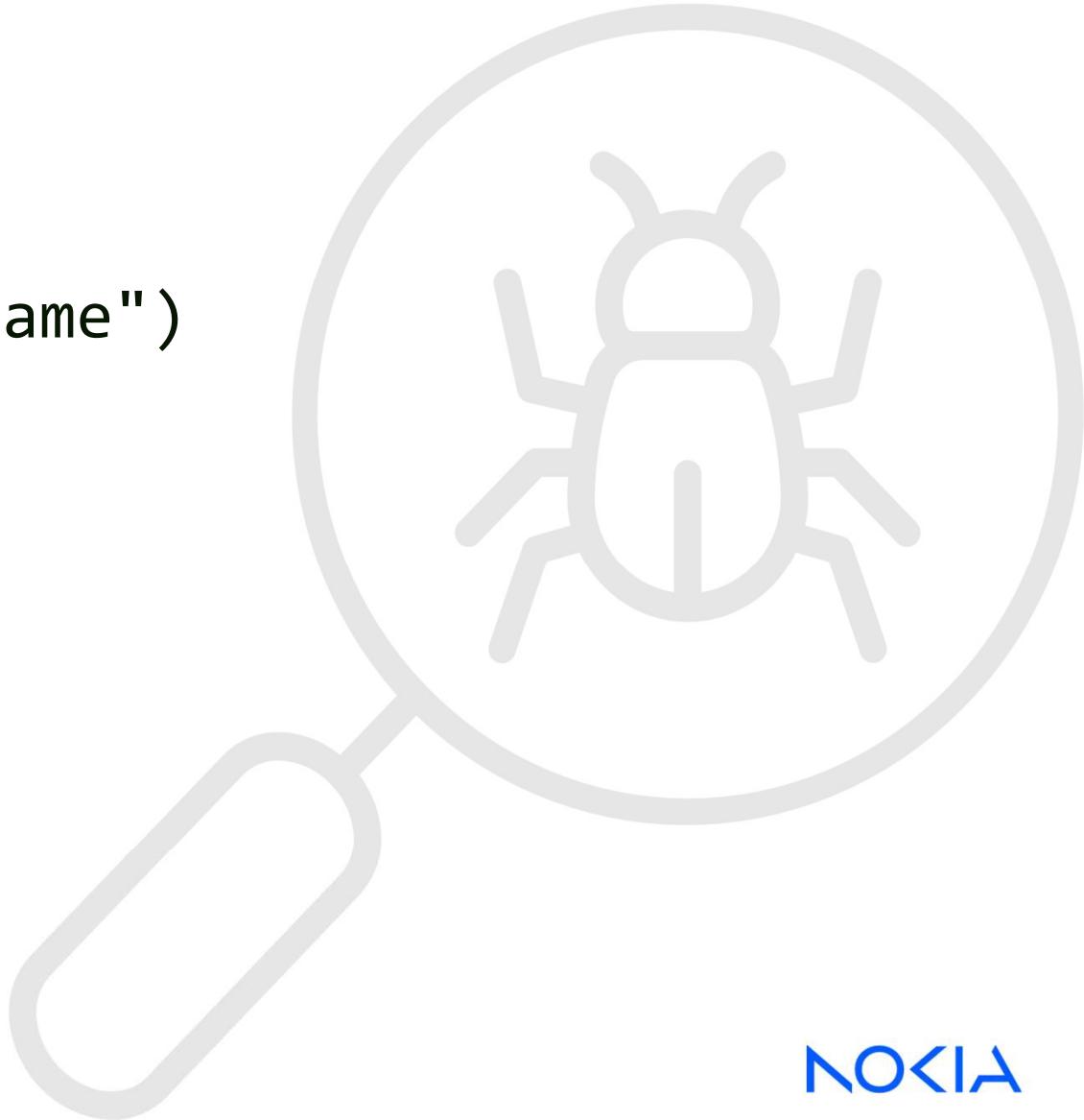
- Biblioteka do testowania **Catch** – źródła i wiki
<https://github.com/catchorg/Catch2>
- Repozytorium z programem startowym
`git clone`
<https://github.com/heireann/paro2023.git>
- Do weryfikacji wyników można użyć kalkulatora online, przykładowo:
<https://www.bowlinggenius.com/>



