KOMENTOWANIE KODU, TWORZENIE DOKUMENACJI ZA POMOCĄ NARZĘDZIA

Marek Michalski

WSIZ Copernicus, 13.04.2019 r.

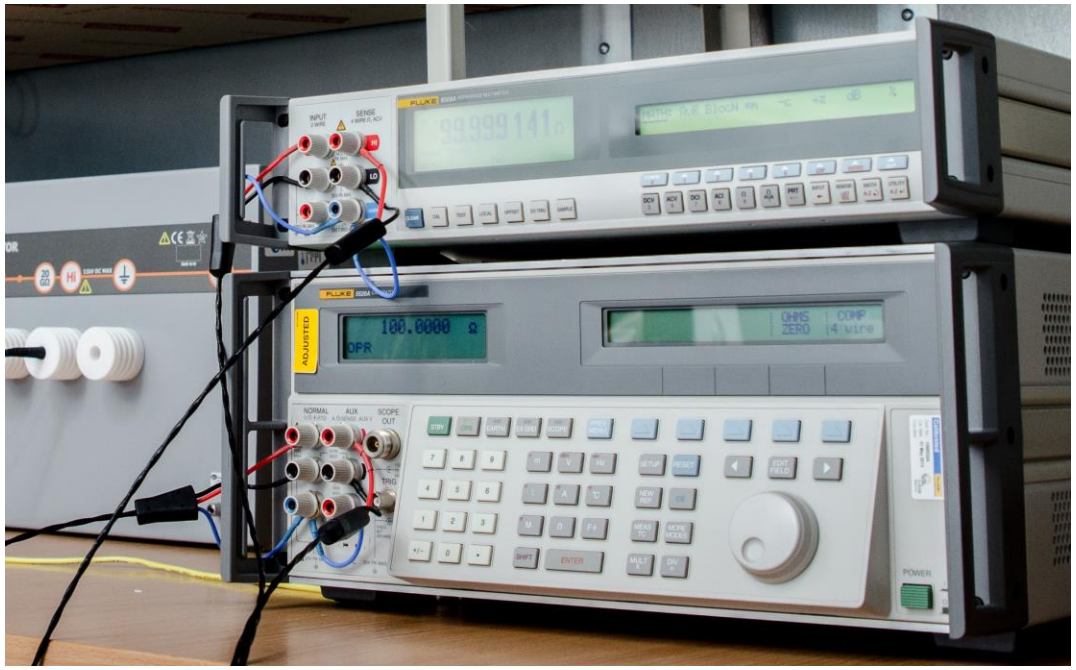
MAREK MICHALSKI

- absolwent Wydziału Elektroniki Politechniki Wrocławskiej
- studia podyplomowe: Tworzenie Oprogramowania w Technologii .NET

