Code Review & Clean Code

Hello!

Łukasz Duda

plan szkolenia

- Code Review
- Clean Code
- Ćwiczenia
- Ustalenia

peer review

- Henry Oldenburg (XVII w.)
- druga para oczu może wiele zauważyć
- rewolucyjne twierdzenie musi zostać zweryfikowane
- warto wiedzieć co robią inni naukowcy

Code Review

historia

- Michael Fagan i IBM
- Fagan Inspection Process
- XP i Continunous Inspection
- Agile Manifesto i jakość (wójek Bob)

Code Review

po co?

- komunikacja!
- wczesne eliminowanie błędów
- redukcja liczby błędów zgłaszanych przez klientów (Fagan 30% - 80%)
- identyfikowanie problemów w procesie

Code Review

lekkie przeglądanie

- SmartBear i Cisco Sytems
- 50 programistów
- 2500 przejrzeń
- 3.2 miliona wierszy kodu
- 10 miesięcy

Code Review

lekkie przeglądanie

- przejrzyj własny kod (pull request)
- do 200-400 wierszy kodu
- do 300-500 wierszy na godzinę
- do 60-90 minut
- nie ma za małych zmian!

Code Review

lekkie przeglądanie

- 90 minut jest wartością uznaną
- 6 razy wydajniej
- brak danych do porównania liczby błędów
- wspomaga częste zmiany

Code Review

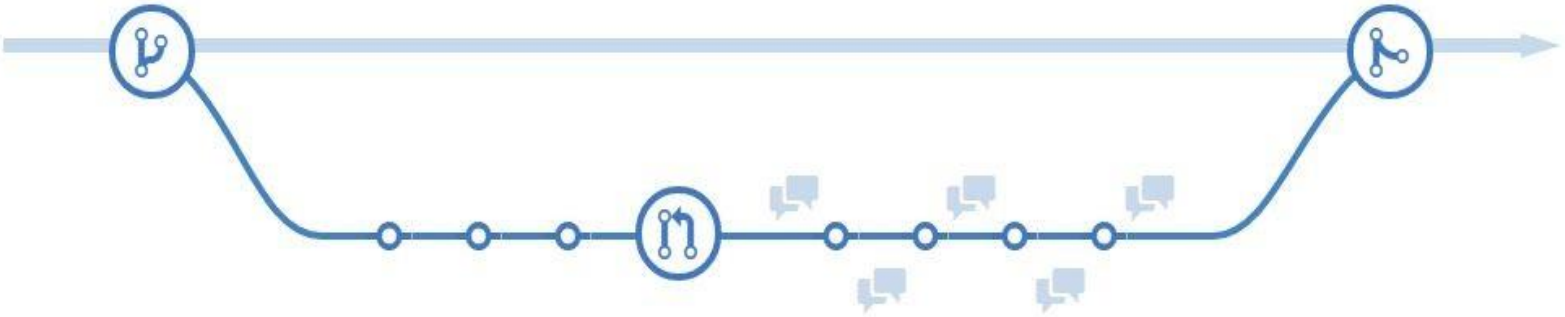
badanie naukowe

- wpływ lekkiego przeglądania na jakość
- metryka: ile błędów przedarło się na produkcję
- przeglądanie wszystkiego to za mało
- dyskusja jest ważna
- wiedza ekspercka

Code Review

GitHub

- komentarze do istniejących commitów
  - całe commity
  - pojedyncze linie
- dyskusje nad pull requestami



Code Review

tak rób

- przygotuj się
- pisz i przeglądaj testy
- używaj narzędzi
- szukaj defektów
- pomyśl o checkliście
- +1
- nie bój się