Implementacja testu

Dodawanie scenariusza testowego

```
TEST_CASE("Test_case_name")  
{  
}
```

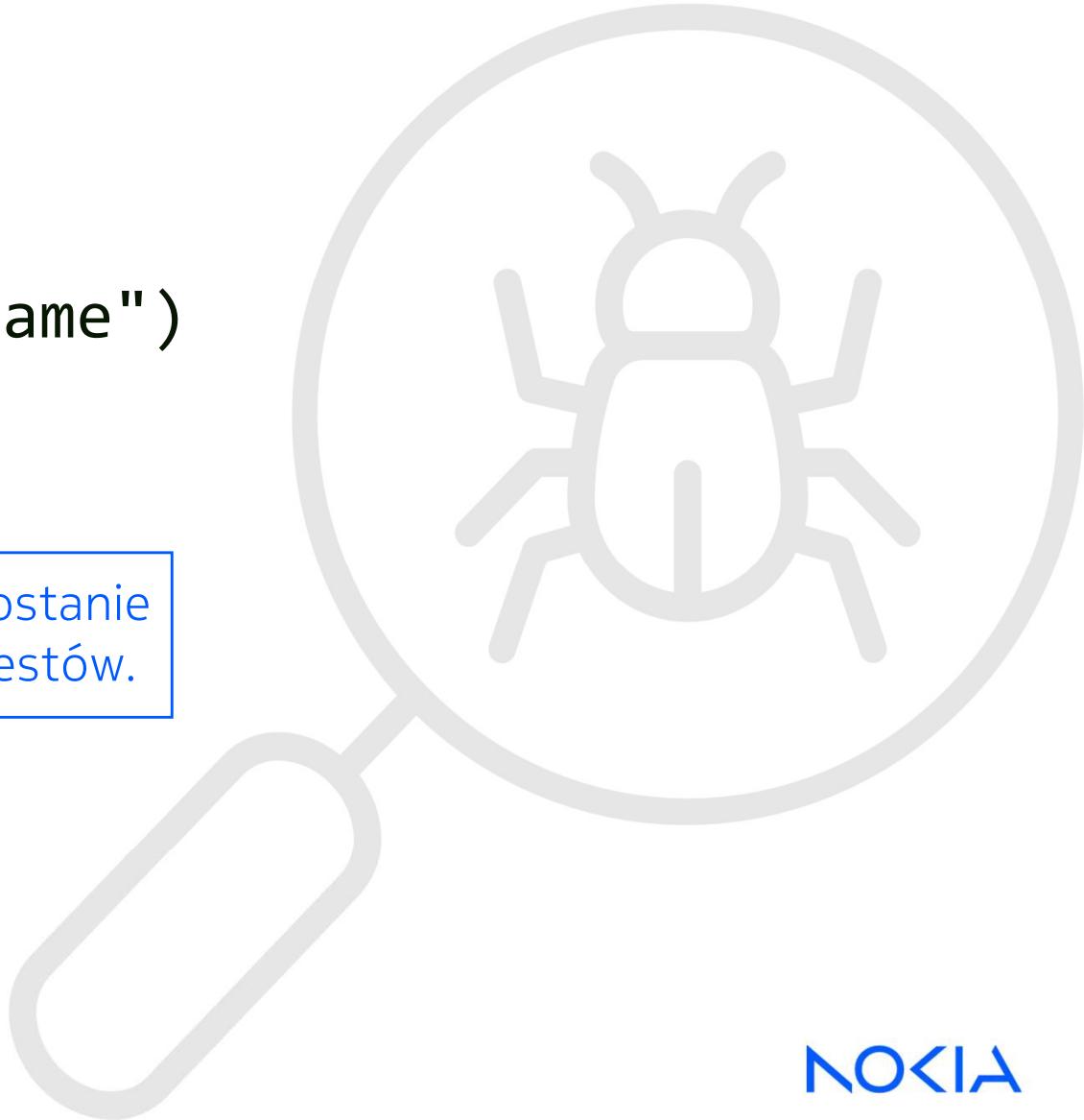


Implementacja testu

Dodawanie scenariusza testowego

```
TEST_CASE("Test_case_name")  
{  
}
```

Definiuje funkcje która zostanie wykonana jako jeden z testów.

