



Czysty Kod

Czyli jak programować aby robić to dobrze?

Krzysztof Pałka

Robert C. Martin

CLEAN CODE
CLEAN CODE

CZYSTY KOD

PODRĘCZNIK
DOBREGO PROGRAMISTY



Poznaj najlepsze metody tworzenia doskonałego kodu

Jak pisać dobry kod, a zły przekształcić w dobry?

Jak formatować kod, aby osiągnąć maksymalną czytelność?

Jak implementować pełną obsługę błędów bez zasmiecania logiki kodu?

Hellon





Schedule

- Po co nam dobry kod
- Znaczące nazwy
- Funkcje
- Komentarze
- Formatowanie
- Style Guide

Po co nam “dobry” kod?

I czym on właściwie jest?

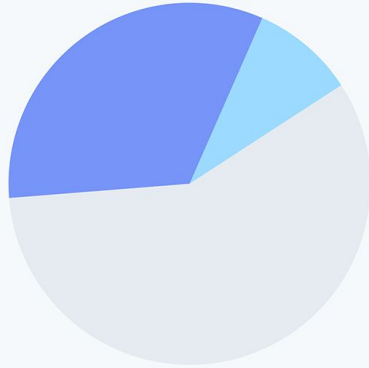




Koszty “niedobrego” kodu

- strata czasu
- nadgodziny
- żmudne debugowanie
- frustracja
- itp...
- 85 miliardów dolarów rocznie?

THE DEVELOPER WORK WEEK



● 13.5 hours
Technical debt

● 3.8 hours
Bad code

41.1 total hours
Average developer work week

„Rzeczywiście, stosunek czasu spędzonego na czytaniu do pisania wynosi ponad 10 do 1. Nieustannie czytamy stary kod w ramach prac nad nowym kodem. ... [Dlatego] ułatwienie czytania ułatwia pisanie ”.