Code Review

tak NIE rób

Nie bądź ignorantem

Code Review

tak NIE rób

Nie wywyższaj się

Code Review

tak NIE rób

Nie formułuj bezsensownych żądań

Code Review

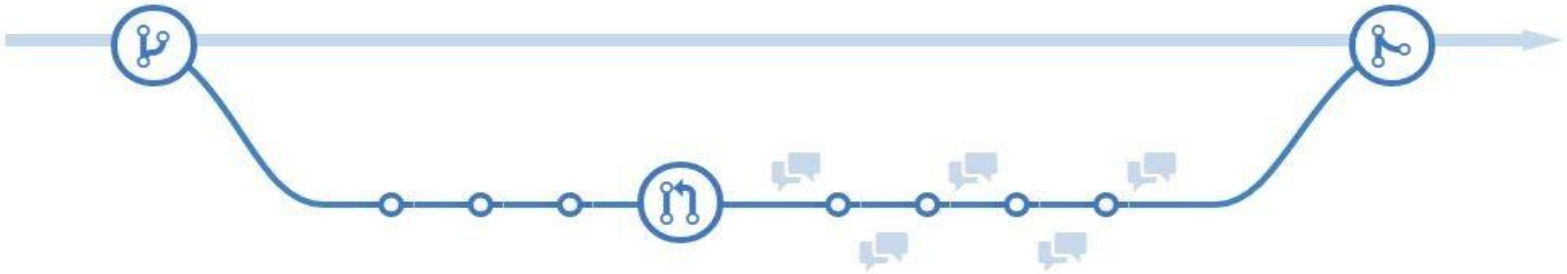
tak NIE rób

Szanuj partnerów

Code Review

GitHub

- dyskusje nad pull requestami



Code Review

problemy

- 20%-30% dłużej
- nadmiar obowiązków
- planowanie

- opór przed nowym i nieznanym
- opór ze strony menadżerów
- opór ze strony programistów

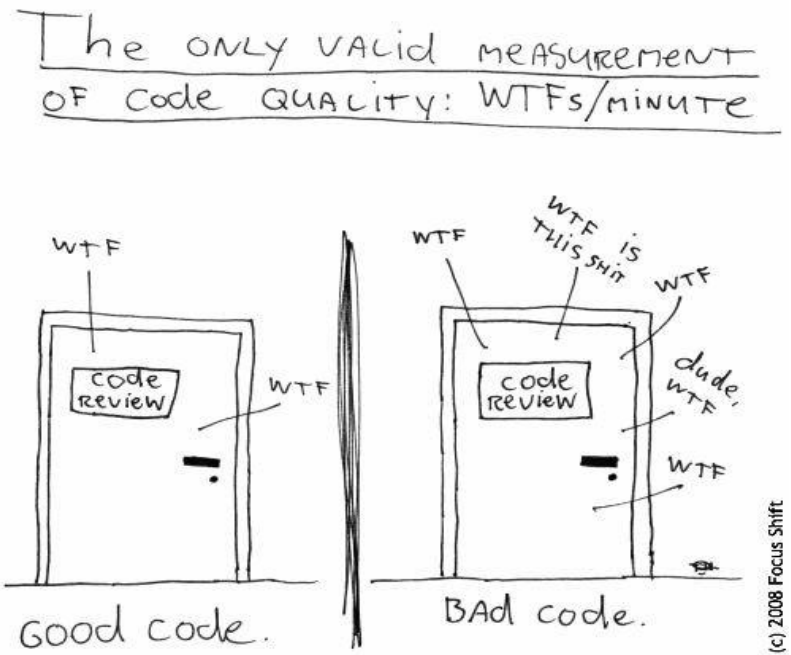
Code Review

korzyści

- komunikacja!
- oszczędności
- dotrzymywanie terminów
- efektywniejsze testowanie
- wiedza i standardy
- dokumentacja
- dojrzałość

Clean Code

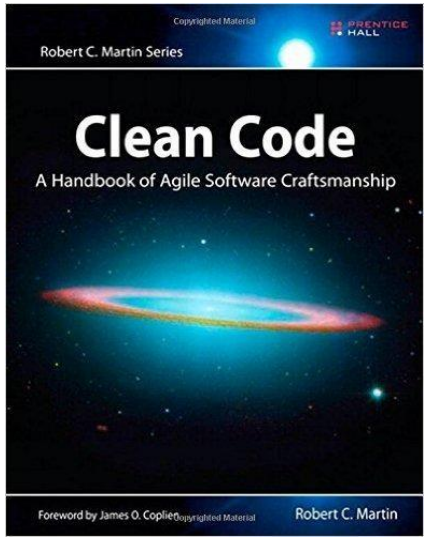
jakość?