  Testowanie Oprogramowania

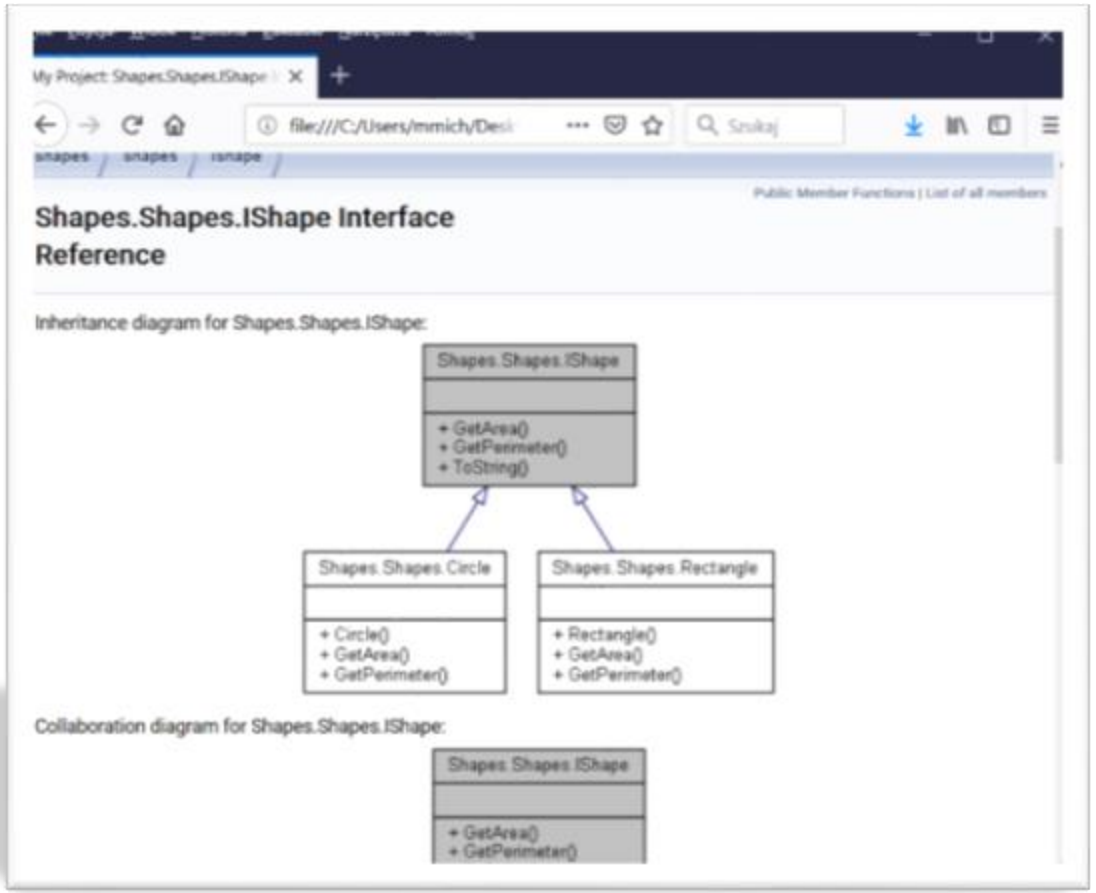
marek_michalski@o2.pl

@mmichalski_pl



CEL PREZENTACJI

```
/// <summary>
/// Print info about error (red color).
/// </summary>
/// <param name="s">Error description.</param>
private static void PrintError(string s)
{
    var c = Console.ForegroundColor;
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine("ERROR: " + s);
    Console.ForegroundColor = c;
}
```



PO CO DOKUMENTOWAĆ KOD?

- dla zespołu (kod tworzy wiele osób jednocześnie)
- dla siebie (łatwiejszy powrót do projektu po czasie)
- dla klienta

Do form dokumentacji zalicza się:

- komentarze
- testy
- diagramy klas
- grafy wywołań
- ...

KOMENTOWANIE KODU

Jakie role pełnią komentarze w kodzie?

Funkcja 1: Wyjaśnia co robi dany fragment kodu

```
//checks if the year is leap
if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
{
    DoSomething();
};
```

```
if (IsLeapYear(year))
{
    DoSomething();
};
```

```
static bool IsLeapYear(int year)
{
    return (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0));
}
```

Zły (nieczytelny) kod należy poprawić zamiast komentować.

Funkcja 2: Wyjaśnia intencję, niestandardowe działanie.

```
static bool ValidateHexNumber(string hexNumber)
{
    //only 4 or 8 digits hex are correct (format #xxxx or #xxxxxxxx)
    Regex regHex = new Regex(@"^#?([a-f0-9]{4}|[a-f0-9]{8})$");
    return regHex.IsMatch(hexNumber);
}
```

Przy wyrażeniach regularnych (Regex) często stosuje się komentarze.

Funkcja 3: Ostrzeżenie.

```
// When I wrote this, only God and I understood what I was doing
// Now, God only knows


// I am not sure if we need this, but too scared to delete.


// 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
```

Informacje o długim czasie wykonania, modyfikacjach w bazie danych, zastąpieniu metody inną itp.

Funkcja 4: Zakomentowywanie linii kodu.

```
//if ((b == int.MaxValue && a > 0) || (b == int.MaxValue && b > 0))
if ((a == int.MaxValue && b > 0) || (b == int.MaxValue && a > 0))
    throw new System.OverflowException();
```

Praktyka niedopuszczalna w kodzie produkcyjnym.

Nie wiadomo do czego służy zakomentowany kawałek kodu (poprawka błędu?, testy?).

Programiści będą się obawiać go usunąć.