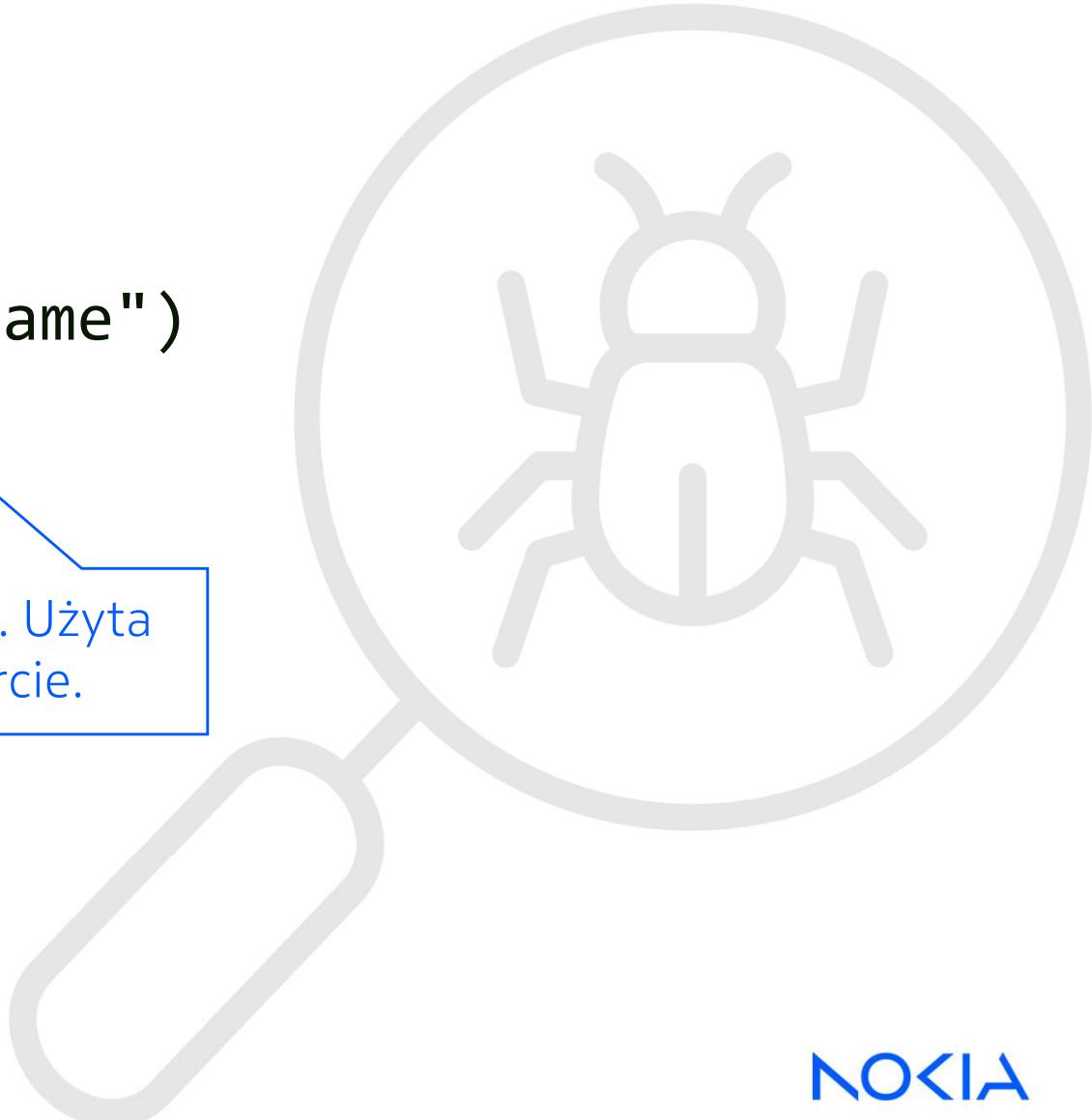


Implementacja testu

Dodawanie scenariusza testowego

```
TEST_CASE("Test_case_name")  
{  
}
```

Definiuje nazwę testu. Użyta
będzie ona w raporcie.