Robert C. Martin

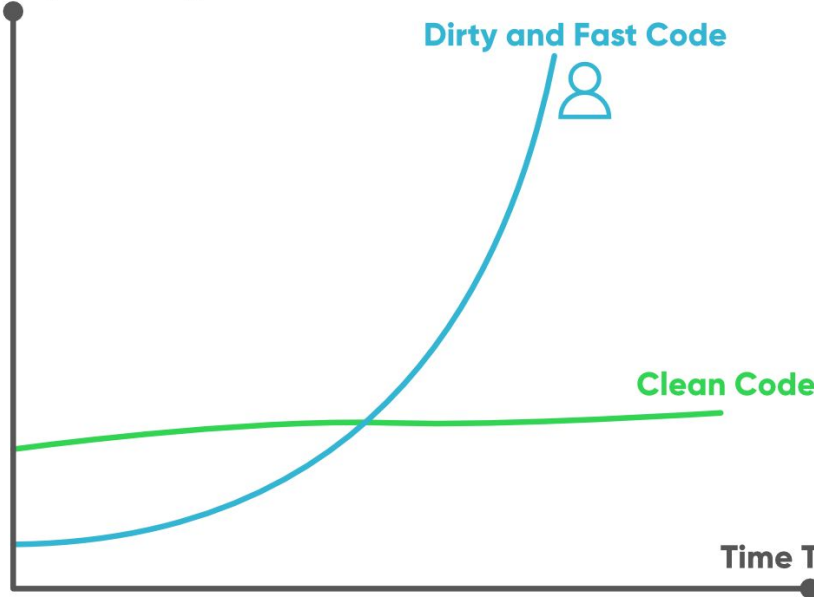
Cost per change

Dirty and Fast Code

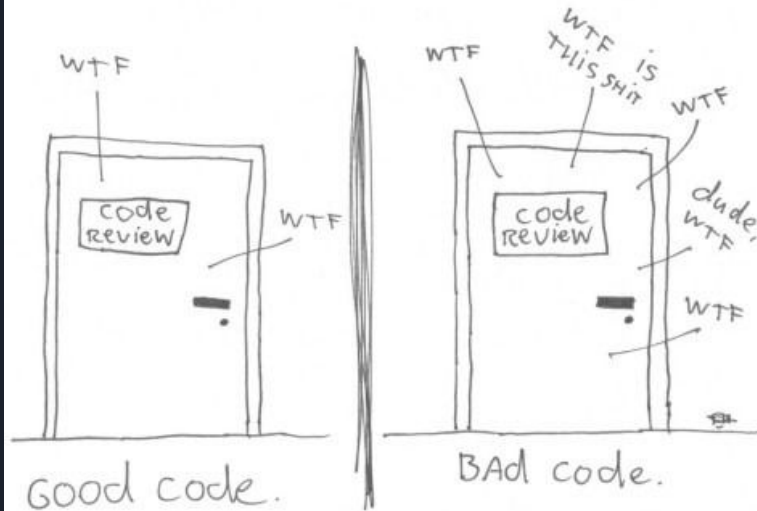


Clean Code

Time Taken



The ONLY valid measurement
of code quality: WTFs/minute



Znaczące nazwy

Czy to ma znaczenie?



„Nie ma gorszego powodu użycia nazwy c niż ten, że a i b są już zajęte.”.

Robert C. Martin



Jaka powinna być nazwa?



Jaka powinna być nazwa?

- po angielsku

```
// nie wygląda to zbyt dobrze
int znak;
long long dataWydania;
String imie;
```

```
boolean czyJestPierwsza(int kandydat)
void ustawZnak(char znak)
int getWiek()
```

```
// that's better
int sign;
long long issueDate;
String name;
```

```
boolean isPrime(int candidate)
void setSign(char sign)
int getAge()
```



Jaka powinna być nazwa?

- po angielsku
- self explanatory

```
// bardzo ogólna nazwa, może znaczyć wszystko  
int t;
```

```
// łatwo domyślić się kontekstu, znamy jednostkę  
int daysSinceCreation ;  
int daysSinceModification ;
```

```
int remainingTimeToEnd ;  
int timeInMillisFromStart ;
```




Jaka powinna być nazwa?

- po angielsku
- self explanatory
- gramatycznie poprawna

```
Date d;  
GameBoard b;  
Router r;  
PC pc_1;
```

```
void dStHFunc()  
int get_t_v()  
void set()  
boolean good()
```

```
Date modificationDate ;  
GameBoard board ;  
Router router ;  
PC pc ;
```

```
void doSomething()  
int getThisVariable()  
void setSomethink()  
boolean isGood()
```



Jaka powinna być nazwa?

- po angielsku
- self explanatory
- gramatycznie poprawna

```
Date d;  
GameBoard b;  
Router r;  
PC pc_1;
```

```
void dSthFunc()  
int get_t_v()  
void set()  
boolean good()
```

```
Date modificationDate ;  
GameBoard board ;  
Router router ;  
PC pc;
```

```
void doSomething()  
int getThisVariable () // akcesor  
void setSomethink () // mutator  
boolean isGood () // predykat
```



Jaka powinna być nazwa?

- po angielsku
- self explanatory
- gramatycznie poprawna
- bez odwzorowania mentalnego

```
String[] l = {"Austin", "New York"};

for (int i = 0; i < l.length; i++) {
    String li = l[i];
    doStuff();
    doSomeOtherStuff();
    // ...
    // ...
    // a czym jest `li`?
    dispatch(li);
}
```

```
String[] locations = {"Austin", "New York"};

for (String location : locations) { // FOREACH
    doStuff();
    doSomeOtherStuff();
    // ...
    // ...
    // ...
    dispatch(location);
}
```



Jaka powinna być nazwa?

- po angielsku
- self explanatory
- gramatycznie poprawna
- bez odwzorowania mentalnego
- nic nie koduje

// notacja węgierska

// koduje typ

`int` iLiczba;

`long` l_Liczba;

`string` s_liczba;

// wiele nazw na to samo

```
public class Part {  
    String m_dsc; // Opis tekstowy  
    void setName(String name) {  
        m_dsc = name;  
    }  
}
```

// tutaj praktycznie nie da się pomylić

```
public class Part {  
    String description;  
    void setDescription(String description)  
    {  
        this.description = description;  
    }  
}
```



Jaka powinna być nazwa?

