



# Inżynieria Oprogramowania

## 5. Diagram sekwencji i inżynieria odwrotna

Opracował: Maciej Penar

## Spis treści

1. Zanim zaczniemy .....	3
2. (6 pkt) Diagramy sekwencji .....	3
Zadania główne .....	3
Kod dla CAS Proxy (Jakby to miało pomóc) .....	4
Zadanie dodatkowe.....	5
3. (6 pkt) Inżynieria Odwrotna - Obfuskatory .....	6
4. (*1 pkt) Inżynieria Odwrotna – 5 nieczystych zagrań .....	7

## 1. Zanim zaczniemy

Zrelaksować się i przyswoić sobie teorię dot. diagramów sekwencji.

Materiały do PU:

- Materiały z Lucidchart: <https://www.lucidchart.com/pages/uml-sequence-diagram>
- Materiały z IBM: <https://www.ibm.com/developerworks/rational/library/3101.html>

Oprogramowanie:

- Lucidchart (oprogramowanie w modelu SasS do modelowania)
- Enterprise Architect (<http://www.sparxsystems.com/products/ea/>)
- Visual Paradigm (<https://www.visual-paradigm.com/download/community.jsp>)
- Visio (<https://products.office.com/pl-pl/visio/flowchart-software> )

Do obfuskatorów:

- **Confuser**: [link](#)
- **ConfuserEx**: [link](#)
- **ILSpy**: [link](#)

## 2. (6 pkt) Diagramy sekwencji

### ZADANIA GŁÓWNE

1. (0 pkt) Zapoznać się z diagramem sekwencji dla CAS:  
<https://apereo.github.io/cas/4.2.x/protocol/CAS-Protocol.html>
2. (4 pkt) Narysować diagram sekwencji dla procesu kupna przedmiotu z systemu aukcyjnego Allegro Lento, ograniczenia to:
  - a. Aktorzy na scenie to: Kupujący, Allegro Lento, Sprzedający, Bank, Poczta
  - b. Kupujący otrzymuje potwierdzenie kupna
  - c. Kupujący otrzymuje potwierdzenie zapłaty
  - d. Kupujący wystawia komentarz gdy dostanie przedmiot
3. (2 pkt) Narysować diagram sekwencji dla przepływu logowania przez CAS wykorzystanego w projekcie (kod poniżej)

## KOD DLA CAS PROXY (JAKBY TO MIAŁO POMÓC)

```
<html>

    <head>

        <title>CASProxy</title>

    </head>

    <body>

        <?php

            $key = ...;
            $validDate = ...;
            $now = date("Y-m-d H:i:s");

            if($_GET['key'] == $key && strcasecmp($now, $validDate) < 0){

                $curl = curl_init();
                curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "POST");
                curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
                curl_setopt($curl, CURLOPT_URL, .... . $_GET['ticket']);
                $result = curl_exec($curl);
                curl_close($curl);

                print_r("<div id=\"ticket\">");
                print_r(base64_encode($result));
                print_r("</div>");

            }else{

                print_r("<div id=\"ticket\">");
                print_r('Access Forbidden');
                print_r("</div>");

            }

        ?>

        <script type="text/javascript">

            var url_string = window.location.href;
            var url = new URL(url_string);
            var redirectUrl = url.searchParams.get("redirect");
            window.location.replace(redirectUrl + "?response=" +
document.getElementById("ticket").textContent);

        </script>

    </body>

</html>
```

## ZADANIE DODATKOWE

4. (\* 1 pkt) W pierwszym prototypie CAS Proxy kod był następujący:

```
<html>

  <script type="text/javascript">

    var url_string = window.location.href;
    var url = new URL(url_string);
    var redirectUrl = url.searchParams.get("redirect");
    var ticket = url.searchParams.get("ticket");
    function httpGet(theUrl)
    {
        var xmlHttp = new XMLHttpRequest();
        xmlHttp.open( "GET", theUrl, false );
        xmlHttp.send( null );
        return xmlHttp.responseText;
    }
    function b64EncodeUnicode(str) { return ...; }
    var response = httpGet(.... + ticket);
    var based = b64EncodeUnicode(response);
    window.location.replace(redirectUrl + "?response=" + based);

  </script>

</html>
```

Narysować diagram dla przepływu dla prototypu CAS Proxy. Omówić wady.

### 3. (6 pkt) Inżynieria Odwrotna - Obfuskatory

Na repo są dwa programy: NonObfuscated.exe i Obfuscated.exe [link](#)

Są spakowane w .zip, zahasłowany: **obfuscate**

Po rozpakowaniu defender/antywirus może krzyczeć że Obfuscated.exe to wirus – dodać wyjątek.