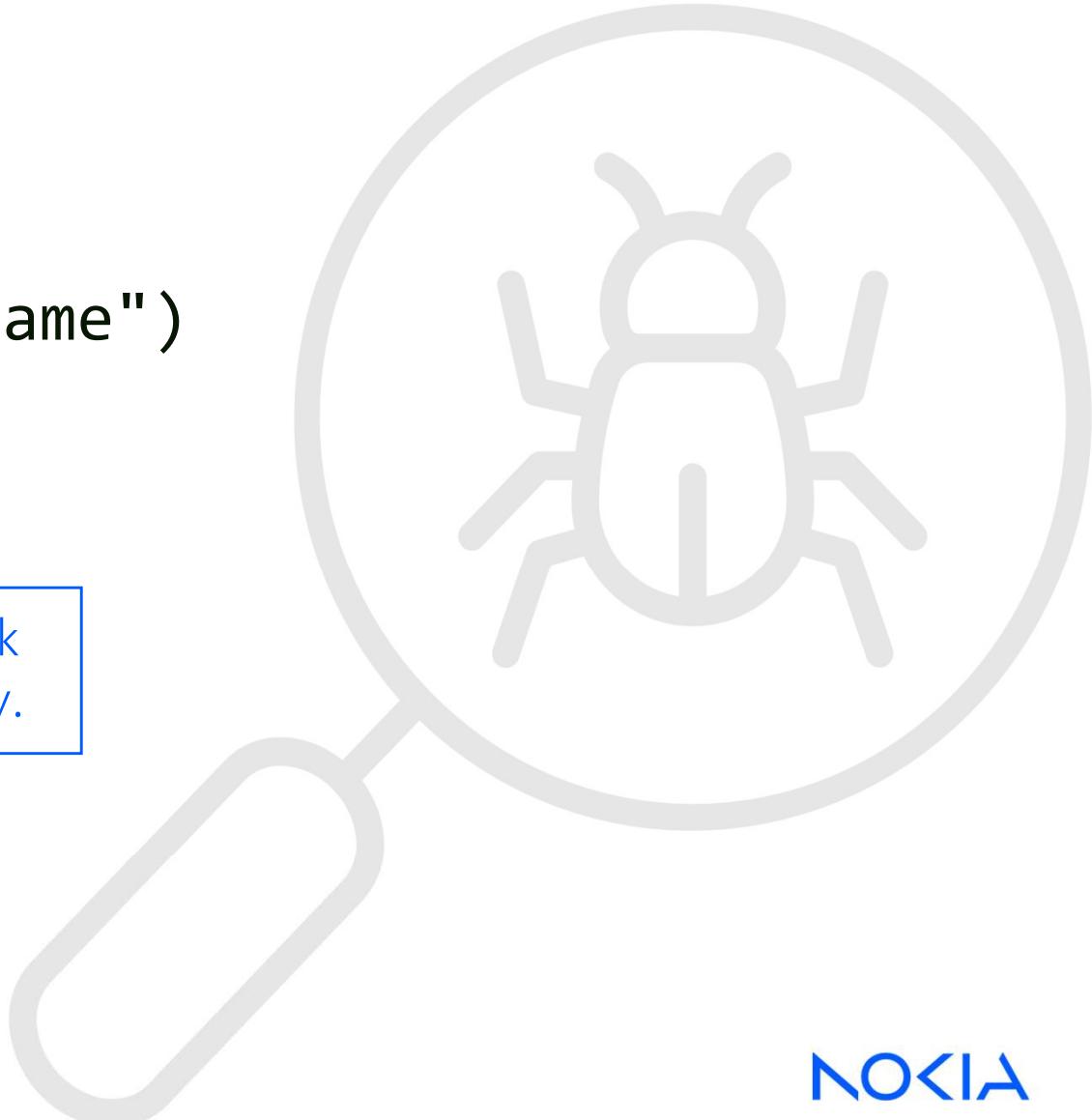
Implementacja testu

Dodawanie scenariusza testowego

```
TEST_CASE("Test_case_name")
```

```
{  
}
```

Definiuje ciało testu, co i jak
uruchamiamy i sprawdzamy.



