

# Praktyczne Aspekty Rozwoju Oprogramowania: **TDD na przykładzie C++**

Hanna Senhadri (hanna.senhadri@nokia.com)

Kamil Witecki (kamil.witecki@nokia.com)

# Kwestie organizacyjne

- Kurs Praktyczne Aspekty Rozwoju Oprogramowania został przygotowany przez pracowników firmy Nokia Solutions and Networks
- 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 w trakcie zajęć?



# Kwestie organizacyjne... a zdalna forma zajęć

- Sprawdzenie listy obecności – na podstawie Teams
- Na zajęciach przysługują dodatkowe punkty za aktywność (+, ++):
  - Wykonanie przykładu publicznie
  - Prezentacja zadania publicznie
  - Pytania i odpowiedzi
- Okno czatu – pomagamy i odpowiadamy na pytania!
- Osoby wypowiadające się: prosimy o oznaczenie się na czacie
- Propozycje dobrych praktyk z poprzednich zajęć?
  - Możemy współdzielić/pokazywać kod przez dedykowany link: <https://rextester.com/>  
<https://godbolt.org/>
  - ...



czym jest tdd?

# test driven development

## Zasady pracy w TDD



Test

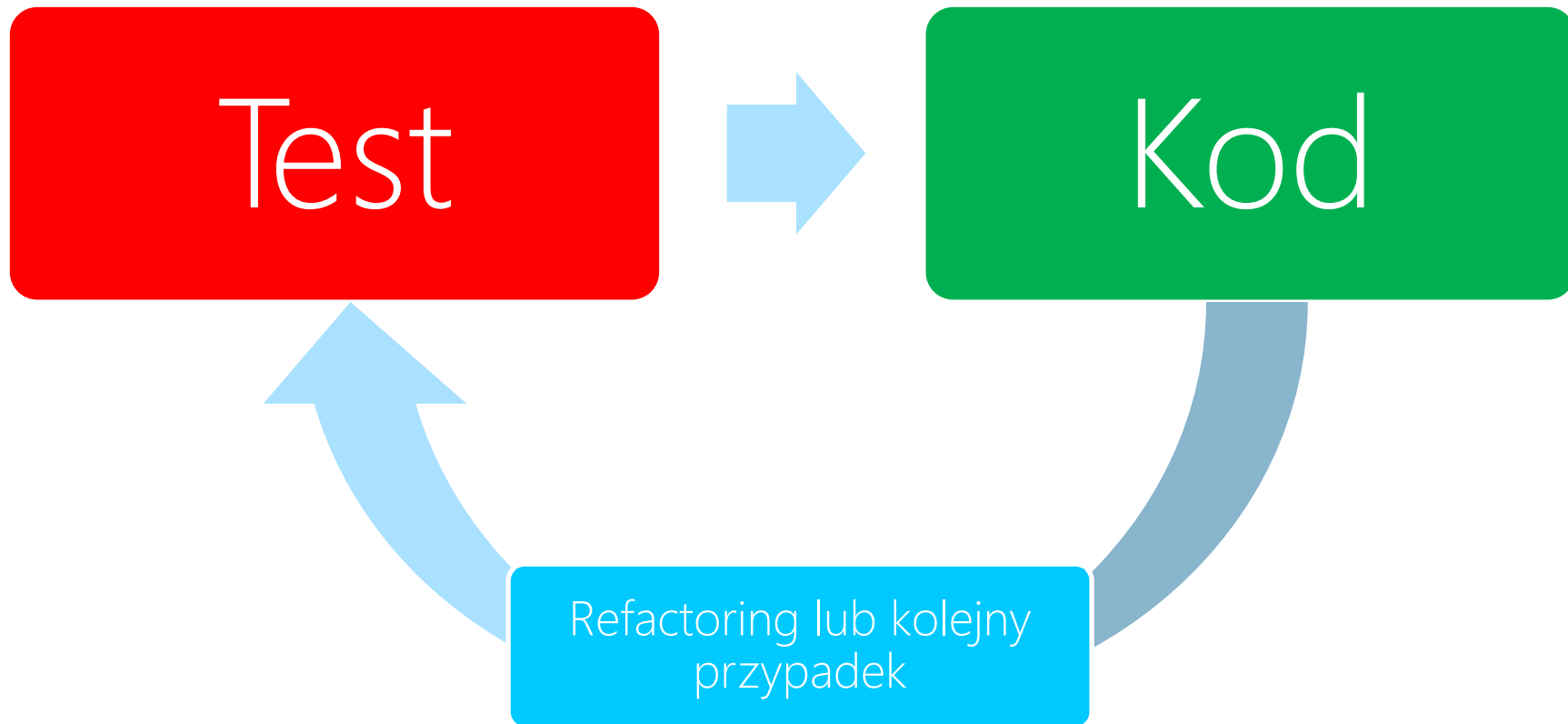
## Zasady pracy w TDD

Test



Kod

## Zasady pracy w TDD





# Zadanie na dziś

# Zadanie na dziś

Liczenie punktów podczas gry w kręgle

- 10 rund, po 2 rzuty
- 10 pinów
- Wynik za rundę: ilość przewróconych pinów + bonusy

