# Czysty Kod

Czyli jak programować aby robić to dobrze?

Krzysztof Pałka

# Schedule

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

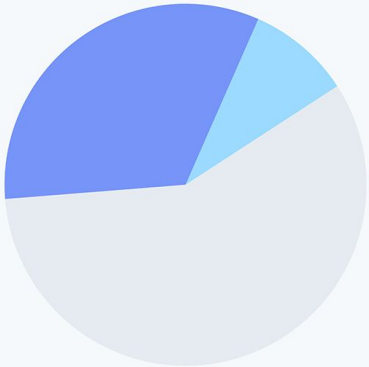
# Po co nam “dobry” kod?

I czym on właściwie jest?

# Koszty “niedobrego” kodu

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

## THE DEVELOPER WORK WEEK



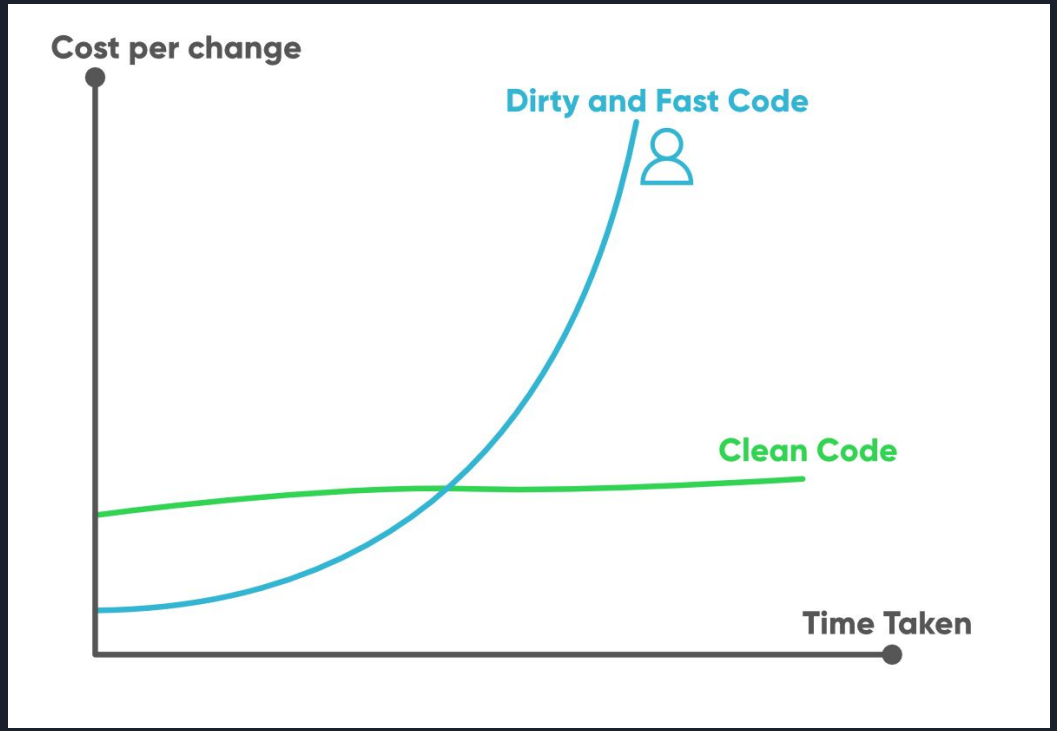
13.5 hours
Technical debt

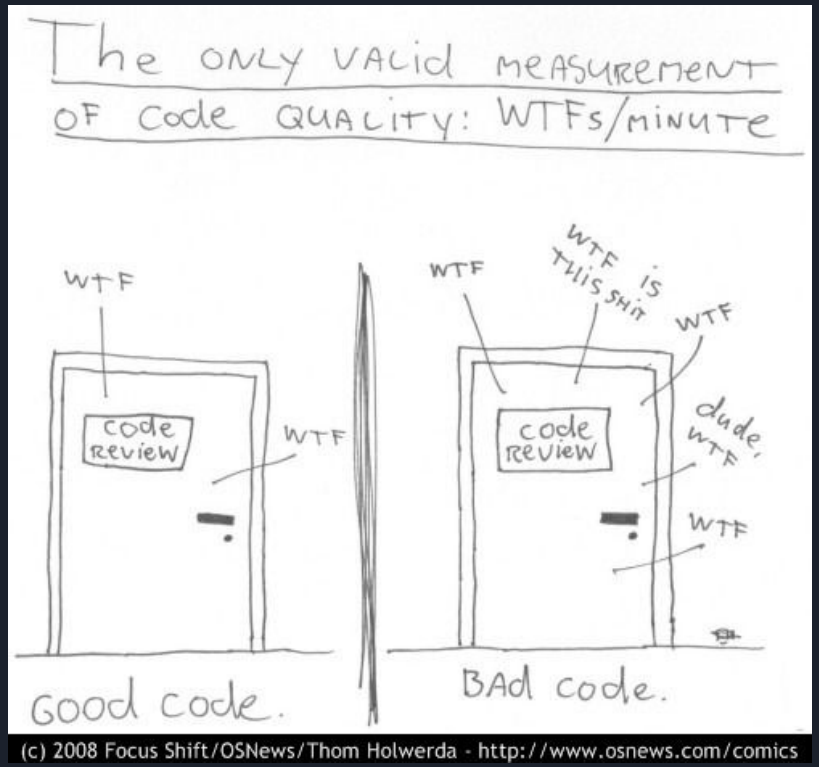
3.8 hours
Bad code

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
WTF
code review
WTF
Good code.
WTF
WTF IS THIS SHIT
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

# 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**

```
int main()
{
   string a, x, y, z ;
   int b=0, c=0, d, e[1000], i, j, n, k=1;
   cin>>a;

   n=a.length() ;
   for(i=1; i<=3; i++)
   {
       for(j=0; j<n; j++)
       {
           if(i==1)
           {
               if(j%2==0 && j%4!=0)
               {
                   cout<<a[j] ;
                   b=j;
               }
               else cout<<'-';
           }
// ...
```

# 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 explained

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

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

int remainingTimeToEnd;
int timeInMillisFromStart;
```

```java
public static boolean prime(int num) {
    boolean flag = false;
    for (int i = 2; i <= num / 2; ++i) {
        // condition for nonprime number
        if (num % i == 0) {
            flag = true;
            break;
        }
    }
    return !flag;
}

public static boolean isPrime(int candidate) {
    boolean foundedDivisor = false;
    for (int divisor = 2; divisor <= candidate / 2; divisor++) {
        if (candidate % divisor == 0) {
            foundedDivisor = true;
            break;
        }
    }
    return !foundedDivisor;
}
```

# Jaka powinna być nazwa?

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

```
// 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 explained
- gramatycznie poprawna
- bez odwzorowania mentalnego
- nic nie koduje
- nie wymaga komentarza

} Opisowa

# Jacy powinniśmy być my?

# Jacy powinniśmy być my?

KONSEKWENTNI

# Konwencje nazewnicze

**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 nazewnicze

**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?

# Jaka powinna być idealna funkcja?

- krótka (max 20-30 wierszy)

