Zajęcia 1. Stworzenie projektu działającego w trybie wiersza poleceń

1.1. Wstęp

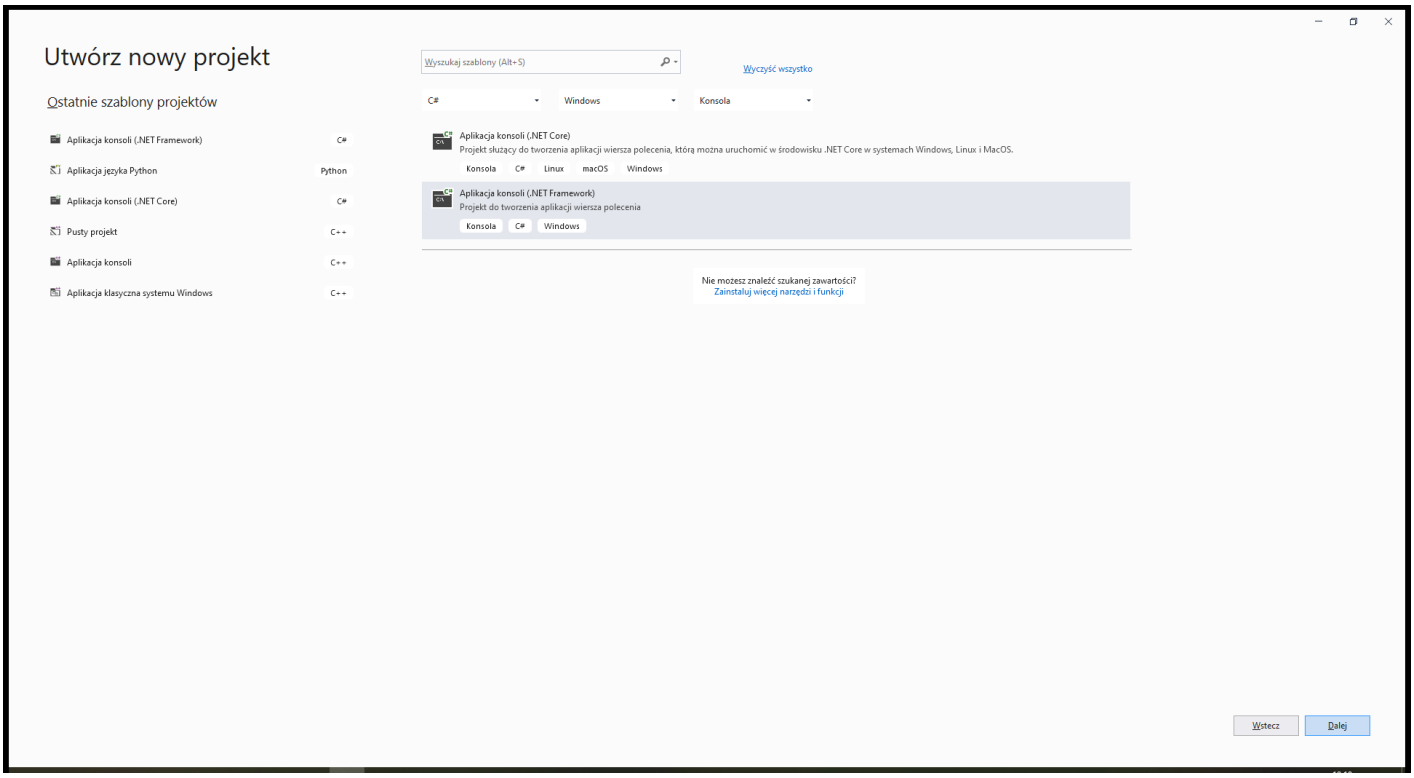
Celem ćwiczenia jest stworzenie projektu typu **Aplikacja konsoli** działającego w ramach platformy **Windows**. W skład tworzonego szablonu projektu wchodzić będą klasy, które stanowić będą typy konieczne do definiowania (w ujęciu programistycznym) pracowników, a mianowicie **`Data`** oraz **`Adres`**. Dzięki temu możliwe będzie tworzenie obiektów typu **`Pracownik`** i manipulowanie ich danymi reprezentowanymi przez datę urodzenia oraz adres zamieszkania. Projekt będzie również zawierał klasę o nazwie **`Program`** z metodą główną **`Main`**. W metodzie tej konieczne będzie zaprezentowanie sposobu tworzenia obiektów klas **`Data`**, **`Adres`** oraz **`Pracownik`**.