- po angielsku
- self explanatory
- gramatycznie poprawna
- bez odwzorowania mentalnego
- nic nie koduje
- nie wymaga komentarza

} Opisowa



Jacy powinniśmy być my?



Jacy powinniśmy być my?

KONSEKWENTNI

Konwencje nazewnnicze

Multiple-word identifier formats	
Formatting	Name(s)
twowords	flat case ^{[13][14]}
TWOWORDS	upper flat case ^[13]
twoWords	(lower) camelCase, dromedaryCase
TwoWords	PascalCase, Upper Camel Case, StudlyCase ^[15]
two_words	snake_case, pothole_case
TWO_WORDS	SCREAMING_SNAKE_CASE, MACRO_CASE, CONSTANT_CASE
two_Words	camel_Snake_Case
Two_Words	Pascal_Snake_Case
two-words	kebab-case, dash-case, lisp-case
two words	doner case
TWO-WORDS	TRAIN-CASE, COBOL-CASE, SCREAMING-KEBAB-CASE
Two-Words	Train-Case, ^[13] HTTP-Header-Case ^[16]

Konwencje nazewnnicze

Multiple-word identifier formats	
Formatting	Name(s)
twowords	flat case ^{[13][14]}
TWOWORDS	upper flat case ^[13]
twoWords	(lower) camelCase, dromedaryCase
TwoWords	PascalCase, Upper Camel Case, StudlyCase ^[15]
two_words	snake_case, pothole_case
TWO_WORDS	SCREAMING_SNAKE_CASE, MACRO_CASE, CONSTANT_CASE
two_Words	camel_Snake_Case
Two_Words	Pascal_Snake_Case
two-words	kebab-case, dash-case, lisp-case
two words	doner case
TWO-WORDS	TRAIN-CASE, COBOL-CASE, SCREAMING-KEBAB-CASE
Two-Words	Train-Case, ^[13] HTTP-Header-Case ^[16]

metody, zmienne
klasy

stałe

```
getBackground()  
float myWidth;  
  
class ImageSprite {}  
  
final int MAX_PARTICIPANTS = 10;
```

Funkcje

Oraz dlaczego są za duże





Funkcje

„Pierwsza zasada dotycząca konstruowania funkcji jest taka, że powinny być małe. Druga zasada mówi, że powinny być mniejsze, niż są.”.

Robert C. Martin



Jaka powinna być idealna funkcja?

- krótka (max 20-30 wierszy)


```

public class PrimeGenerator
{
    private static boolean[] crossedOut;
    private static int[] result;

    public static int[] generatePrimes(int maxValue) {
        if (maxValue < 2) return new int[0];
        else {
            UncrossIntegersIntoResult(maxValue);
            crossOutMultiples();
            putUncrossedIntegersIntoResult();
            return result;
        }
    }

    private static void crossOutMultiples(int maxValue) {
        crossedOut = new boolean[maxValue + 1];
        for (int i = 2; i < crossedOut.length; i++)
            crossedOut[i] = false;
    }

    private static void crossOutMultiples() {
        int limit = determineIterationLimit();
        for (int i = 2; i <= limit; i++)
            if (notCrossed(i)) crossOutMultiplesOf(i);
    }

    private static int determineIterationLimit() {
        // Każda wielokrotność w tablicy ma dzielnik będący liczbą pierwszą
        // mniejszą lub równą pierwiastkowi kwadratowemu wielkości tablicy,
        // więc nie musimy wykreślać wielokrotności większych od tego pierwiastka.
        double iterationLimit = Math.sqrt(crossedOut.length);
        return (int) iterationLimit;
    }
}

```

```

private static void crossOutMultiplesOf(int i) {
    for (int mult=2*i; mult<crossedOut.length; mult+= i)
        crossedOut[mult] = true;
}

private static boolean notCrossed(int i)
{
    return crossedOut[i] == false;
}

private static void putUncrossedIntegersIntoResult() {
    result = new int[numberOfUncrossedIntegers()];
    for (int j = 0, i = 2; i < crossedOut.length; i++)
        if (notCrossed(i)) result[j++] = i;
}

private static int numberOfUncrossedIntegers() {
    int count = 0;
    for (int i = 2; i < crossedOut.length; i++)
        if (notCrossed(i)) count++;
    return count;
}
}