Clean Code

Uncle Bob

- aka Robert Cecil Martin
- Agile Manifesto



Clean Code

nazewnictwo

- niech nazwę da się wymówić

Bad

```
var yyyymmdstr = moment().format('YYYY/MM/DD’);
```

Good

```
var currentDate = moment().format('YYYY/MM/DD');
```

Clean Code

nazewnictwo

- niech nazwy się różnią

```
function divdCalc(divdPrt1, divdPrt2) {
    if (divdPrt2 == 0) {
        throw new Exception("Error");
    }
    return divdPrt1 / divdPrt2;
}
```

```
function divideNumbers(dividend, divisor) {
    if (divisor == 0) {
        throw new InvalidArgumentException("Division by zero!");
    }
    return dividend / divisor;
}
```

Clean Code

nazewnictwo

- nadmiar informacji

Bad

```
var Car = {
  carMake: 'Honda',
  carModel: 'Accord',
  carColor: 'Blue'
};

function paintCar(car) {
  car.carColor = 'Red';
}
```

Good

```
var Car = {
  make: 'Honda',
  model: 'Accord',
  color: 'Blue'
};

function paintCar(car) {
  car.color = 'Red';
}
```

Clean Code

nazewnictwo - niech nazwa mówi o Twoich zamiarach

```
function calculate(data) {
    var r = 0;
    $.each(data, function(index, data1) {
        r = r + data1;
    });
    return r;
}
```

```
function sumValuesOfArrayElements(arrayToSum) {
    var sumOfValues = 0;
    $.each(arrayToSum, function(index, component) {
        sumOfValues = sumOfValues + component;
    });
    return sumOfValues;
}
```

Clean Code

nazewnictwo

- niech nazwę da się wyszukać

```
for (i=0; i<5; i++) {
    divideNumbers(i, 5);
}
```

```
var numberOfDivisions = 5;
for (i=0; i<numberOfDivisions; i++) {
    divideNumbers(i, numberOfDivisions);
}
```

Clean Code

nazewnictwo

- niech nazwa opisuje wartości stałe

Bad

```
setTimeout(blastOff, 86400000);
```

Good

```
var MILLISECONDS_IN_A_DAY = 86400000;
setTimeout(blastOff, MILLISECONDS_IN_A_DAY);
```

Clean Code

nazewnictwo

- używaj nazw z domeny problemu

```
function isCompanyDefunct()
{
    /*...*/
}
```

```
function isFallenFlag()
{
    /*...*/
}
```

Clean Code

nazewnictwo

- wytłumacz co tam masz

Bad

```
var address = 'One Infinite Loop, Cupertino 95014’;
var cityZipCodeRegex = /^[^,\\]+[,\\\s]+(.+?)\s*(\d{5})?$/;
saveCityZipCode(address.match(cityZipCodeRegex)[1], address.match(cityZipCodeRegex)[2]);
```