1.2. Stworzenie projektu typu Aplikacja konsoli

Aby w środowisku **Visual Studio** utworzyć nowy projekt działający w trybie wiersza poleceń, po uruchomieniu środowiska należy kliknąć na kafelek **Utwórz nowy projekt**. W kolejnym kroku konieczne jest wybranie:

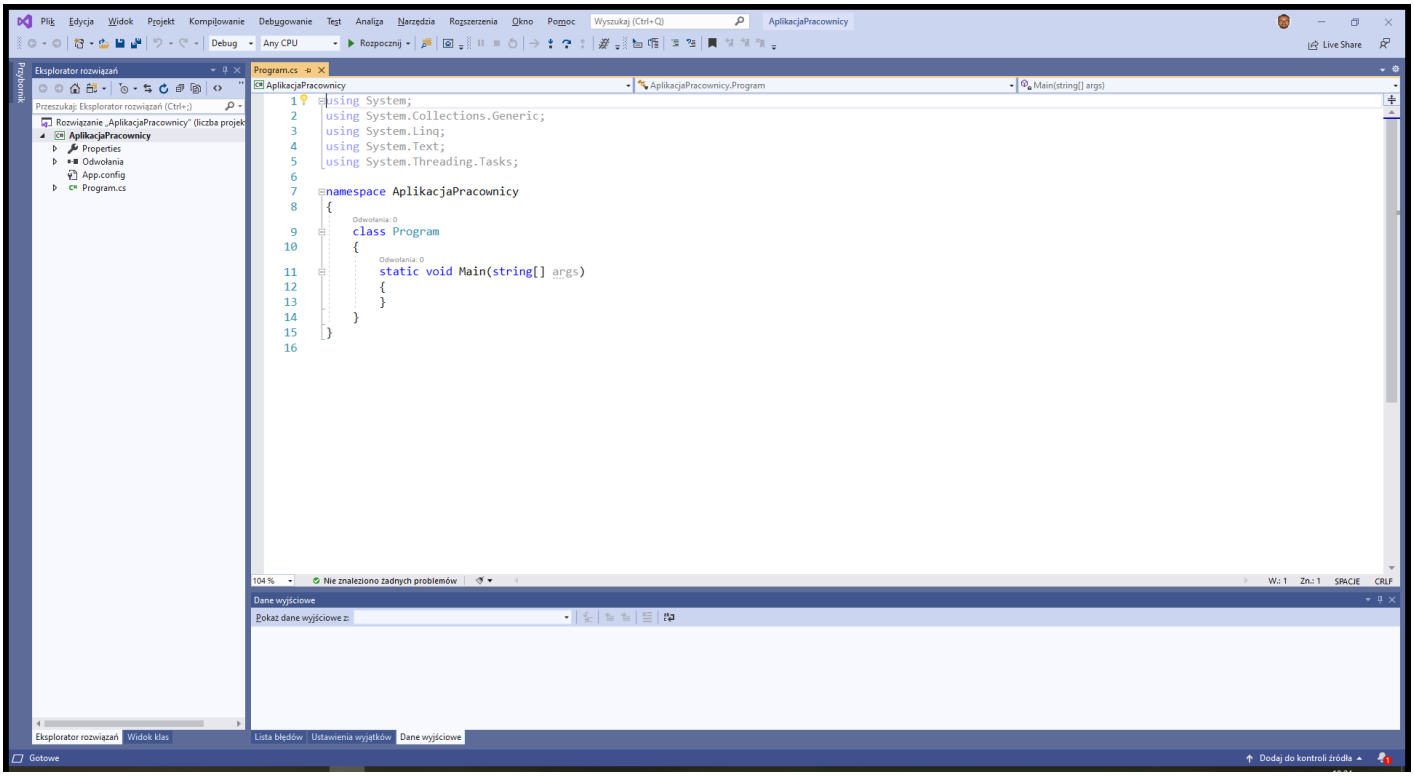
- w menu rozwijanym **Wszystkie języki**: **C#**,
- w menu rozwijanym **Wszystkie platformy**: **Windows**,
- w menu rozwijanym **Wszystkie typy projektów**: **Konsola**.