```

```
public static int[] generatePrimes(int maxValue) {
    if (maxValue < 2) return new int[0];
    else {
        uncrossIntegersUpTo(maxValue);
        crossOutMultiples();
        putUncrossedIntegersIntoResult();
        return result;
    }
}
```

```
private static void uncrossIntegersUpTo(int maxValue) {
    crossedOut = new boolean[maxValue + 1];
    for (int i = 2; i < crossedOut.length; i++)
        crossedOut[i] = false;
}
```

```
private static void crossOutMultiples() {
    int limit = determineIterationLimit();
    for (int i = 2; i <= limit; i++)
        if (notCrossed(i)) crossOutMultiplesOf(i);
}
```

```
private static void putUncrossedIntegersIntoResult() {
    result = new int[numberOfUncrossedIntegers()];
    for (int j = 0, i = 2; i < crossedOut.length; i++)
        if (notCrossed(i)) result[j++] = i;
}
```

```
private static int determineIterationLimit () {
    // Każda wielokrotność w tablicy ma dzielnik
    // będący liczbą pierwszą mniejszą lub równą
    // pierwiastkowi kwadratowemu wielkości tablicy,
    // więc nie musimy wykreślać wielokrotności
    // większych od tego pierwiastka.
    double iterationLimit =
    Math.sqrt(crossedOut.length);
    return (int) iterationLimit ;
}
```

```
private static void crossOutMultiplesOf (int i) {
    for (int mult=2*i; mult<crossedOut.length; mult+=
    i)
        crossedOut[mult] = true;
}
```

```
private static int numberOfUncrossedIntegers ()
{
    int count = 0;
    for (int i = 2; i < crossedOut.length; i++)
        if (notCrossed(i)) count++;
    return count;
}
```

```
private static boolean notCrossed (int i)
{
    return crossedOut[i] == false;
}
```



Jaka powinna być idealna funkcja?

- krótka (max 20-30 wierszy)
- wykonuje jedną czynność



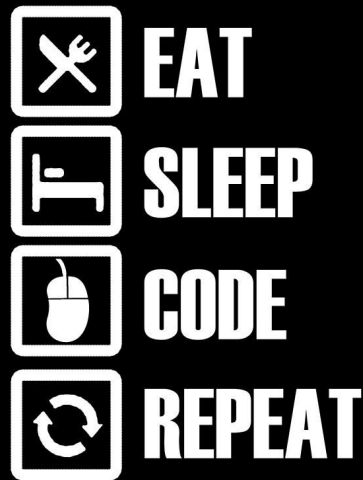
Jaka powinna być idealna funkcja?

- krótka (max 20-30 wierszy)
- wykonuje jedną czynność
- używa jednego poziomu abstrakcji



Jaka powinna być idealna funkcja?

- krótka (max 20-30 wierszy)
- wykonuje jedną czynność
- używa jednego poziomu abstrakcji





Jaka powinna być idealna funkcja?

- krótka (max 20-30 wierszy)
- wykonuje jedną czynność
- używa jednego poziomu abstrakcji

```
public void repeat () {  
    eat () ;  
    sleep () ;  
  
    turnOnPC () ;  
    typeOnKeyboard (USER_PASSWORD) ;  
    typeOnKeyboard ( "cout<<\nHello Word! \n"<<endl ) ;  
    clickMouse (BUILD_RUN_BUTTON) ;  
    turnOffPC () ;  
  
    repeat () ;  
}
```

```
public void repeat () {  
    eat () ;  
    sleep () ;  
    code () ;  
  
    repeat () ;  
}
```