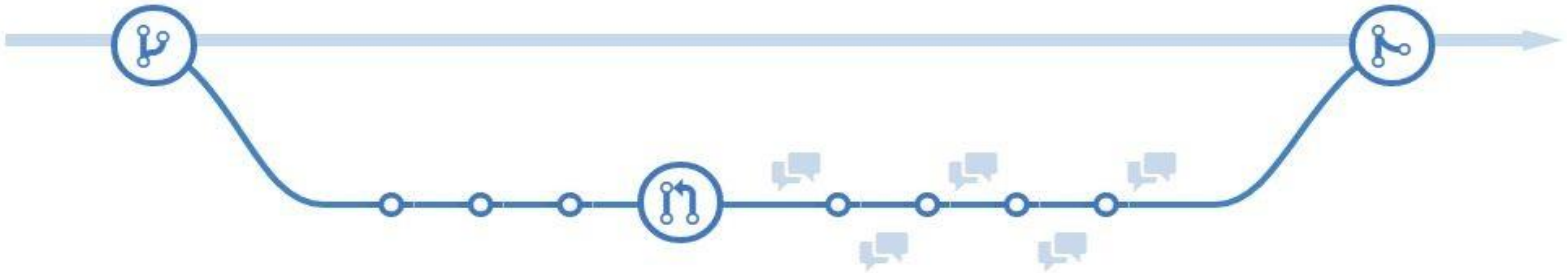
Good

```
var address = 'One Infinite Loop, Cupertino 95014’;
var cityZipCodeRegex = /^[^,\\]+[,\\\s]+(.+?)\s*(\d{5})?$/;
var [, city, zipCode] = address.match(cityZipCodeRegex) || [];
saveCityZipCode(city, zipCode);
```

Clean Code

GitHub

- dyskusje nad pull requestami



Clean Code

funkcje

- funkcja - czasownik; argument - rzeczownik

```
function divideNumbers(dividend, divisor) {
    if (divisor == 0) {
        throw new InvalidArgumentException("Division by zero!");
    }
    return dividend / divisor;
}
```

Clean Code

funkcje - niech opisuje dokładnie zachowanie

Bad

```
function addToDate(date, month) {
  // ...
}
var date = new Date();
addToDate(date, 1);
```

Good

```
function addMonthToDate(month, date) {
  // ...
}
var date = new Date();
addMonthToDate(1, date);
```

Clean Code

funkcje

- niech robi tylko jedną rzecz

Bad

```
function createFile(name, temp) {
  if (temp) {
    fs.create(`./temp/${name}`);
  } else {
    fs.create(name);
  }
}
```

Good

```
function createFile(name) {
  fs.create(name);
}

function createTempFile(name) {
  createFile(`./temp/${name}`);
}
```

Clean Code

funkcje

- jedna operacja, mało zagnieżdżeń, mało argumentów

```
function calculateSecretValue(data) {
    var numberOfDivisions = 0;
    var hashedValue = 0;
    $.each(arrayToSum, function(index, component) {  numberOfDivisions =
        numberOfDivisions + component;
    });
    for (i=0; i<numberOfDivisions; i++) {
        if (numberOfDivisions == 0) {
            throw new InvalidArgumentException("Division by zero!");
        }
        hashedValue = hashedValue + i / numberOfDivisions;
    }
    return hashedValue;
}
```

Clean Code

funkcje

- jedna operacja, mało zagnieżdżeń, mało argumentów

```
function calculateSecretHash(valueToHash) {
    var numberOfDivisions = sumValuesOfArrayElements(valueToHash);
    return hashNumbers(numberOfDivisions);
}

function sumValuesOfArrayElements(arrayToSum) {
    var sumOfValues = 0;
    $.each(arrayToSum, function(index, component) {
        sumOfValues = sumOfValues + component;
    });
    return sumOfValues;
}
```

Clean Code

funkcje

- jedna operacja, mało zagnieżdżeń, mało argumentów

```
function hashNumbers(numberOfDivisions) {
    var hashedValue = 0;
    for (i=0; i<numberOfDivisions; i++) {
        hashedValue = hashedValue + divideNumbers(i, numberOfDivisions);
    }
    return hashedValue;
}

function divideNumbers(dividend, divisor) {
    if (divisor == 0) {
        throw new InvalidArgumentException("Division by zero!");
    }
    return dividend / divisor;
}
```

Clean Code

funkcje - warunki zamykaj w kapsułki

Bad

```
if (fsm.state === 'fetching' && isEmpty(listNode)) {
  // ...
}
```

Good

```
function shouldShowSpinner(fsm, listNode) {
  return fsm.state === 'fetching' && isEmpty(listNode);
}

if (shouldShowSpinner(fsmInstance, listNodeInstance)) {
  // ...
}
```