Wymagania:

Wynik gracza
+ void roll(int pins)
+ int getScore()



# Zadanie na dziś

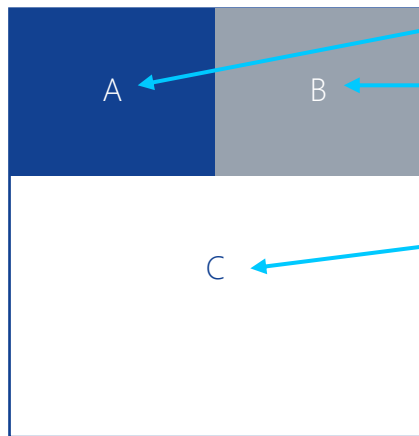
## Liczenie punktów podczas gry w kręgle

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

Gracz strąca 1 kręgiel w pierwszym rzucie.



## Prosta Gra

Gracz strąca 4 kręgle w następnym rzucie.

Suma za rundę wynosi 5.

1	4
5	

## Prosta Gra

Gracz strąca 4 kręgle w następnym rzucie.

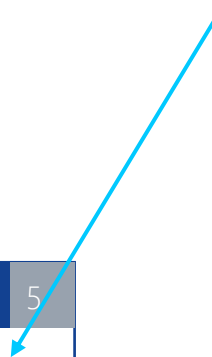
1	4
5	

## Prosta Gra

Gracz strąca 5 kręgów w następnym rzucie.

1	4
5	

4	5
14	

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




## Prosta Gra

Gracz strąca 6 kręgów w następnym rzucie.

1	4	4	5	6	
5	14				

## Prosta Gra

Gracz strąca 4 kręgle w następnym rzucie.

Jest to SPARE, strącenie 10 kręgów 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

Gracz strąca 5 kręgli.

Doliczanie bonusu za spare

1	4	4	5	6	/	5	
5	14	29					


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

## Prosta Gra

Gracz strąca 5 kręgów.

Następny SPARE

1	4	4	5	6	/	5	/
5	14	29					

## Prosta Gra

Gracz strąca 10 kręgli w jednym rzucie

Jest to STIKE, strącenie 10 kręgli w pierwszym rzucie rundy.

1	4	4	5	6	/	5	/	X
5	14	29	49					

$$49 = 5 + 5 + (10) + 29$$

## Prosta Gra

Gracz strąca 10 kręgów w jednym rzucie

Jest to STIKE, strącenie 10 kręgów w pierwszym rzucie rundy.

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

1	4	4	5	6	/	5	/		X
5	14	29	49						

# Prosta Gra

Gracz nie strąca ani jednego kręglu

1	4	4	5	6	/	5	/	X	0
5		14		29		49			

# Prosta Gra

Gracz strąca 1 kręgiel.

1	4	4	5	6	/	5	/	X	0	1
5	14	29	49	60		61				

$$60 = 10 + (0 + 1) + 49$$





# Prosta Gra

Ostatnia, finałowa runda.

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.

14	45	6/	5/	X	01	7/	6/	X	2/6
5	14	29	49	60	61	77	97	117	133

## Wyniki kilku gier

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

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

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



## Wyniki kilku gier

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

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

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



# Praktyka

# Ściągawka

- Biblioteka do testowania Catch – źródła i wiki  
<https://github.com/catchorg/Catch2>
- Repozytorium z programem startowym  
**git clone https://github.com/heireann/paro2022**
- 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
```

Test:

```
TEST_CASE("SimpleCalculator_add_test") {  
    SimpleCalculator testObj; // tworzenie  
    instancji testowanego obiektu  
  
    REQUIRE(testObj.add(2, 2) == 4);  
  
    REQUIRE(testObj.add(2, 4) == 6);  
  
    REQUIRE(testObj.add(4, 2) == 6);  
  
    REQUIRE(testObj.add(2, testObj.add(2, 2))  
    == 6);  
}
```



# Podsumowanie...



## Po zajęciach...

Przykładowa implementacja dzisiejszego zadania:

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





# Dziękujemy!