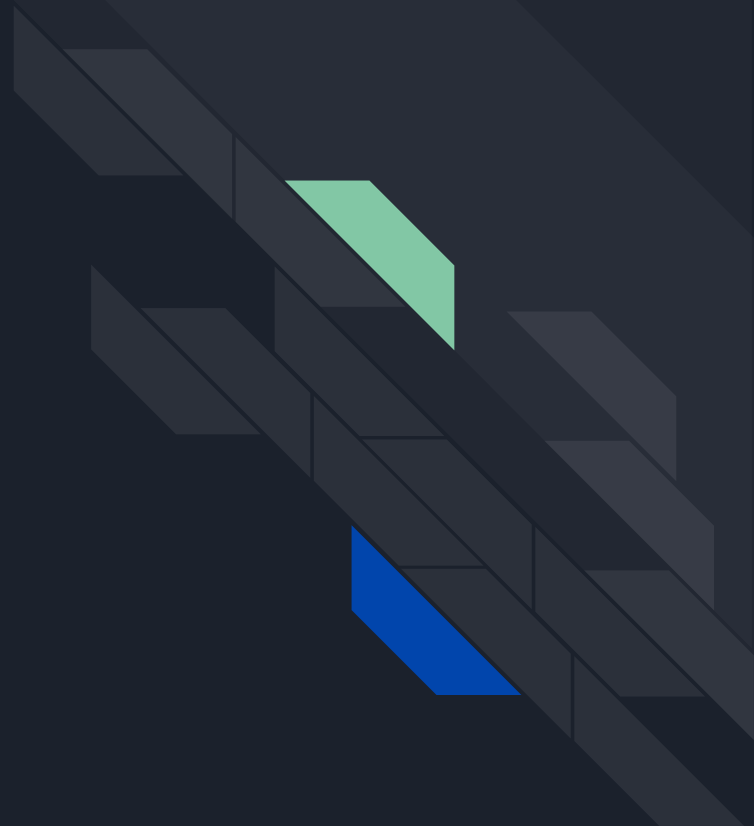


Jaka powinna być idealna funkcja?

- krótka (max 20-30 wierszy)
- wykonuje jedną czynność
- używa jednego poziomu abstrakcji

Komentarze

Szminka dla złego kodu





Komentarze

„Obecność komentarzy zawsze sygnalizuje nieporadność programisty. [...] Gdy uznamy, że konieczne jest napisanie komentarza, należy pomyśleć, czy nie istnieje sposób na wyrażenie tego samego w kodzie”.

Robert C. Martin



Komentarze - te złe

```
// Sprawdzenie, czy pracownik ma prawo do wszystkich korzyści  
if ((employee.flags & HOURLY_FLAG) && (employee.age > 65))  
  
    -- VS --  
  
if (employee.isEligibleForFullBenefits())
```



Komentarze - te złe

```
// Metoda użytkowa kończąca pracę, gdy this.closed ma wartość true. Zgłasza wyjątek,  
// jeżeli przekroczony zostanie czas oczekiwania.  
public synchronized void waitForClose(final long timeoutMillis) throws Exception {  
    if(!closed) {  
        wait(timeoutMillis) ;  
        if(!closed)  
            throw new Exception("MockResponseSender could not be closed" );  
    }  
}
```

Komentarze - te złe

```
// Metoda użytkowa kończąca pracę, gdy this.closed ma wartość true. Zgłasza wyjątek,  
// jeżeli przekroczony zostanie czas oczekiwania.  
public synchronized void waitForClose(final long timeoutMillis) throws Exception {  
    if(!closed) {  
        wait(timeoutMillis) ;  
        if(!closed)  
            throw new Exception("MockResponseSender could not be closed" );  
    }  
}
```

Junior devs writing comments:





Komentarze - te złe

```
/*  
 * Classname  
 *  
 * Version info  
 *  
 * Copyright notice  
 */  
  
// System.out.println(tab[i]);
```

```
/**  
 * 2021-03-06: Renamed clean to cleanCode (DL)  
 * 2020-01-03: Changed return value (LB)  
 * 2019-05-12: Added clean method (DL)  
 */
```

```
doStuff() ;  
// doOtherStuff();  
// doSomeMoreStuff();  
// doSoMuchStuff();
```