Nie trzeba uruchamiać tych .exe

Oba programy są na jedno kopyto:

```
class Program
{
    private static string Password { get; set; } = "???"

    static void Main(string[] args)
    {
        Console.WriteLine("Hi! Pls enter password:");
        var input = Console.ReadLine();
        Console.WriteLine(Password.Equals(input) ? "???" : "???");
        Console.ReadLine();
    }
}
```

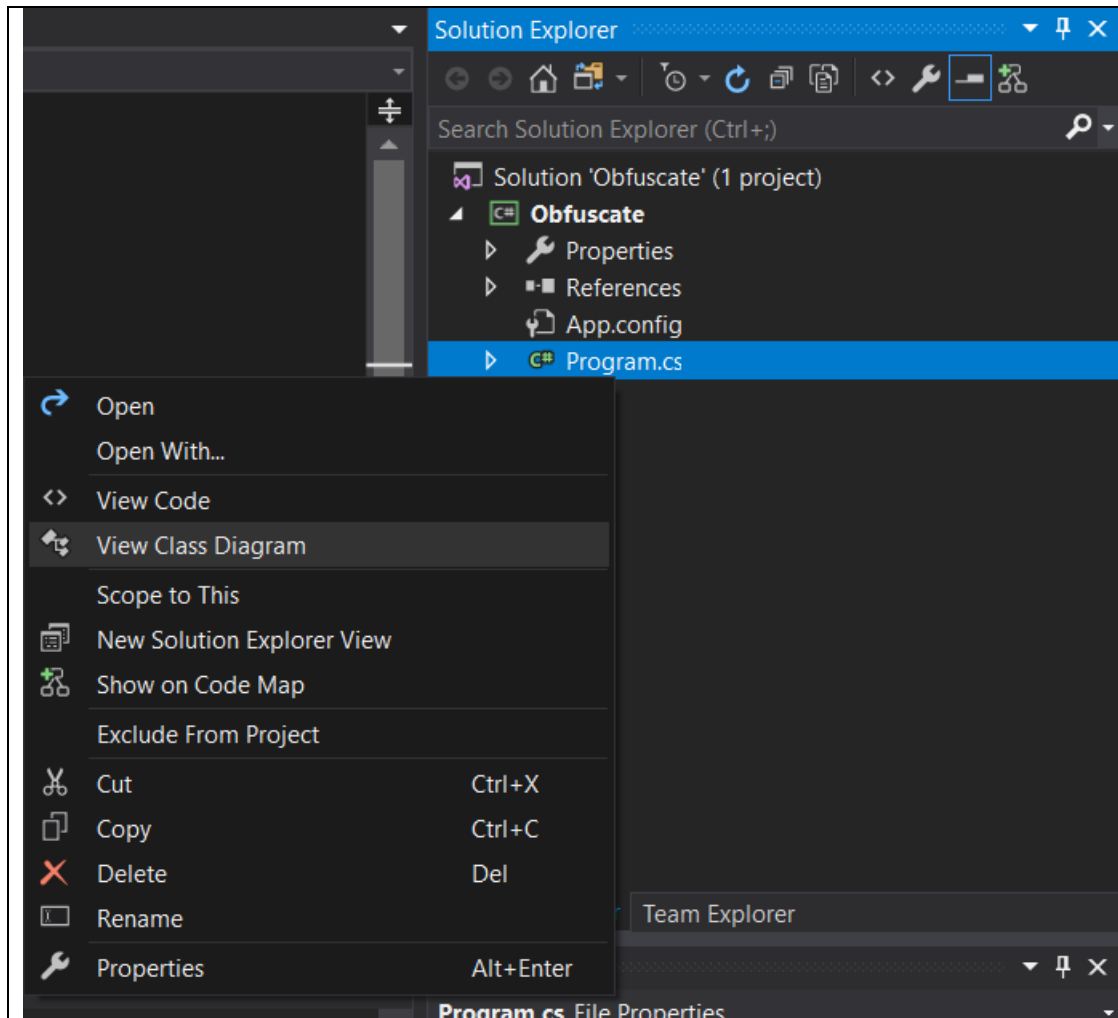
Wykonać polecenia (użyć ILSpy):

- (3 pkt) Wykonać + odpowiedzieć (sobie) na pytania:
  - Znaleźć hasło i komunikaty wyjściowe dla NonObfuscated.exe
  - Czym się różnią oba programy?
  - Dlaczego antywirus/defender podnieśli wrzask (powinni)?
- \*(? Pkt + achievement „Silna Wola”) Znaleźć hasło i komunikaty wyjściowe dla Obfuscated.exe
- (3 pkt) Wykonać obfuskację prostego kodu za pomocą **ConfuserEx**:
  - Tak żeby podmienić nazwy pól
  - Dowolną metodą

## 4. (\*1 pkt) Inżynieria Odwrotna – 5 nieczystych zagrań

Przeczytać i zapamiętać 5 rzeczy:

1. Można wykonać rysowanie diagramu UML na podstawie kodu. W Visual Studio Enterprise/Ultimate 2017 ta opcja jest dostępna tu:



2. Można też generować diagramy sekwencji
3. Istnieje wiele narzędzi CASE (Computer-Aided Software Engineering), w szczególności prym wiodą narzędzia dla Javy. Najlepiej jak narzędzia do Inżynierii Oprogramowania (w tym odwrotnej) są wbudowane w IDE.
4. **Na podstawie komentarzy w kodzie można generować dokumentację (najczęściej strony w HTML'u)**
5. Istnieje koncepcja „samodokumentującego się kodu”  
Jak ktoś się zapyta czemu nie dokumentujecie kodu, odpowiadacie „mój kod jest samodokumentujący”.  
**Nigdy taki nie był.**  
**Nie jest.**  
**Nigdy nie będzie.**  
Tydzień nie będziecie patrzeć na kod.  
Wróćcie.  
„Kto to napisał?” zapytacie.