# Jaka powinna być idealna funkcja?

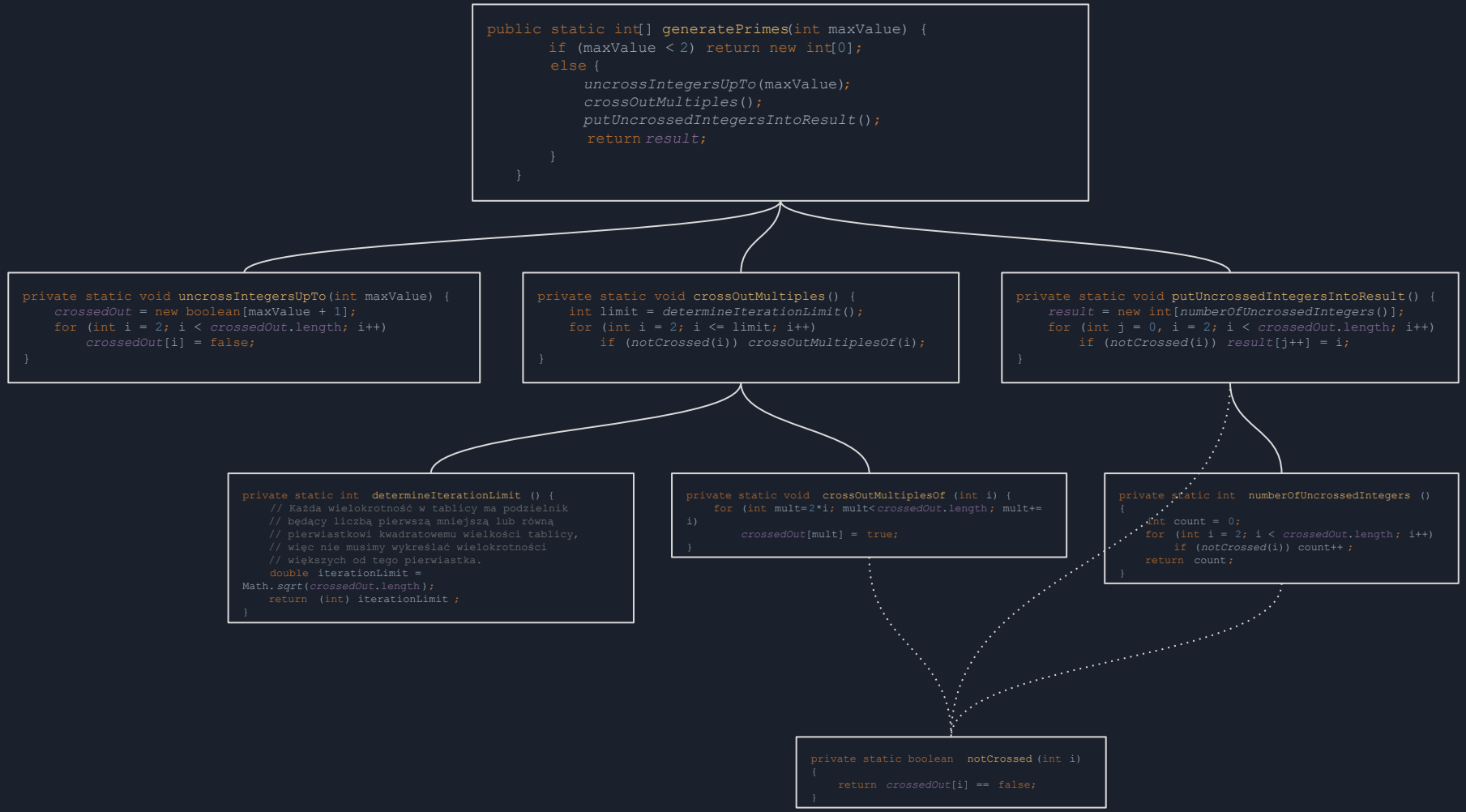
- krótka (max 20-30 wierszy)

```java
public static int[] generatePrimes(int maxValue)
{
   if (maxValue >= 2) { // Jedyny prawidłowy przypadek.
       // Deklaracje.
       int s = maxValue + 1; // Rozmiar tablicy.
       boolean[] f = new boolean[s];
       int i;
       // Inicjalizacja tablicy wartościami true.
       for (i = 0; i < s; i++)
           f[i] = true;
       // Usuwanie znanych liczb niebędących pierwszymi.
       f[0] = f[1] = false;
       // Sito.
       int j;
       for (i = 2; i < Math.sqrt(s) + 1; i++) {
           if (f[i]) { // Jeżeli i nie jest wykreślone, wykreślamy jego wielokrotności.
               for (j = 2 * i; j < s; j += i)
                   f[j] = false; // Wielokrotności nie są pierwsze.
           }
       }
       // Ile mamy liczb pierwszych?
       int count = 0;
       for (i = 0; i < s; i++) {
           if (f[i]) count++; // Licznik trafień.
       }
       int[] primes = new int[count];
       // Przeniesienie liczb pierwszych do wyniku.

       for (i = 0, j = 0; i < s; i++) {
           if (f[i]) // Jeżeli pierwsza.
               primes[j++] = i;
       }
       return primes; // Zwracamy liczby pierwsze.
   } else // maxValue < 2
       return new int[0]; // Zwracamy pustą tablicę, jeżeli niewłaściwe dane wejściowe.
}
```

```java
public static int[] generatePrimesSchorter(int maxValue) {
   if (maxValue < 2) return new int[0];

   int s = maxValue + 1;
   boolean[] f = new boolean[s];
   int i;
   for (i = 0; i < s; i++) f[i] = true;
   f[0] = f[1] = false;

   int j;
   for (i = 2; i < Math.sqrt(s) + 1; i++) {
       if (f[i]) for (j = 2 * i; j < s; j += i) f[j] = false;
   }

   int count = 0;
   for (i = 0; i < s; i++) if (f[i]) count++;

   int[] primes = new int[count];

   for (i = 0, j = 0; i < s; i++) if (f[i]) primes[j++] = i;
   return primes;
}
```

```java
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 {
           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 int determineIterationLimit() {
       // Każda wielokrotność w tablicy ma podzielnik 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;
   }
}
```

```java
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 {
            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 int determineIterationLimit() {
        // Każda wielokrotność w tablicy ma podzielnik 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 podzielnik
// 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

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

    turnOnPC();
    typeOnKeyboard(USER_PASSWORD) ;
    typeOnKeyboard("cout<<\"Hello Word!\"<<endl");
    clickMouse(BUILD_RUN_BUTTON) ;
    turnOffPC();

    repeat();
}
```