Komentarze - te złe

```
while(isRunning){  
    // ----- processing package -----  
    for(HashMap.Entry<String, Socket> one : sockets.entrySet()){  
        Socket s = one.getValue();  
        Package p = s.receivePackageFromPort();  
        if(p != null){  
            if(printStat) System.out.print(routerName + ": received package " + p.toString());  
            proceedPackage(p);  
        }  
    }  
  
    // ----- waiting a while -----  
    try{  
        Thread.sleep(tickTime);  
        timeFromStart += tickTime;  
    }catch(InterruptedException e) {  
        return;  
    }  
}
```



Komentarze - te złe

```
while(isRunning){
    // ----- processing package -----
    for(HashMap.Entry<String, Socket> one : sockets.entrySet()){
        Socket s = one.getValue();
        Package p = s.receivePackageFromPort();
        if(p != null){
            if(printStat) System.out.print(routerName + ": received package " + p.toString());
            proceedPackage(p);
        }
    }

    // ----- waiting a while -----
    try{
        Thread.sleep(tickTime);
        timeFromStart += tickTime;
    }catch(InterruptedException e) {
        return;
    }
}
```

```
while(isRunning){
    for(HashMap.Entry<String, Socket> socket: sockets.entrySet()){
        processPackageIfNecessary(socket);
    }
    waitWhile();
}
```



Komentarze - te dobre

```
// Copyright (C) 2003,2004,2005 by Object Mentor, Inc. All rights reserved.  
// Released under the terms of the GNU General Public License version 2 or later
```

```
// SimpleDateFormat nie jest bezpieczna dla wątków,  
// więc musimy każdy obiekt tworzyć niezależnie.  
SimpleDateFormat df = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z");  
df.setTimeZone(TimeZone.getTimeZone("GMT"));
```

```
String listItemContent = match.group(3).trim();  
// Wywołanie trim jest naprawdę ważne. Usuwa początkowe  
// spacje, które mogą spowodować, że element będzie  
// rozpoznany jako kolejna lista
```

```
// TODO to może być zoptymalizowane
```




Komentarze - te dobre

```
// Copyright (C) 2003,2004,2005 by Object Mentor, Inc. All rights reserved.  
// Released under the terms of the GNU General Public License version 2 or later
```

Nota prawna

```
// SimpleDateFormat nie jest bezpieczna dla wątków,  
// więc musimy każdy obiekt tworzyć niezależnie.  
SimpleDateFormat df = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z");  
df.setTimeZone(TimeZone.getTimeZone("GMT"));
```

Ostrzeżenie

```
String listItemContent = match.group(3).trim();  
// Wywołanie trim jest naprawdę ważne. Usuwa początkowe  
// spacje, które mogą spowodować, że element będzie  
// rozpoznany jako kolejna lista
```

Podkreślenie
istotności operacji

```
// TODO to może być zoptymalizowane
```

TODO

Formatowanie

Czy spacje mają znaczenie?



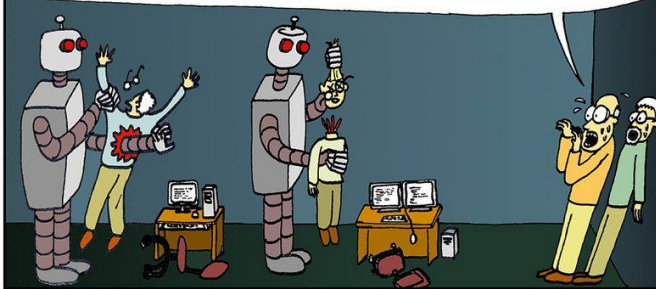
There are two types of people.

```
if (Condition)
{
    Statements
    /*
     *
     */
}
```

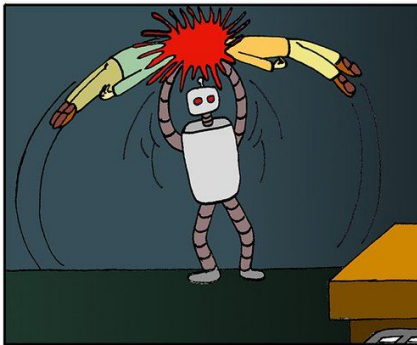
```
if (Condition) {
    Statements
    /*
     *
     */
}
```

Programmers will know.

OH NO! THE ROBOTS ARE KILLING US!!!