Clean Code
funkcje

- unikaj zaprzeczeń

Bad

```
function isDOMNodeNotPresent(node)
{
  // ...
}

if (!isDOMNodeNotPresent(node)) {
  // ...
}
```

Good

```
function isDOMNodePresent(node) {
  // ...
}

if (isDOMNodePresent(node)) {
  // ...
}
```

Clean Code

funkcje

- upraszczaj skracaj kod

Bad

```
function getFirstName(name) {
  if (name) {
    return name;
  } else {
    return 'brak'
  }
}
```

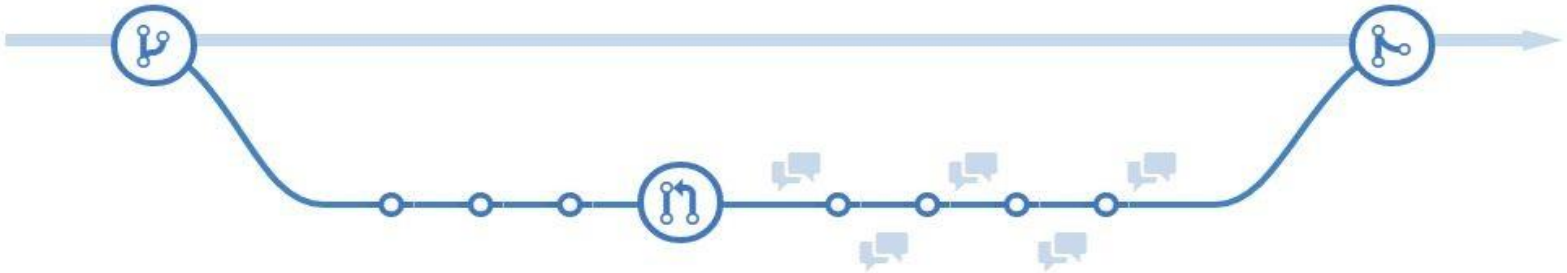
Good

```
function getFirstName(name) {
  return name || 'brak'
}
```

Clean Code

GitHub

- dyskusje nad pull requestami



Clean Code

komentarze

- czytelny kod nie wymaga komentarzy

```
/*...*/
// check if vehicle is exceeding speed limit
if (speed > 15) {
    /*...*/
}
/*...*/
```

```
if (isExceedingSpeedLimit(speed)) {
    /*...*/
}
function isExceedingSpeedLimit(currentSpeed) {
    var speedLimit = 15;
    return currentSpeed > speedLimit;
}
```

Clean Code

komentarze

- specyficzne wykorzystanie komentarzy w JavaScript

```
/**
 *  @param {Array} arrayToSum
 *  @return {Number}
 */
function sumValuesOfArrayElements(arrayToSum)
{
    sumOfValues = 0;
    $.each(arrayToSum, function(index, component) {
        sumOfValues = sumOfValues + component;
    });
    return sumOfValues;
}
```

Clean Code

komentarze

- komentarze, które nic nie wnoszą

```
// Number of days per standard week
var numberOfDaysInWeek = 7;

/**
 * Returns number of days in a week
 * @param {Number} weekNumber number of the week in year
 * @return {Number} number of days in chosen week
 */
function getNumberOfDaysInWeek(weekNumber)
{
    /*...*/
}
```

Clean Code

komentarze

- komentarze, które nic nie wnoszą

```
function divideNumbers(dividend, divisor)
{
    // Świętosław <swietoslaw@example.com> added it to handle zeros
    if (divisor == 0) {
        throw new InvalidArgumentException("Division by zero!");
    }
    return dividend / divisor;
}
```

Clean Code

komentarze

- komentarze, które nic nie wnoszą

```
//function someSillyObsolateCode(args)
//{
//      $.each (args, function(key, arg) {
//          args[key] = arg*arg;
//      }
//      return args;
//}
```