Implementacja testu

Dodawanie scenariusza testowego

```
TEST_CASE("Test_case_name")
{
}
TEST_CASE("Test_case_name1")
{
}
...
TEST_CASE("Test_case_nameN")
{}
```

Liczba testów jest teoretycznie nieograniczona.

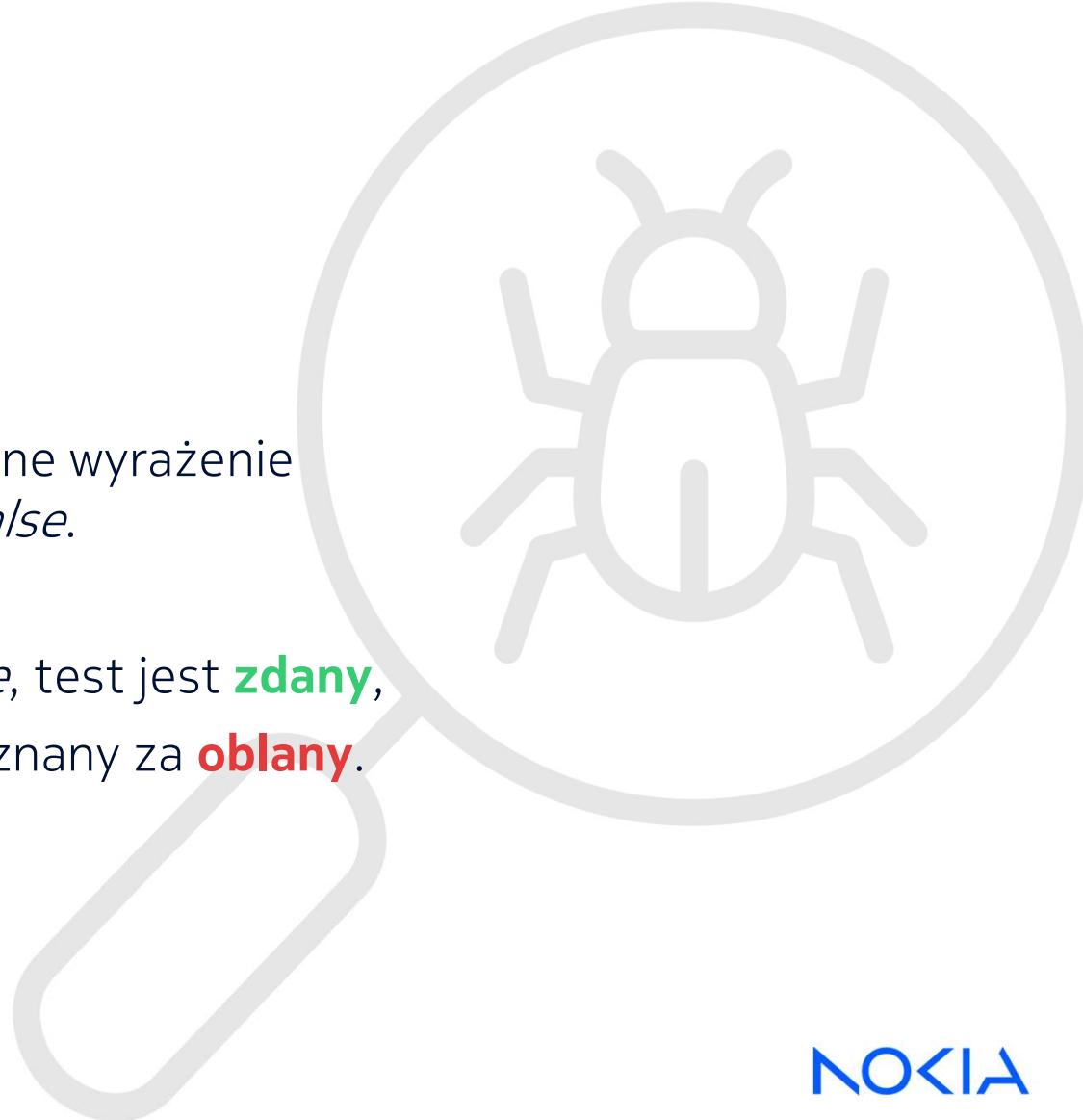
Implementacja testu

Sprawdzanie wyniku

```
REQUIRE(/* warunek_testu */)
```

Warunkiem testu powinno być logiczne wyrażenie zwracające wartości bool: *true* lub *false*.