BUT WHY?!? WE
NEVER PROGRAMMED
THEM TO DO THIS!!!



```
static bool isCrazyMurderingRobot = false;
```

```
void interact_with_humans (void){  
    if(isCrazyMurderingRobot = true)  
        kill(humans);  
    else  
        be_nice_to(humans);  
}
```

CTRL+ALT+L



Style guide

- limit znaków w linii
- limit linii w pliku, funkcji
- reguły dla białych znaków
- ustawienia nawiasów klamrowych
- nazewnictwo zmiennych i funkcji
- komentarze
- dokumentacja
- zakazane konstrukcje



Style guide

- limit znaków w linii
- limit linii w pliku, funkcji
- reguły dla białych znaków
- ustawienia nawiasów klamrowych
- nazewnictwo zmiennych i funkcji
- komentarze
- dokumentacja
- zakazane konstrukcje

```
cout<< (+++++x) --<<endl;
```



Style guide

- limit znaków w linii
- limit linii w pliku, funkcji
- reguły dla białych znaków
- ustawienia nawiasów klamrowych
- nazewnictwo zmiennych i funkcji
- komentarze
- dokumentacja
- zakazane konstrukcje



To tylko konwencje, ale
warto się ich
konsekwentnie trzymać



Style guide - przykłady

- [Google Java Style Guide](#)
- [Oracle Java Code Conventions](#)
- [Java Style Guidelines](#) - od twórców JDK
- [Cornell University Java Code Style](#)

Style guide - od twórców JDK

Dos

```
void method() {  
    ""  
}
```

```
try {  
    something();  
} catch (Exception e) {  
    ""  
}
```

```
for (int[] row : matrix) {  
    for (int val : row) {  
        sum += val;  
    }  
}
```

Don'ts

```
// Wrong placement of opening brace  
void method()  
{  
    ""  
}
```

```
// Newline in front of catch should be avoided  
try {  
    something();  
}  
catch (Exception e) {  
    ""  
}
```

```
// Braces should be used  
if (flag)  
    // Restore x  
    x = 1;
```

```
// Use braces if block comes last in enclosing block  
// to avoid accidentally indenting the closing brace.  
for (int[] row : matrix) {  
    for (int val : row)  
        sum += val;  
}
```



gh-pages ▾

styleguide

intellij-java-google-style.xml

Go to file



eccentricon Revert "Project import generated by Copybara." ... ✓

Latest commit 505ba68 on 30 Jan 2018 History

4 contributors



598 lines (598 sloc) | 21.6 KB

Raw

Blame



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <code_scheme name="GoogleStyle">
3   <option name="OTHER_INDENT_OPTIONS">
4     <value>
5       <option name="INDENT_SIZE" value="2" />
6       <option name="CONTINUATION_INDENT_SIZE" value="4" />
7       <option name="TAB_SIZE" value="2" />
8       <option name="USE_TAB_CHARACTER" value="false" />
9       <option name="SMART_TABS" value="false" />
10      <option name="LABEL_INDENT_SIZE" value="0" />
11      <option name="LABEL_INDENT_ABSOLUTE" value="false" />
12      <option name="USE_RELATIVE_INDENTS" value="false" />
13    </value>
14  </option>
15  <option name="INSERT_INNER_CLASS_IMPORTS" value="true" />
16  <option name="CLASS_COUNT_TO_USE_IMPORT_ON_DEMAND" value="999" />
17  <option name="NAMES_COUNT_TO_USE_IMPORT_ON_DEMAND" value="999" />
18  <option name="PACKAGES_TO_USE_IMPORT_ON_DEMAND">
19    <value />
20  </option>
21  <option name="IMPORT_LAYOUT_TABLE">
22    <value>
```

Appearance & Behavior

Keymap

Editor

General

Code Editing

Font

Color Scheme

Code Style

Java

EditorConfig

Groovy

HTML

JSON

Kotlin

Markdown

Properties

Shell Script

XML

YAML

Other File Types

Inspections

File and Code Templates

File Encodings

Live Templates

File Types

Android Layout Editor

Copyright

Editor > Code Style > Java

Scheme: Default IDE

Tabs and Indents

Spaces

Wrapping and

Hard wrap at

Wrap on typing

Visual guides

Keep when reformatting

Line breaks

Comment at first column

Control statement in one line

Multiple expressions in one line

Simple blocks in one line

Simple methods in one line

Simple lambdas in one line

Simple classes in one line

Ensure right margin is not exceeded

Braces placement

In class declaration

In method declaration

In lambda declaration

Other

Extends/implements list

Align when multiline

Extends/implements keyword

Throws list

Align when multiline

Align 'throws' to method start

Throws keyword

Method declaration parameters

Align when multiline

New line after '('

Place ')' on new line



Copy to Project...

Duplicate...

Restore Defaults

Export

Import Scheme

IntelliJ IDEA code style XML

Eclipse XML Profile

JavaDoc

Imports

Arrangement

Code Generation