Clean Code

komentarze

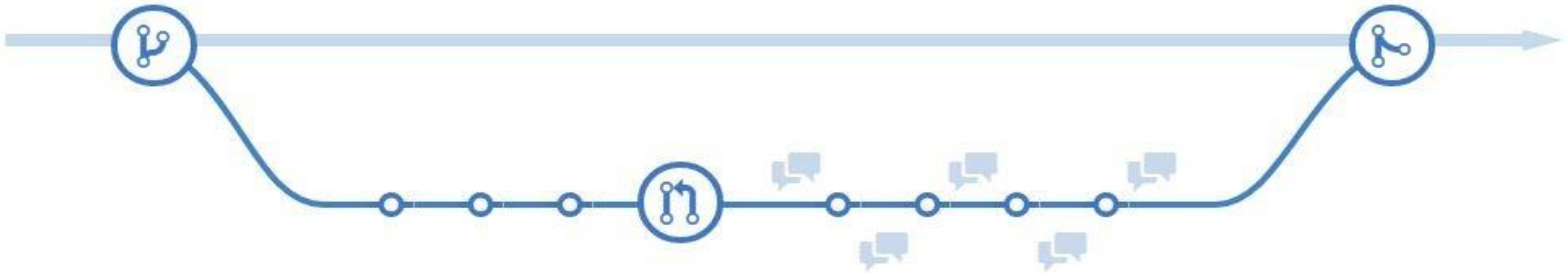
- komentarze, które przekazują ważne informacje

```
//
// Dear maintainer:
//
// Once you are done trying to 'optimize' this routine,
// and have realized what a terrible mistake that was,
// please increment the following counter as a warning
// to the next guy:
//
// total_hours_wasted_here = 42
//
```

Clean Code

GitHub

- dyskusje nad pull requestami



Clean Code

formatowanie

- oddzielaj sekcje kodu

```
function divideNumbers(dividend, divisor) {
    if (divisor == 0) {
        throw new InvalidArgumentException("Division by zero!");
    }
    return dividend / divisor;
}
function isNotExceedingSpeedLimit(currentSpeed) {
    speedLimit = 15;
    return currentSpeed <= speedLimit;
}
function sumValuesOfArrayElements(arrayToSum) {
    sumOfValues = 0;
    $.each(arrayToSum, function(index, component) {
        sumOfValues = sumOfValues + component;
    });
```

Clean Code

formatowanie

- oddzielaj sekcje kodu

```
function divideNumbers(dividend, divisor) {
    if (divisor == 0) {
        throw new InvalidArgumentException("Division by zero!");
    }
    return dividend / divisor;
}

function isNotExceedingSpeedLimit(currentSpeed) {
    var speedLimit = 15;
    return currentSpeed <= speedLimit;
}

function sumValuesOfArrayElements(arrayToSum) {
    var sumOfValues = 0;
    $.each(arrayToSum, function(index, component) {
        sumOfValues = sumOfValues + component;
    });
```

Clean Code

formatowanie

- nie oddzielaj powiązanych fragmentów kodu

```
/**
 * @var {Integer}
 */
var speedLimit = 15;

/**
 * @var {Number}
 */
var currentSpeed = getCurrentSpeed();
```

```
var speedLimit = 15;
var currentSpeed = getCurrentSpeed();
```

Clean Code

formatowanie

- przestrzegaj standardów (Ctrl+Alt+L)

```
if(a == b )
return c;

function thisIsReallyBadlyFormatted(   a,b)
{
        return a *b;
}
```

```
if (a == b) {
    return c;
}

function thisIsReallyBadlyFormatted(a, b) {
    return a * b;
}
```

Clean Code

formatowanie

```
        <ul id="bigBarNavigation">
<li><a href="/">HOME</a>
    </li><li><a href="/contact">CONTACT US</a></li><li>
        <a href="/about">ABOUT US</a></li></ul>
```

Confusing mess...

```
<ul id="bigBarNavigation">
    <li><a href="/">HOME</a></li>
    <li><a href="/contact">CONTACT US</a></li>
    <li>
        <a href="/about">ABOUT US</a>
        <div class="subMenu">
                <!-- Just an example to
                     show indentation -->
        </div>
    </li>
</ul>
```