Kiedy wyrażenie zwraca wartość *true*, test jest **zdany**, jeśli zwraca wartość *false* test jest uznany za **oblany**.



Implementacja testu

Trywialny przykład

Implementacja:

```
class SimpleCalculator {  
public:  
    int add(int a, int b) {  
        return a + b;  
    }  
};  
  
// prosta klasa z metodą "add"  
// zwracająca sumę dwóch liczb całkowitych
```

Testy:

```
TEST_CASE("SimpleCalculator_add_two_numbers") {  
    SimpleCalculator testObj;  
  
    REQUIRE(testObj.add(2, 2) == 4);  
    REQUIRE(testObj.add(2, 4) == 6);  
}  
  
TEST_CASE("SimpleCalculator_addition_is_commutative") {  
    SimpleCalculator testObj;  
  
    REQUIRE(testObj.add(2, 4) == 6);  
    REQUIRE(testObj.add(4, 2) == 6);  
}  
  
TEST_CASE("SimpleCalculator_addition_is_associative") {  
    SimpleCalculator testObj;  
  
    REQUIRE(testObj.add(testObj.add(2, 4), 8) == 14);  
    REQUIRE(testObj.add(2, testObj.add(4, 8)) == 14);  
}
```

Uruchomienie testu

Test niezaliczony

```
-----  
SimpleCalculator_addition_is_commutative  
-----
```

```
/home/user/paro2023/src/SCT.cpp:17  
.....
```

```
/home/user/paro2023/src/SCT.cpp:20: FAILED:
```

```
    REQUIRE( testObj.add(2, 4) == 7 )
```

```
with expansion:
```

```
6 == 7
```

```
=====
```

```
test cases: 3 | 2 passed | 1 failed
```

```
assertions: 5 | 4 passed | 1 failed
```



Uruchomienie testu

Test niezaliczony

```
-----  
SimpleCalculator_addition_is_commutative
```

```
/home/user/paro2023/src/SCT.cpp:17  
.....
```

```
/home/user/paro2023/src/SCT.cpp:20: FAILED:
```

```
    REQUIRE( testObj.add(2, 4) == 7 )
```

```
with expansion:
```

```
    6 == 7
```

```
=====
```

```
test cases: 3 | 2 passed | 1 failed
```

```
assertions: 5 | 4 passed | 1 failed
```