```
public class ThisIsASampleClass extends C1 implements I1, I2, I3, I4, I5 {  
    private int f1 = 1;  
    private String field2 = "";  
  
    public void foo1(int i1, int i2, int i3, int i4, int i5, int i6, int i7) {  
        // todo something  
        int  
        int i = 0;  
        int[] a = new int[]{1, 2, 0x0052, 0x0053, 0x0054};  
        int[] empty = new int[]{};  
        int var1 = 1;  
        int var2 = 2;  
        foo1(0x0051, 0x0052, 0x0053, 0x0054, 0x0055, 0x0056, 0x0057);  
        int x = (3 + 4 + 5 + 6) * (7 + 8 + 9 + 10) * (11 + 12 + 13 + 14 + 0xFFFFF)  
        String s1, s2, s3;  
        s1 = s2 = s3 = "012345678901456";  
        assert i + j + k + l + n + m <= 2 : "assert description";  
        int y = 2 > 3 ? 7 + 8 + 9 : 11 + 12 + 13;  
        super.getFoo().foo().getBar().bar();  
    }  
}
```

Tab1:

OK

Cancel

Apply

Przykład





Dziękuję za uwagę



Bibliografia

Ogólne źródła:

- Robert C. Martin "Clean Code"
- Sunil Kapil "Czysty kod w Pythonie"
- <https://www.samouczekprogramisty.pl/jak-pisac-kod-wysokiej-jakosci-w-jezyku-java/>
- <https://github.com/leonardolemie/clean-code-java#table-of-contents>



Bibliografia

Wstęp:

- <https://www.pullrequest.com/blog/cost-of-bad-code/>
- <https://www.osnews.com/story/19266/wtfsm/>
- <https://www.cnbc.com/2018/09/06/companies-worry-more-about-access-to-software-developers-than-capital.html>
- <https://blog.knoldus.com/keep-your-code-clean/>
- <https://www.goodreads.com/quotes/835238-indeed-the-ratio-of-time-spent-reading-versus-writing-is>



Bibliografia

Nazwy:

- https://pl.wikipedia.org/wiki/Notacja_w%C4%99gierska
- [https://en.wikipedia.org/wiki/Naming_convention_\(programming\)](https://en.wikipedia.org/wiki/Naming_convention_(programming))
- <https://ucgosu.pl/2020/04/style-guide-i-coding-standard-czy-to-juz-jak-osc/>



Bibliografia

Style Guide:

- <https://google.github.io/styleguide/javaguide.html#s3-source-file-structure>
- <https://github.com/google/styleguide/blob/gh-pages/intelij-java-google-style.xml>
- <https://www.oracle.com/technetwork/java/codeconventions-150003.pdf>
- <https://ucgosu.pl/2020/04/style-guide-i-coding-standard-czy-to-juz-jakosc/>
- <http://cr.openjdk.java.net/~alundblad/styleguide/index-v6.html>
- <https://www.cs.cornell.edu/courses/JavaAndDS/JavaStyle.html>



Bibliografia

Użyte przykłady:

- Robert C. Martin "Clean Code"
- <https://www.programiz.com/java-programming/examples/prime-number>
- <https://github.com/leonardolemie/clean-code-java#table-of-contents>
- <https://github.com/thekristopl>