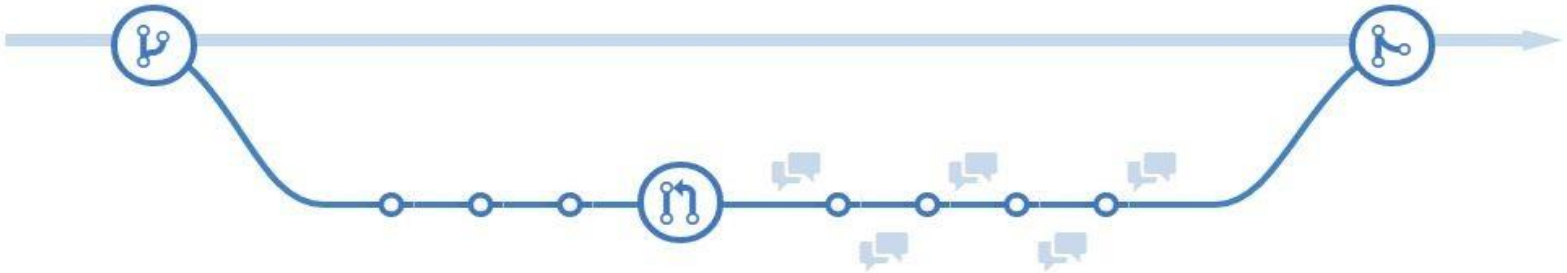
Nice and clean. mmmmmmmm...

nie twórz własnych standardów! (Ctrl+Alt+L)

Clean Code

GitHub

- dyskusje nad pull requestami



Clean Code

dobre praktyki

- DRY- do not repeat yourself

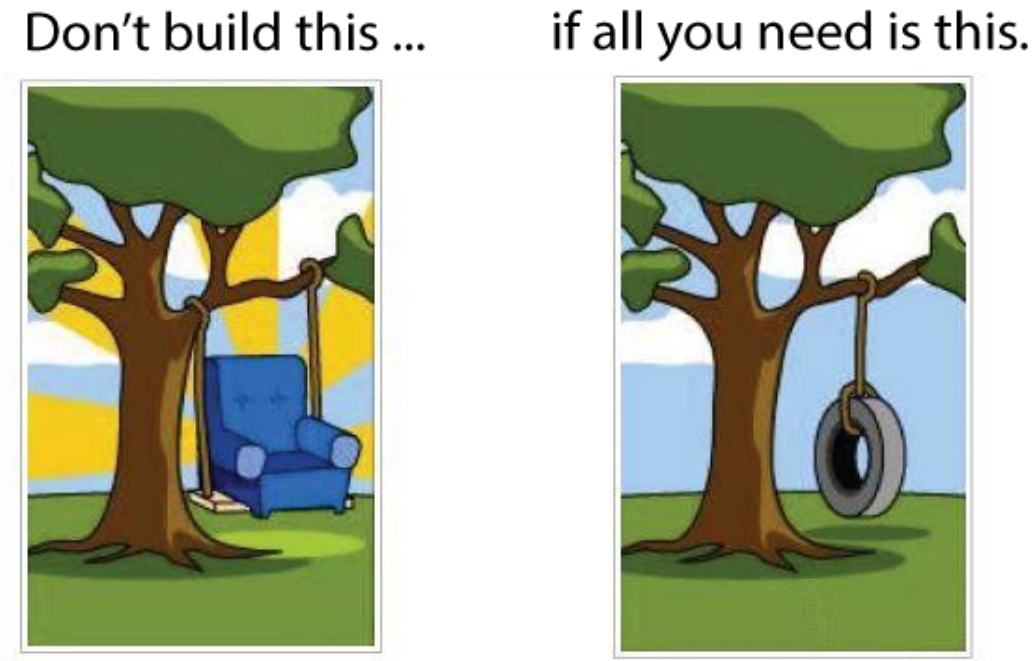
```
if (b != 0) {
    x = a / b;
}
if (d != 0) {
    y = c / d;
}
```

```
var x = divideNumbers(a, b);
var y = divideNumbers(c, d);

function divideNumbers(dividend, divisor) {
    if (divisor == 0) {
        throw new InvalidArgumentException("Division by zero!");
    }
    return dividend / divisor;
}
```

dobre praktyki

- YAGNI - you aren't gonna need it



Clean Code

dobre praktyki

- YAGNI - you aren't gonna need it

- nie zapełniaj spiżarni
- nie przewidzisz przyszłości
- skup się tym, o czym wiesz
- jeśli masz wątpliwości - pytaj