Nazwa oblanego testu

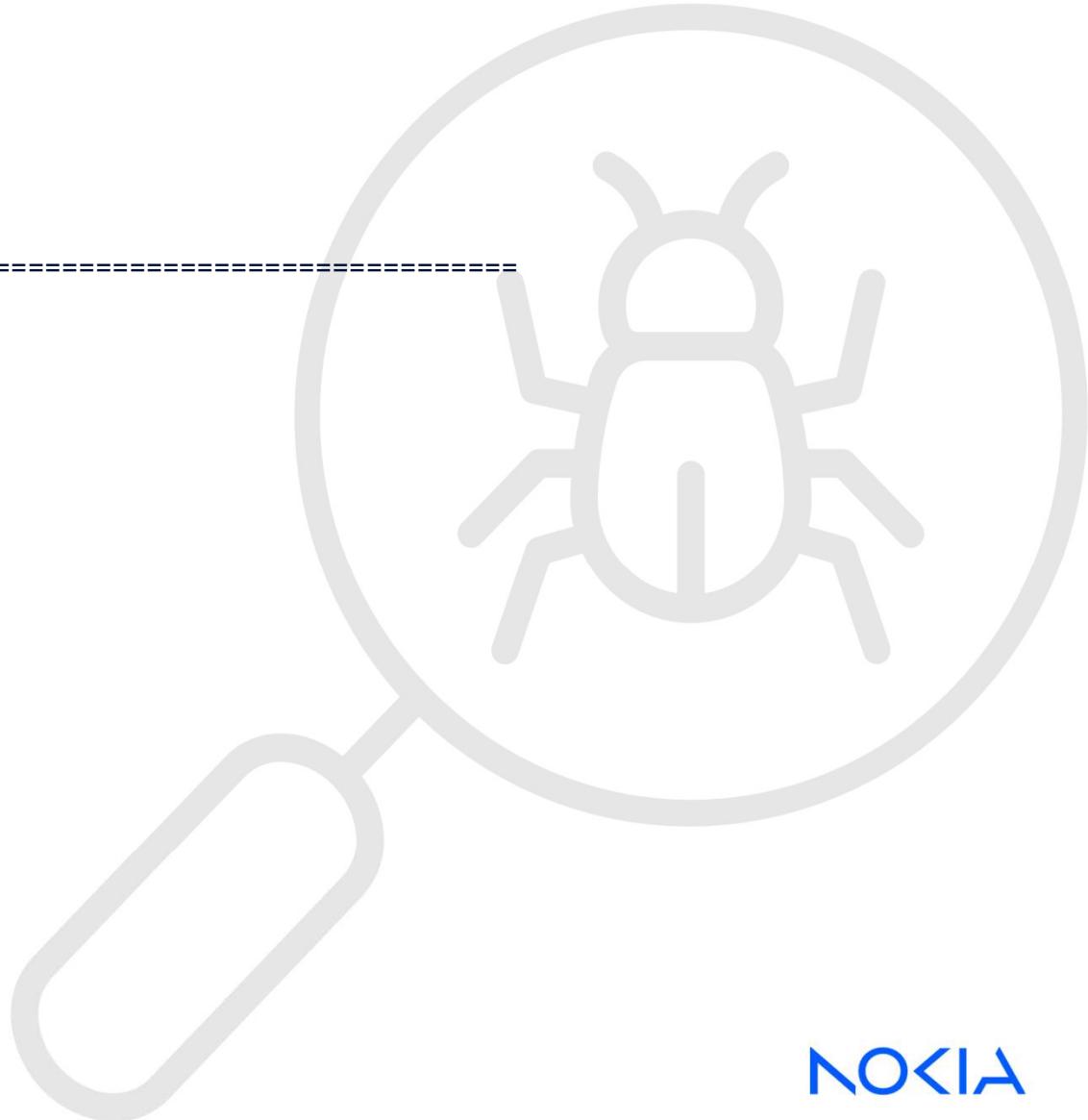
Linijka w której test się zaczyna
(Makro "TEST_CASE")

Niezaliczona asercja

Uruchomienie testu

Wszystkie testy zaliczone

All tests passed (6 assertions in 3 test cases)

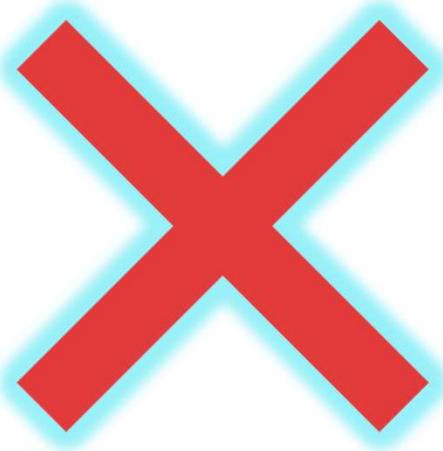


NOKIA

Pytania?



Podsumowanie



Po zajęciach...

Przykładowa implementacja dzisiejszego zadania

<https://github.com/heireann/bowling>



NOKIA

Dla zainteresowanych źródła

- Kent Beck – “**TDD. Sztuka tworzenia dobrego kodu.**”
- Robert C. Martin – “**Czysty Kod**”
- Martin Fowler et al. – “**Refaktoryzacja**”

Dziękujemy!



NOKIA