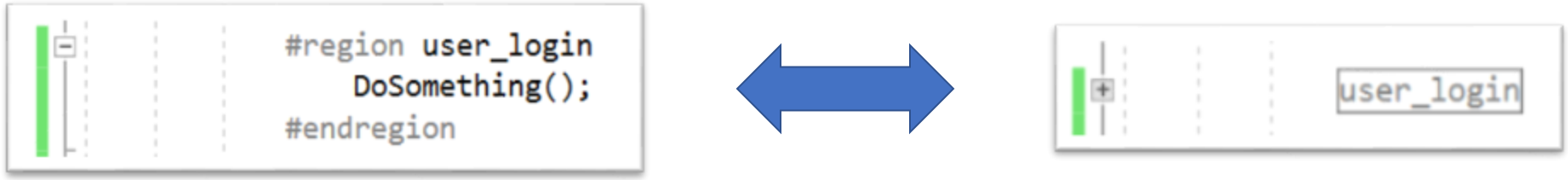
Funkcja 5: Wydzielenie bloku kodu.

```
// ************** USER LOGIN **************

    DoSomething();

// ************** END USER LOGIN **************
```

Visual Studio oferuje możliwość definiowania regionów.



Funkcja 6: Umieszczanie informacji prawnych.

```
// Copyright(C) 2019 by Company, Inc.All rights reserved.
// Released under the terms of the GNU General Public License version 2 or later.
```

Jest ok o ile informacja nie jest za długa.

Funkcja 7: Historia zmian.

```
//21.01.1998 First version
//23.01.1998 Add connection to database
//24.01.1998 Fix bug in print function
//25.01.1998 Add new serwer IP
```

Można spotkać w starszych projektach (brak systemów kontroli wersji).

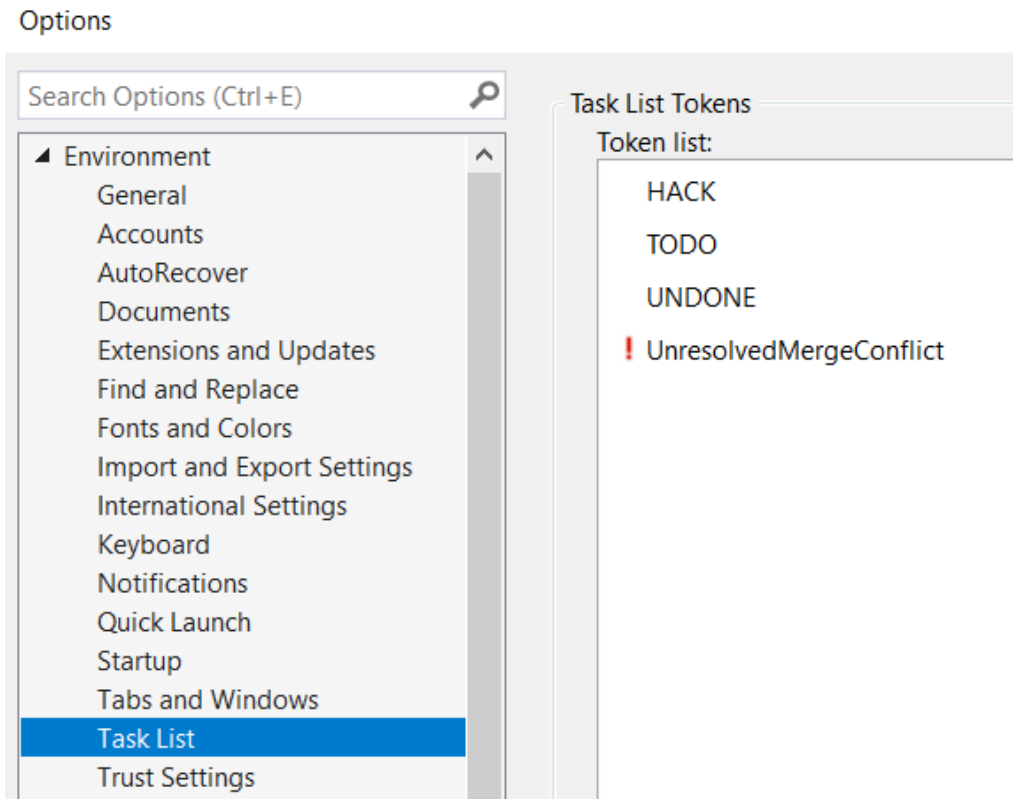
Funkcja 8: Oznaczanie końców bloków kodu.

```
            }
        } //foreach
    } //while
} //if
```

Jeżeli blok jest za długi należy go skrócić.