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

    repeat();
}
```

# 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

```
/*
* Classname
*
* Version info
*
* Copyright notice
*/
```

```
// System.out.println(routerName + ": ~");
```

```
/**
* 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

```
// 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" );
    }
}


// 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" );
    }
}
```

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

      -- VS --

if (employee.isEligibleForFullBenefits())
```

# 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()){
        processPackageIfNessesary(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 używane są przestarzałe funkcje, trzeba je zmieinić
```

# 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
```

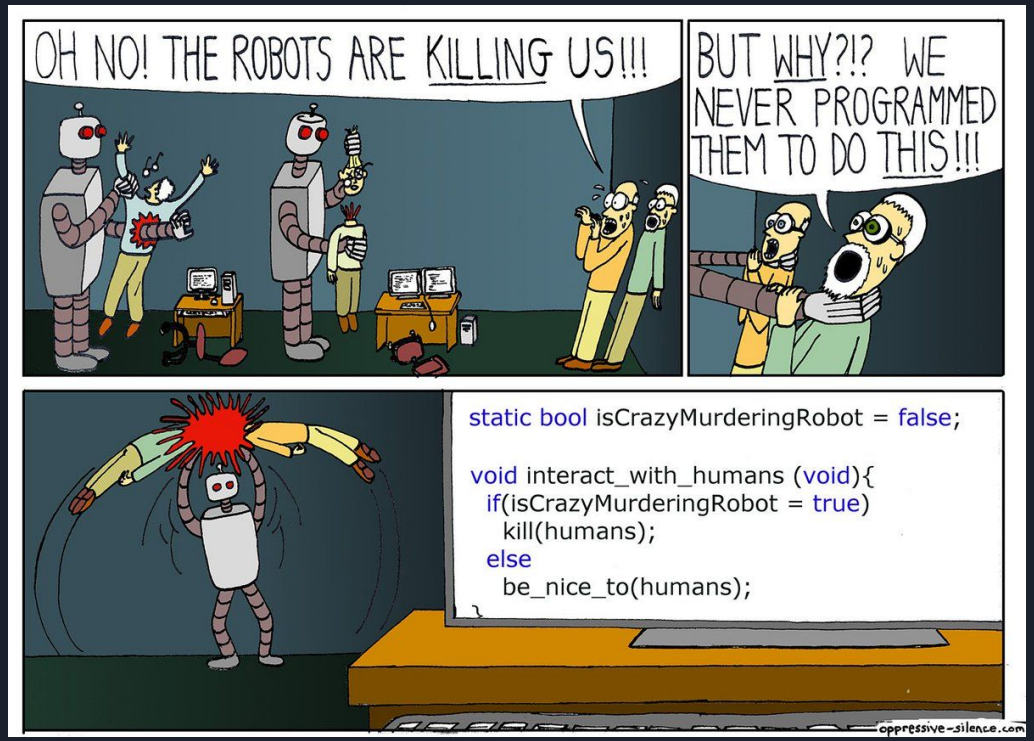
Podkreślenie istotności operacji

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

TODO

# Formatowanie

Czy spacje mają znaczenie?


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);
}
oppressive-silence.com