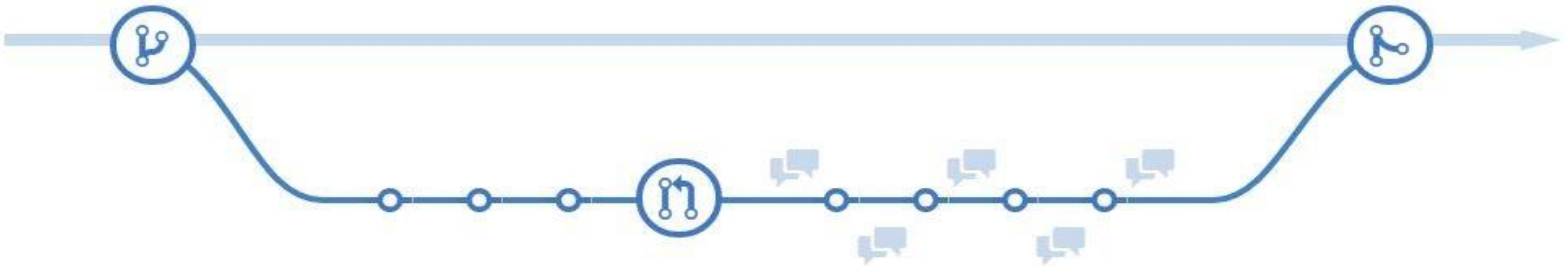
Clean Code

dobre praktyki

- KISS - keep it simple stupid
- dobry rzemieślnik wybiera proste rozwiązania
- nadmierna inżynieria overengineering

GitHub

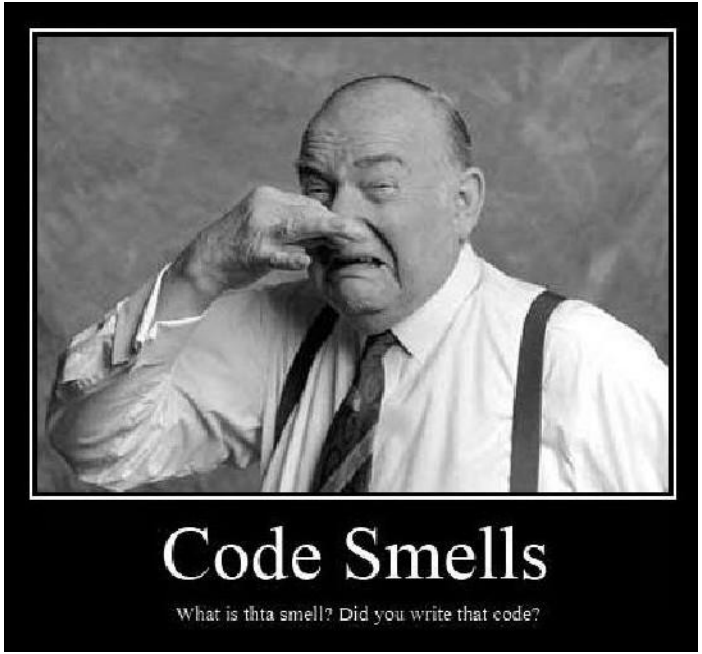
- dyskusje nad pull requestami



Clean Code

refactoring

- Code Smells
- Alt+Enter
- Ctrl+Alt+ Shift+T
- Jeśli coś jest głupie, ale działa, to nie jest głupie



Thanks!!

Any questions?

contact@lukaszduda.com