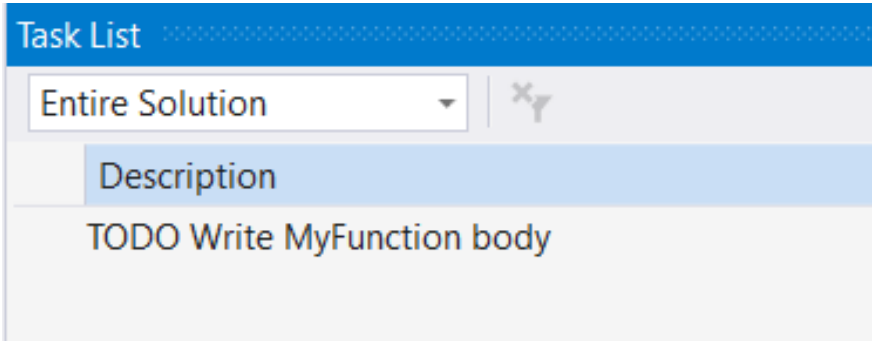
Funkcja 9: Tokeny listy zadań w Visual Studio.

Tools → Options




```
static void MyFunction()
{
    //TODO Write MyFunction body
}
```

View → Task list

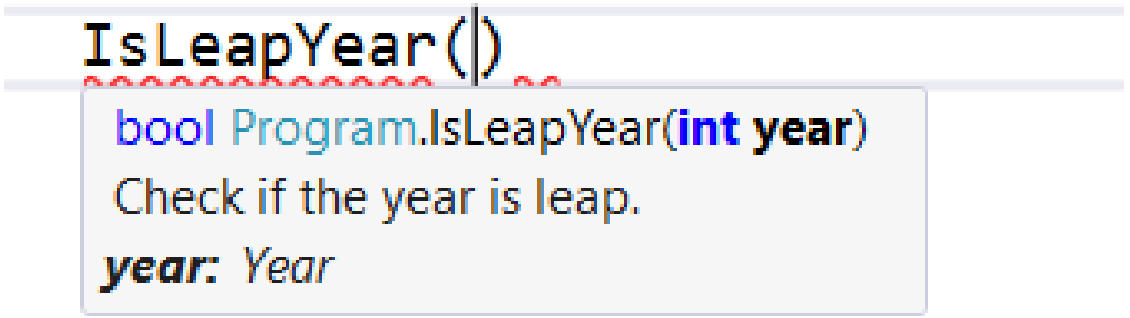


Funkcja 10: Komentarze dokumentujące.

Przykład udokumentowania funkcji.

```
/// <summary>
/// Check if the year is leap.
/// </summary>
/// <param name="year">Year</param>
/// <returns>true is year is leap, false if not</returns>
static bool IsLeapYear(int year)
{
    return (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0));
}
```

Integracja z IntelliSense



Lista wspieranych tagów.

CO BĘDZIE POTRZEBNE?

http://www.doxygen.nl/

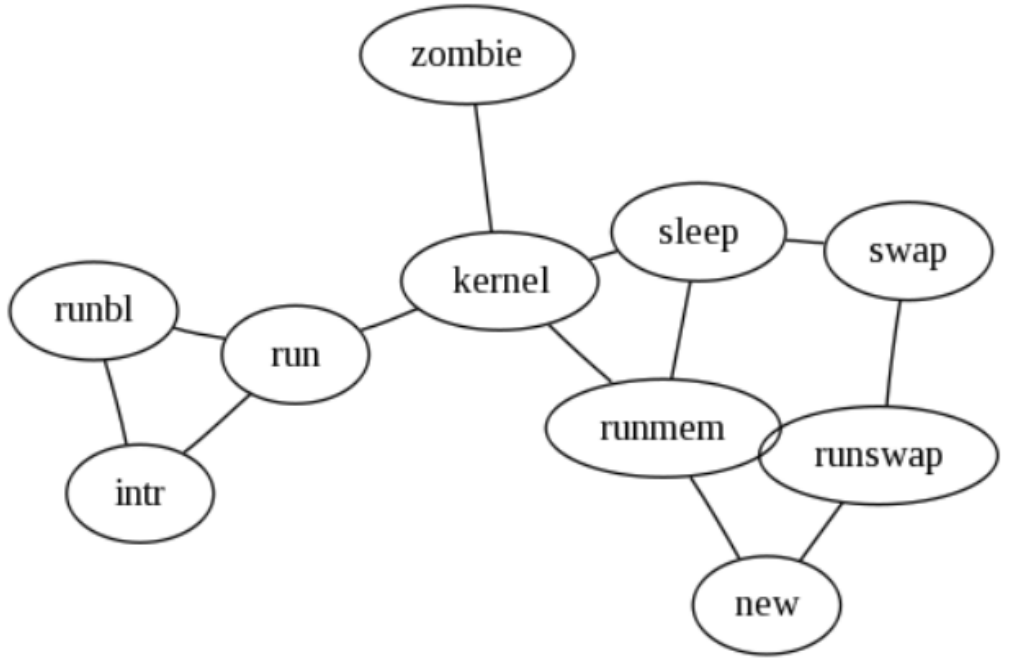
- Automatycznie tworzy dokumentację kodu na podstawie analizy plików źródłowych projektu.
- Jest darmowy.
- Dostępny na wiele OS (m.in. Windows, Linux).
- Wspiera wiele języków (m.in. C#, Java, C++, VHDL).
- Kilka formatów wyjściowych (m.in. HTML, LATEX).
- Działa autonomicznie (nie potrzeba kompilatora).