```
private static int findClosingBracket(String expr){
    int openBracketCounter=0;
    for(int i=0;i<expr.length();i++){
        if(expr.charAt(i)=='(')openBracketCounter++;
        else if(expr.charAt(i)==')'&&openBracketCounter>1)openBracketCounter--;
        else if(expr.charAt(i)==')'&&openBracketCounter==1)return i;
    }
    return -1;
}
```

```
private static int findClosingBracket(String expr) {
    int openBracketCounter = 0;
    for (int i = 0; i < expr.length(); i++) {
        if (expr.charAt(i) == '(') openBracketCounter++;
        else if (expr.charAt(i) == ')' && openBracketCounter > 1) openBracketCounter--;
        else if (expr.charAt(i) == ')' && openBracketCounter == 1) return i;
    }
    return -1;
}
```

```
private static int findClosingBracket(String expr) {
    int openBracketCounter = 0;
    for (int i = 0; i < expr.length(); i++) {
        if (expr.charAt(i) == '(') openBracketCounter++;
        else if (expr.charAt(i) == ')' && openBracketCounter > 1) openBracketCounter--;
        else if (expr.charAt(i) == ')' && openBracketCounter == 1) return i;
    }
    return -1;
}
```

```
private static int findClosingBracket(String expr)
{
    int openBracketCounter = 0;
    for (int i = 0; i < expr.length(); i++)
    {
        if (expr.charAt(i) == '(') openBracketCounter++;
        else if (expr.charAt(i) == ')' && openBracketCounter > 1) openBracketCounter--;
        else if (expr.charAt(i) == ')' && openBracketCounter == 1) return i;
    }
    return -1;
}
```

There are two types of people.



Programmers will know.

# 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
- nazewnictwo zmiennych i funkcji
- komentarze
- zakazane konstrukcje
- ustawienia nawiasów klamrowych
- dokumentacja

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 (AnException 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 (AnException 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;
}
```

