



# Hands On – OOP & CSV



# HELLO

Michał Sosnowski

Senior Developer at OKE Poland





# Czego się dzisiaj dowiemy?

- OOP: powiązania między klasami
- Praca z plikami CSV
  - Odczyt
  - Zapis
- Rozwiązywanie konfliktów

# OOP

## Powiązania między klasami



# Modyfikatory dostępu

Kiedy używać?

- **Public** – kiedy chcemy, używać danej metody/pola w innych klasach.
- **Private** – kiedy metoda/pole, jest wykorzystywane tylko wewnątrz obecnej klasy (np. Metoda jest wywołane wewnątrz metody publicznej, ale nigdzie indziej).
- **Protected** – podobnie jak private, ale dodatkowo po danej klasie dziedziczy inna klasa, która również potrzebuje metody/pola.
- **Internal** – kiedy tworzymy moduł stanowiący pewną całość, która będzie wykorzystywana przez inny zespół. Chcemy, żeby klasy w tym module (projekcie) mogły korzystać z metody/pola, ale jednocześnie nie chcemy, żeby użytkownicy modułu nie mieli do nich dostępu.

Więcej informacji: <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/access-modifiers>

- Używamy, kiedy chcemy ustawić wartość pola tylko raz, przy tworzeniu obiektu:
  - Np. w Controllerze tworzymy obiekt XyzService. Chcemy korzystać z jego metod, ale nie chcemy go przypadkiem nadpisać nową instancją.

Więcej informacji: <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/readonly>



- Kiedy dane pole/metoda są niezależne od instancji obiektu.
- Możemy wykorzystać, gdy chcemy stworzyć klasę ze zbiorem prostych metod (Helperów)
- Możemy wykorzystać kiedy chcemy mieć wspólne pole dla wszystkich egzemplarzy o danym typie (danej klasy).
  - Np. Licznik utworzonych obiektów, listę z datami kiedy dane obiekty zostały utworzone)
  - Np. Wartość jest wspólna dla całej aplikacji (jak statyczna lista, którą wykorzystaliśmy na poprzednich zajęciach z ASP.NET Core MVC)



# Kiedy używać dziedziczenia?

- Kiedy kilka naszych klas korzysta z podobnych metod/pól.
  - Np. Mamy aplikację webową w MVC, która ma kilka modeli ze wspólnym zbiorem pól
- Kiedy kilka klas stanowi szczególne przypadki ogólnego typu.
- Kiedy wymaga tego od nas framework (np. Dziedziczenie po klasie Controller w ASP.NET Core MVC)
- Nie szukajmy dziedziczenia na siłę! 😊





# Kiedy używać klas abstrakcyjnych?

- Kiedy mamy klasę bazową, która nie może działać samodzielnie.
- Np. Implementuje część logiki poza jedną metodą, która będzie różna dla każdej z klas dziedziczących.
- Np. Implementuje identyczną logikę, ale każda z klas będzie posiadać inne wartości pól.

