Czysty kod, część I

Zaawansowane metody programowania

mgr inż. Krzysztof Rewak

20 marca 2019

Wydział Nauk Technicznych i Ekonomicznych Państwowa Wyższa Szkoła Zawodowa im. Witelona w Legnicy

Plan prezentacji

- 1. Czysty kod
- 2. Nomenklatura
- 3. Komentarze
- 4. Podsumowanie

Czysty kod

Czysty kod

Czym jest czysty kod?

Bjarne Stroustrup

Lubię, gdy mój kod jest elegancki i efektywny.

Logika kodu powinna być **prosta**, aby nie mogły się w niej kryć błędy; zależności minimalne dla uproszczenia utrzymania; obsługa błędów kompletna zgodnie ze zdefiniowaną strategią; a wydajnośc zbliżona do optymalnej, aby nikogo nie **kusiło** psucie kodu w celu wprowadzenie niepotrzebnych optymalizacji.

Czysty kod wykonuje dobrze jedną operację.

Dave Thomas i Andy Hunt

Kod jest jak budynek.

Budynek z rozbitymi oknami wygląda, jakby nikt się o niego nie **troszczył**. Dlatego inni ludzie również przestają się o niego troszczyć. W ten sposób coraz więcej okien jest rozbijanych. W końcu sami je rozbijają. Zaczynają bazgrać na fasadzie i pozwalają na powstanie gór śmieci.

Jedno rozbite okno rozpoczyna **proces upadku**.

Grady Booch

Czysty kod jest prosty i bezpośredni. Czysty kod czyta się jak dobrze napisaną prozę. czysty kod nigdy nie zaciemnia zamiarów projektanta; jest pełen trafnych abstrakcji i prostych ścieżek sterowania.

Dave Thomas

Czysty kod może być czytany i rozszerzany przez innego programistę niż jego autor. Posiada on testy jednostkowe i akceptacyjne. Zawiera znaczące nazwy. Oferuje jedną, a nie wiele ścieżek wykonania jednej operacji. Posiada minimalne zależności, które są jawnie zdefiniowane, jak również zapewnia jasne i minimalne API. Kod powinien być opisywany przy jednoczesnej zależności od języka - nie wszystkie potrzebne informacje mogą być wyrażane bezpośrednio w kodzie.

Michael Feathers

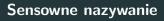
Mógłbym wymieniać wszystkie cechy, jakie zauważam w czystym kodzie, ale istnieje jedna, która prowadzi do pozostałych. Czysty kod zawsze wygląda, jakby był napisany przez kogoś, komu na nim zależy. Nie ma w nim nic oczywistego, co mógłbyś poprawić.

Autor kodu pomyślał o wszystkim i jeżeli próbujemy sobie wyobrazić usprawnienia, prowadzą one nas tam, skąd zaczęliśmy, co pozwala nam docenić kod, który ktoś dla nas napisał - kod, który napisał ktoś, kto naprawdę przyjmuje się swoimi zadaniami.

Ward Cunningham

Wiemy, że pracujemy na czystym kodzie, jeżeli każda procedura okazuje się taką, **jakiej się spodziewaliśmy**. Można nazywać go również pięknym kodem, jeżeli wygląda, jakby ten język został stworzony do rozwiązania danego problemu.

Nomenklatura



Wybór odpowiedniej nazwy to dobra inwestycja.

Sensowne nazywanie

Każda nazwa zmiennej, funkcji, argumentu, klasy czy interfejsu powinna być przemyślana oraz powinna przedstawiać intencję.



```
public array[Student] list
```

public float calculate()



return (float) list.sum(s => s.finalGrade) / list.size();



Sensowne nazywanie

Czasami nazwiemy zmienne inaczej niż a i b, ale i tak wiele nam to nie pomoże. Nazwy powinny idealnie odzwierciedlać swoje przeznaczenie i nie generować niepewności.



class Cart

public array[Product] CartObject public array[string] CartData public array[array[string] CartInfo



Dezorientacja może występować w kodzie na kilka sposobów.

```
•••
```

class Cart

```
public int products = 0
```

```
public bool checkIfCartHasAlcoholicBeverages()
public bool checkIfCartHasntAlcoholicBeverages()
```

```
* (...) */
```

```
public bool returnTrue()
    return false
```

Nie koduj

Kod nie powinien też zawierać w sobie żadnych zbędnych informacji.

Przykładowo dawniej popularna notacja węgierska jest obecnie coraz rzadziej używana ze względu na swoją redundancję.



```
class JSONExtractor implements IFormatExtractor
    public bool bReady = false
    public int nLinesOfCode
```

public string szContent

public IFormatExtractor load(string szFilename) szContent = open(szFilename)

return this

Nie kombinuj

Słowa mają swoje synonimy, jednak powinniśmy używać ich z głową. Lepiej wybrać jedno słowo i się go trzymać wewnątrz projektu czy żonglować w co drugiej klasie różnymi nazwami?



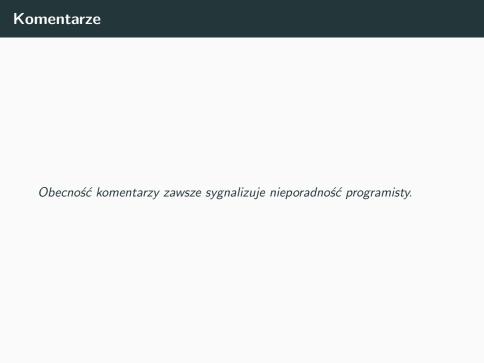
interface Spaceship

```
public int getCurrentSpeed()
public int fetchNumberOfWorkingFTLDrives()
public int retrieveSpaceFuelLevel()
```

Komentarze



Nie komentuj złego kodu - popraw go.



Komentarze

Komentarz może być przydatnym narzędziem, niestety najczęściej jest przeszkodą w tworzeniu dobrego kodu.

• • •

class Square

public static int calculate(int a, int b) // static, no need for other

• • •

if not equals

return equals

throw new NotEqualsException()

function bool equals(Model a, Model b)

bool equals = a.id == b.id && a.name == b.name



```
class Farcaster
public void transmit(Transmitable object)
```

```
// checkObjectSize(object)
// fetchTCInstructions()
```

```
/ initializePowerSource()
```

throw new FallException()



Czasami komentarz przetrzymuje dodatkowe informacje. Ale czy warto go używac?



if not equals

return equals

function bool equals(Model a, Model b)

bool equals = a.id == b.id && a.name == b.name

throw new NotEqualsException()



- * Created by PhpStorm
- * User: jkeats
- * Date: 2019/03/20
 - Time: 19:27

 ${\tt class\ PrisonerController\ extends\ Controller}$

Komentarze

Zatem kiedy korzystać z komentarzy?

Komentarze

- do informacji o prawach autorskich, jeżeli musimy takie informacje zawrzeć?
- do wyjaśnienia trudnych rzeczy?
- do oznaczenia TODO?

Podsumowanie



Kod prezentacji dostępny jest w repozytorium git pod adresem https://bitbucket.org/krewak/pwsz-zmp



Wszystkie informacje dot. kursu dostępne są pod adresem http://pwsz.rewak.pl/kursy/10