CO BĘDZIE POTRZEBNE?

https://www.graphviz.org/

Generuje grafy i diagramy (pliki graficzne) na podstawie opisu za pomocą skryptu.

```
graph G {
        run -- intr;
        intr -- runbl;
        runbl -- run;
        run -- kernel;
        kernel -- zombie;
        kernel -- sleep;
        kernel -- runmem;
        sleep -- swap;
        swap -- runswap;
        runswap -- new;
        runswap -- runmem;
        new -- runmem;
        sleep -- runmem;
}
```



OPCJONALNIE

Kompilator LATEX-a

https://miktex.org/

```
1   \documentclass[11pt,a4paper]{article}
2   \usepackage[polish]{babel}
3   \usepackage[utf8]{inputenc}
4
5   \title{Przykład dokumentu}
6   \author{Marek Michalski}
7   \date{\today}
8
9   \begin{document}
10  \maketitle
11
12  \tableofcontents
13
14  \section{Rozdział}
15      Treść rozdziału.
16  \section{Kolejny rozdział}
17      Treść kolejnego rozdziału.
18  \subsection{Podrozdział}
19      Treść podrozdziału.
20
21  \end{document}
```

Przykład dokumentu

Marek Michalski

30 czerwca 2018

Spis treści

1 Rozdział

Treść rozdziału.

2 Kolejny rozdział

Treść kolejnego rozdziału.

2.1 Podrozdział

Treść podrozdziału.

PRZYKŁAD

Dokumentowanie kodu Java (tagi komentujące).

```java
/**
 * Retrieves the identifier of the doxygen wrapper.
 *
 * @return    a string containing the doxygen wrapper identifier
 */
public abstract String getIdentifier();


/**
 * Retrieve the diretory of the specified file.
 *
 * @param    file    The file for which the directory must be retrieved.
 *
 * @return    The path of the containing directory.
 */
private static IPath getDir( IFile file ) {
    return file.getLocation().makeAbsolute().removeLastSegments( 1 );
}
```

https://www.programcreek.com/java-api-examples/index.php?source_dir=eclox-master/eclox.core/src/eclox/core/doxygen/Doxygen.java

KOMENTOWANIE KODU
PODSUMOWANIE

- Stosujemy się do określonych w firmie standardów.
- Komentarze piszemy wyłącznie po angielsku.
- Zamiast komentowania złego kodu popraw go.
- Komentarze powinny być zwięzłe i rzetelne.
- Komentarze piszemy łącznie z kodem.
- Komentarze należy aktualizować.
- Nie wrzucamy do repozytorium programu z zakomentowanymi fragmentami kodu.

TWORZENIE DOKUMENTACJI NIE JEST PROSTE:

- brak czasu
- przekonanie, że doby kod dokumentuje się sam,
  a dobry programista nie komentuje kodu
- przekonanie, że dokumentacja nie wnosi wartości biznesowej
- Agile manifesto:
  Working software over Comprehensive documentation
- tworzenie dokumentacji jest postrzegane jako nudne

DZIĘKUJĘ ZA UWAGĘ