Po zawężeniu możliwych opcji wystarczy kliknąć na kafelek **Aplikacja konsoli (.NET Framework)** i wcisnąć przycisk **Dalej**. Rysunek 1.1 przedstawia okno tworzenia projektu z poprawnie wybranym typem.



Rysunek 1.1. Tworzenie projektu typu **Aplikacja konsoli** w środowisku Visual Studio.

Ostatni etap to podanie nazwy projektu (np., **AplikacjaPracownicy)** oraz jego lokalizacji. Opcja **Umieść rozwiązanie i projekt w tym samym katalogu** ma pozostać niezaznaczona. Tworzenie prokektu należy zakończyć klknięciem na przycisk **Utwórz**. Taka konfiguracja nowego projektu spowduje wyświetlenie okna śrdowiska programistycznego, które przedstawione jest na Rysunku 1.2.



Rysunek 1.2. Widok nowostworzonego projektu typu **Aplikacja konsoli** wraz z automatycznie wygenerowaną klasą **`Program`**.

Jak można zauważyć całe okno projektu składa się z kilku mniejszych paneli: edycyjnego (tj. do pisania kodu) na środku – z zakładką o nazwie **Program.cs**, podłużnego u dołu – które służy do wyświetlania m.in. listy błędów, ostrzeżeń lub informacji w programie; używane jest ono również do podglądu wartości obiektów, zmiennych podczas procesu *debugowania* (tj. pracy krokowej). Po lewej stronie znajduje się okno przeglądania zawartości całego rozwiązania z zakładkami **Eksplorator rozwiązań** oraz **Widok klas**. Warto zwrócić uwagę, że (przy aktywnej zakładce **Eksplorator rozwiązań**) w drzewie katalogów w tym oknie znajduje się plik **Program.cs**.

W oknie edycyjnym (plik **Program.cs**) umieszczony został automatycznie wygenerowany kod źródłowy. Kod ten składa się z instrukcji dołączających do projektu wybrane przestrzenie nazw (za pomocą dyrektywy **`using`**) oraz z definicji nowej przestrzeni **`AplikacjaPracownicy`**. W przestrzeni tej zdefiniowana jest klasa o nazwie **`Program`**. Wygląd okna projektu może różnić się w zależności od domyślnych ustawień środowiska **Visual Studio**.