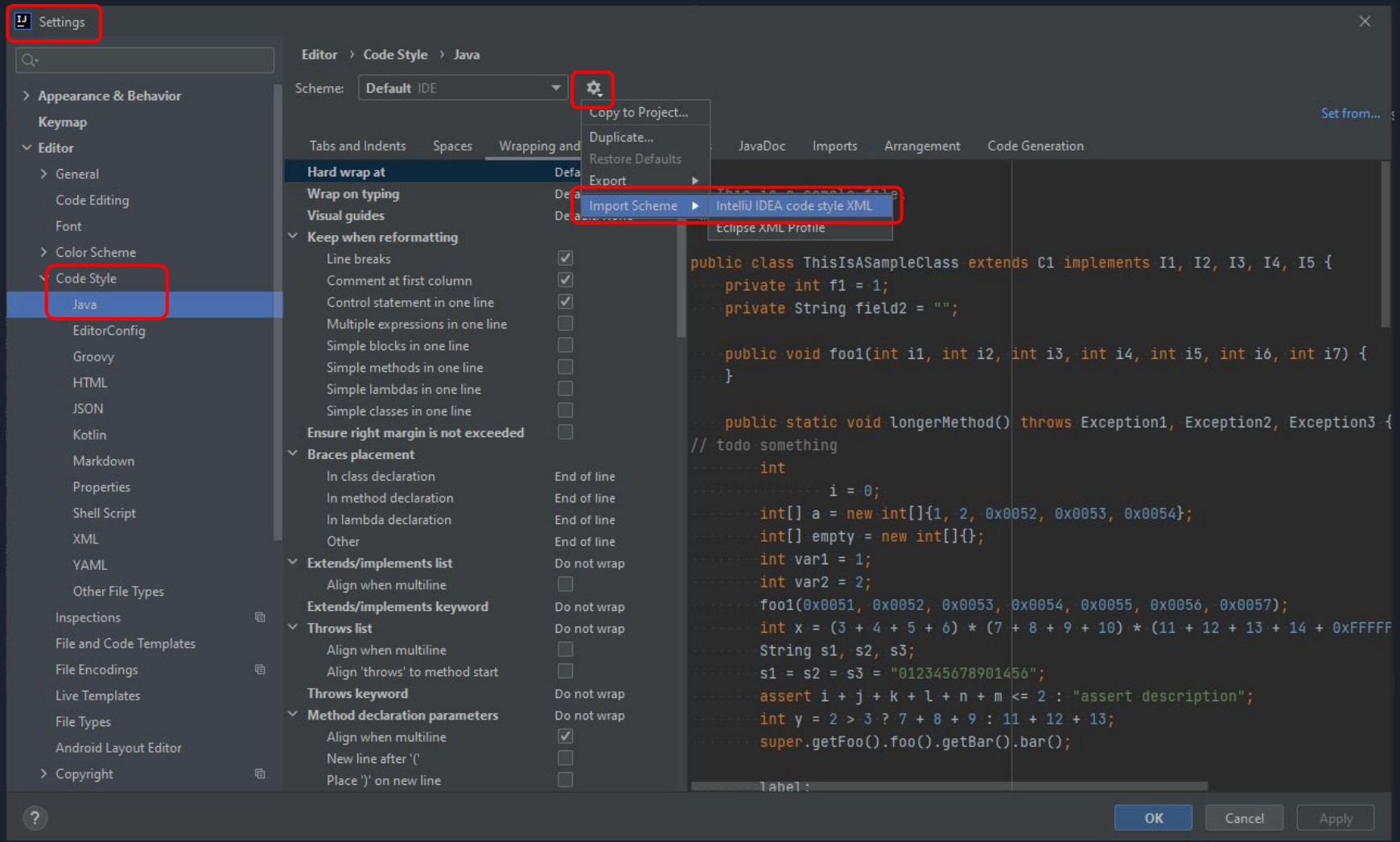
gh-pages styleguide / intellij-java-google-style.xml Go to file ···