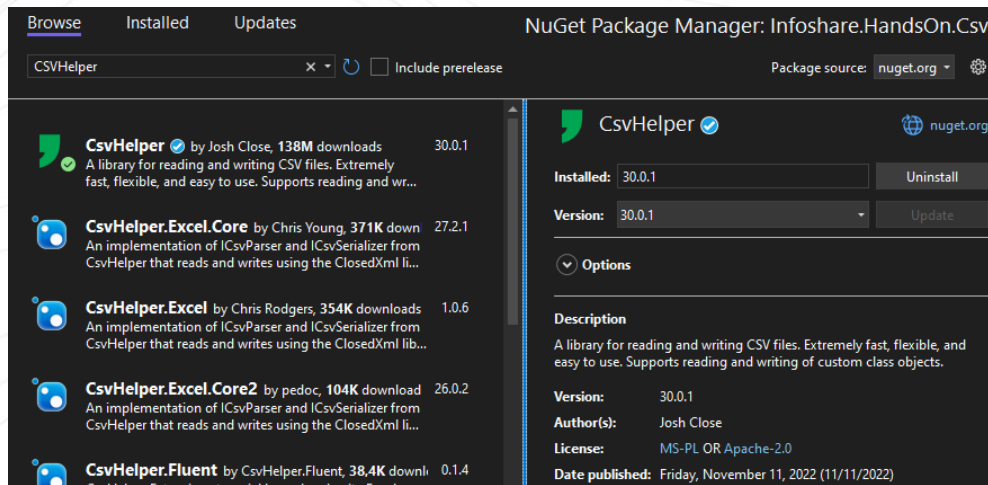


# Kiedy używać interfejsów?

- Interfejsy dają większą elastyczność kodu
- Odraczamy moment decyzji, jakiego typu obiekt ma zostać użyty
- Pozwala nam przekazywać do metod obiekty różnego typu, które jednak posiadają wspólny zbiór metod.
  - Np. Przydaje się to w testowaniu. Do testowanej metody możemy przekazać obiekt, który łączy się z prawdziwą bazą danych, lub obiekt który tylko to symuluje na potrzeby testów.
- Przy kolekcjach, gdy interesuje nas tylko określony zbiór metod (np. GetEnumerator() do iterowania z pomocą pętli foreach), a nie to jakiego typu jest faktycznie kolekcja. Wtedy przykładowo nasza metoda obsłuży zarówno List, Array, HashSet itd. do wyświetlenia wszystkich zawartych w nich elementów.

# Praca z plikami CSV

Powszechnie używana biblioteka, którą należy pobrać za pomocą Nugeta



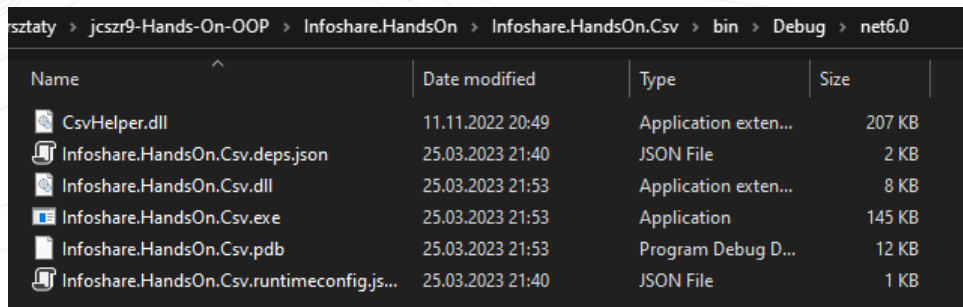
Przydatne linki:

- <https://joshclose.github.io/CsvHelper/getting-started/>
- <https://joshclose.github.io/CsvHelper/examples/>



# Ścieżka względna

Punktem wyjściowym dla ścieżki względnej jest folder z plikiem wykonawczym exe.



Name	Date modified	Type	Size
CsvHelper.dll	11.11.2022 20:49	Application exten...	207 KB
Infoshare.HandsOn.Csv.deps.json	25.03.2023 21:40	JSON File	2 KB
Infoshare.HandsOn.Csv.dll	25.03.2023 21:53	Application exten...	8 KB
Infoshare.HandsOn.Csv.exe	25.03.2023 21:53	Application	145 KB
Infoshare.HandsOn.Csv.pdb	25.03.2023 21:53	Program Debug D...	12 KB
Infoshare.HandsOn.Csv.runtimeconfig.js...	25.03.2023 21:40	JSON File	1 KB

Aby cofnąć się w strukturze folderów do głównego katalogu projektu trzeba użyć następującej ścieżki względnej:

../\..\..\

```
using (var reader = new StreamReader("path\\to\\file.csv"))  
using (var csv = new CsvReader(reader, CultureInfo.InvariantCulture))  
{  
    var records = csv.GetRecords<Foo>();  
}
```

```
using (var writer = new StreamWriter("path\\to\\file.csv"))  
using (var csv = new CsvWriter(writer, CultureInfo.InvariantCulture))  
{  
    csv.WriteRecords(records);  
}
```



```
[Delimiter(",")]
[CultureInfo("")] // Set CultureInfo to InvariantCulture
public class Foo
{
    [Name("Identifier")]
    public int Id { get; set; }

    [Index(1)]
    public string Name { get; set; }

    [BooleanTrueValues("yes")]
    [BooleanFalseValues("no")]
    public bool IsBool { get; set; }

    [Constant("bar")]
    public string Constant { get; set; }

    [Optional]
    public string Optional { get; set; }

    [Ignore]
    public string Ignored { get; set; }
}
```

# Dzięki za uwagę