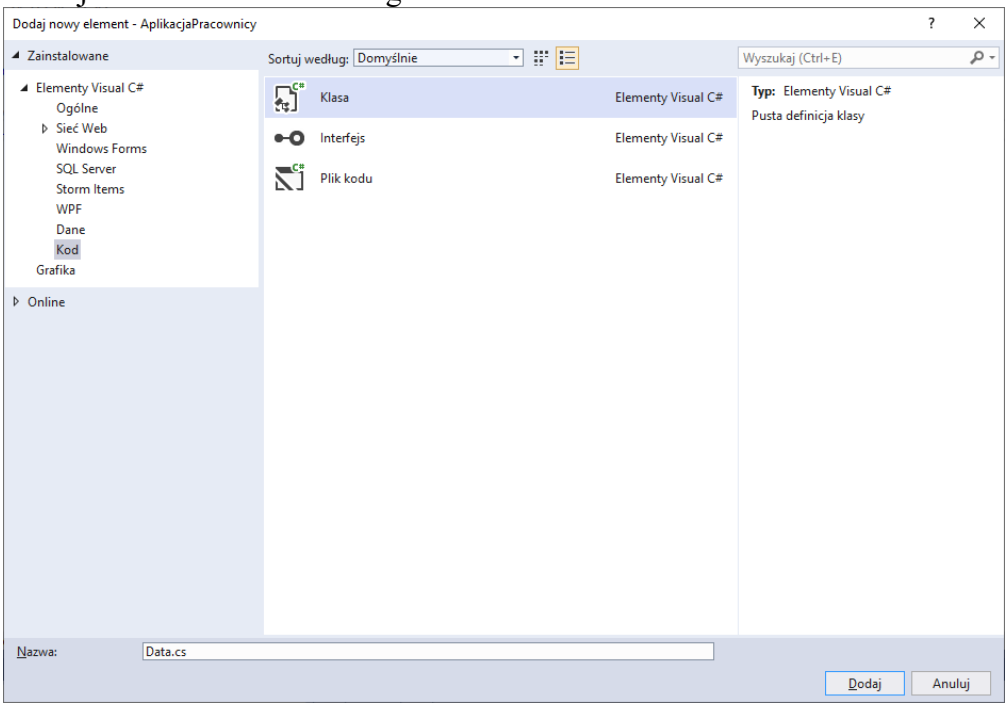
1.3. Definicja klasy `Data`

Istnieją trzy możliwości dodania nowej klasy do istniejącego już projektu. Można to dokonać poprzez:

- wybranie opcji **Dodaj klasę…** z menu **Projekt**,
- kliknięcie przycisku **Dodaj nowy element** na pasku przycisków (lub w menu **Projekt**),
- kliknięcie prawym przyciskiem myszki na nazwie projektu **AplikacjaPracownicy** w oknie przeglądania zawartości rozwiązania (zarówno przy aktywnej zakładce **Eksplorator rozwiązań** jak i **Widok klas**) oraz wybranie opcji **Klasa** z menu **Dodaj**.

Skorzystanie z dowolnej z wyżej wymienionych opcji dodawania klasy spowoduje, że na ekranie pojawi się okno dialogowe umożliwiające dodawanie składnika (klasa „widziana" jest w środowisku Visual Studio jako składnik, ponieważ reprezentuje kod źródłowy i jest zdefiniowana w odrębnym pliku). Aby zawężyć możliwość wyboru dodawanego składnika, wygodnie jest zaznaczyć opcję **Kod** w panelu **Elementy Visual C#**, a następnie **Klasa** w oknie wyboru. Teraz wystarczy w polu **Nazwa** podać nazwę klasy **Data.cs** a następnie

kliknąć przycisk **Dodaj**. Do projektu w przestrzeni **`AplikacjaPracownicy`** dodana zostanie klasa **`Data`**. Rysunek 1.3 prezentuje omawiane okno dialogowe.



Rysunek 1.3. Dodawanie klasy **`Data`** do projektu.

W klasie **`Data`** należy zdefiniować następujące składowe:

- Prywatne pola:
  - **`int dzien`**,
  - **`string miesiac`**,
  - **`int rok`**,
  - **`static readonly string[] miesiace`**, którą należy wypełnić łańcuchami nazw poszczególnych miesięcy.
- Publiczne właściwości:
  - **`Dzien`** zwracającą i ustawiającą wartość pola **`dzien`**,
  - **`Miesiac`** zwracającą i ustawiającą wartość pola **`miesiac`**,
  - **`Rok`** zwracającą i ustawiającą wartość pola **`rok`**.
- Publiczne metody:
  - bezargumentowy konstruktor domyślny.
  - konstruktor inicjalizujący wszystkie pola składowe na podstawie odpowiednio dobranych argumentów: **`Data(int dzien, string miesiac, int rok)`**
  - konstruktor kopiujący inicjalizujący wszystkie pola składowe na podstawie argumentu wzorcowego: **`Data(Data wzor)`**.
  - **`override string ToString()`** zwracającą łańcuch opisujący datę urodzenia pracownika w formie: dzień miesiąc rok.
    Należy skorzystać z metody: **`String.Format()`**
  - **`static string ZwrocMiesiac(int miesiac)`** zwracającą miesiąc roku w formie łańcucha na podstawie argumentu metody. Wybrany miesiąc można wydobyć poprzez odwołanie się do odpowiedniego elementu tablicy **`miesiace`**. Metoda ma ponadto ograniczyć błędnie podany miesiąc do przedziału 1, …, 12.

Uwaga: klasa **`Data`** musi być zdefiniowana z dostępem pubicznym. W tym celu przed definicją klasy należy umieścić modyfikator **`public`**.