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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<code_scheme name="GoogleStyle">
  <option name="OTHER_INDENT_OPTIONS">
    <value>
      <option name="INDENT_SIZE" value="2" />
      <option name="CONTINUATION_INDENT_SIZE" value="4" />
      <option name="TAB_SIZE" value="2" />
      <option name="USE_TAB_CHARACTER" value="false" />
      <option name="SMART_TABS" value="false" />
      <option name="LABEL_INDENT_SIZE" value="0" />
      <option name="LABEL_INDENT_ABSOLUTE" value="false" />
      <option name="USE_RELATIVE_INDENTS" value="false" />
    </value>
  </option>
  <option name="INSERT_INNER_CLASS_IMPORTS" value="true" />
  <option name="CLASS_COUNT_TO_USE_IMPORT_ON_DEMAND" value="999" />
  <option name="NAMES_COUNT_TO_USE_IMPORT_ON_DEMAND" value="999" />
  <option name="PACKAGES_TO_USE_IMPORT_ON_DEMAND">
    <value />
  </option>
  <option name="IMPORT_LAYOUT_TABLE">
    <value>
```


Settings
Editor › Code Style › Java
Scheme: Default IDE
Copy to Project...
Duplicate...
Restore Defaults
Export
Import Scheme
IntelliJ IDEA code style XML
Eclipse XML Profile
Set from...
Tabs and Indents
Spaces
Wrapping and
JavaDoc
Imports
Arrangement
Code Generation
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
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
End of line
Do not wrap
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) {
}
public static void longerMethod() throws Exception1, Exception2, Exception3 {
// todo something
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();
label:
OK
Cancel
Apply

# Przykład

```
public void sendPackageThruPort(Package p)
{
   if(!isOn) return;

   if(p.TTL == p.TTLmax)  //this is first node
   {
       p.log = "From " + parentRouter.getName() ;
       p.source = this.address;
   }

   p.onGoThruPort(getFullName()) ;
   outerSocket.pushPackageToBuff(p) ;
}
```

```
public void sendPackageThruPort(Package p)
{
    if(!isOn) return;

    if(p.TTL == p.TTLmax) //this is first node
    {
        p.log = "From " + parentRouter.getName() ;
        p.source = this.address;
    }

    p.onGoThruPort(getFullName()) ;
    outerSocket.pushPackageToBuff(p) ;
}
```

```
public void sendPackageThruPort(Package p)
{
   if(!isOn) return;

   if(p.isThisFirstNode())
   {
       p.log = "From " + parentRouter.getName() ;
       p.source = this.address;
   }

   p.onGoThruPort(getFullName()) ;
   outerSocket.pushPackageToBuff(p) ;
}
```

```
public void sendPackageThruPort(Package p)
{
    if(!isOn) return;

    if(p.isThisFirstNode())
    {
        p.log = "From " + parentRouter.getName() ;
        p.source = this.address;
    }

    p.onGoThruPort(getFullName()) ;
    outerSocket.pushPackageToBuff(p) ;
}
```

```
public void sendPackageThruPort(Package p)
{
    if(!isOn) return;

    if(p.isThisFirstNode())
    {
        p.log = "From " + parentRouter.getName() ;
        p.source = this.address;
    }

    p.addNameToLog(this.getFullName());
    outerSocket.pushPackageToBuff(p) ;
}
```

```
public void sendPackageThruPort(Package p)
{
    if(!isOn) return;

    if(p.isThisFirstNode())
    {
        p.log = "From " + parentRouter.getName();
        p.source = this.address;
    }

    p.addNameToLog(this.getFullName());
    outerSocket.pushPackageToBuff(p);
}
```

```
public void sendPackageThruPort(Package p)
{
   if(!isOn) return;

   if(p.isThisFirstNode())
       p.source = this.address;

   p.addNameToLog(this.getFullName());
   outerSocket.pushPackageToBuff(p);
}
```

```
public void processAndSend(Package p)
{
    if(!this.isOn)
        return;

    if(p.isThisFirstNode())
         p.setSourceAddress(this.address);

    p.addNameToLog(this.getFullName());

    outerSocket.pushPackageToBuff(p) ;
}
```

# Zadania

# 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://ucgosu.pl/2020/04/style-guide-i-coding-standard-czy-to-juz-jakosc/

# Bibliografia

Style Guide:


- https://google.github.io/styleguide/javaguide.html#s3-source-file-structure
- https://github.com/google/styleguide/blob/gh-pages/intellij-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

Przykłady:


- https://www.programiz.com/java-programming/examples/prime-number
- https://github.com/leonardolemie/clean-code-java#table-of-contents
- https://github.com/piotrjwegrzyn
- https://github.com/thekristopl