1.4. Definicja klasy `Adres`

W analogiczny sposób do projektu należy dodać klasę **`Adres`**. W klasie **`Adres`** należy zdefiniować następujące składowe:

- Prywatne pola:
  - **`string ulica`**,
  - **`string numerDomu`**,
  - **`string miasto`**.
- Publiczne właściwości:
  - **`Ulica`** zwracającą i ustawiającą wartość pola **`ulica`**,
  - **`NumerDomu`** zwracającą i ustawiającą wartość pola **`numerDomu`**,
  - **`Miasto`** zwracającą i ustawiającą wartość pola **`miasto`**.
- Publiczne metody:
  - bezargumentowy konstruktor domyślny.
  - konstruktor inicjalizujący wszystkie pola składowe na podstawie odpowiednio dobranych argumentów: **`Adres(string ulica, string numerDomu, string miasto)`**
  - konstruktor kopiujący inicjalizujący wszystkie pola składowe na podstawie argumentu wzorcowego: **`Adres(Adres wzor)`**.
  - **`override string ToString()`** zwracającą łańcuch opisujący adres pracownika w formie: ulica numer-domu miasto.
    Należy skorzystać z metody: **`String.Format()`**

Uwaga: klasa **`Adres`** musi być zdefiniowana z dostępem pubicznym. W tym celu przed definicją klasy należy umieścić modyfikator **`public`**.

1.5. Definicja klasy `Pracownik`

Postępując analogicznie jak w punktach **1.3**. i **1.4**, do szablonu projektu należy dodać klasę **`Pracownik`**. W klasie **`Pracownik`** należy zdefiniować następujące składowe:

- Prywatne pola:
  - **`string imie`**,
  - **`string nazwisko`**,
  - **`Data dataUrodzenia`**,
  - **`Adres adresZamieszkania`**.
- Publiczne właściwości:
  - **`Imie`** zwracającą i ustawiającą wartość pola **`imie`**.
  - **`Nazwisko`** zwracającą i ustawiającą wartość pola **`nazwisko`**.
  - **`DataUrodzenia`** zwracającą wartość pola **`dataUrodzenia`**.
  - **`AdresZamieszkania`** zwracającą wartość pola **`adresZamieszkania`**.
- Publiczne metody:
  - bezargumentowy konstruktor domyślny, w którym należy zainicjalizować pola typów referencyjnych: **`dataUrodzenia`** oraz **`adresZamieszkania`** wywołując konstruktor domyślny.
  - konstruktor inicjalizujący wszystkie pola składowe na podstawie argumentów:
    `Pracownik(string imie, string nazwisko, int dzien, string miesiac, int rok, string ulica, string numerDomu, string miasto)`
  - konstruktor kopiujący inicjalizujący wszystkie pola składowe na podstawie argumentu wzorcowego: **`Pracownik(Pracownik wzor)`**
  - **`Pracownik Clone()`**, zwracającą instancję nowostworzonego obiektu na wzór danego (wywołanie konstruktora kopiującego).

- **override string ToString()** zwracającą łańcuch opisujący dane pracownika w formie: imię nazwisko dzień miesiąc rok ulica numer-domu miasto.
  Łańcuch ten można stworzyć w następujący sposób:
  String.Format("{0} {1} {2} {3}", imie, nazwisko, dataUrodzenia, adresZamieszkania);
- **virtual string FormatWyjsciowy()** zwracającą łańcuch opisujący dane pracownika w formie:
  Imię, nazwisko: imię nazwisko
  Data urodzenia: dzień miesiąc rok
  Adres zamieszkania: ulica numer-domu miasto
- **string DataToString()** zwracającą łańcuch opisujący datę pracownika w formie: dzień miesiąc rok
- **string AdresToString()** zwracającą łańcuch opisujący adres pracownika w formie: ulica numer-domu miasto
- **void OdczytConsole()**, której zadaniem jest wczytanie z klawiatury wszystkich danych dla pracownika.
- **void ZapisConsole()**, której zadaniem jest wypisanie na ekran wszystkich danych pracownika.

Uwaga: klasa **Pracownik** musi być zdefiniowana z dostępem pubicznym. W tym celu przed definicją klasy należy umieścić modyfikator **public**.

1.6. Metoda główna Main

W metodzie głównej **Main** klasy **Program** należy stworzyć obiekty typu **Data**, **Adres** oraz **Pracownik** i wywołać poszczególne metody na rzecz tych obiektów.