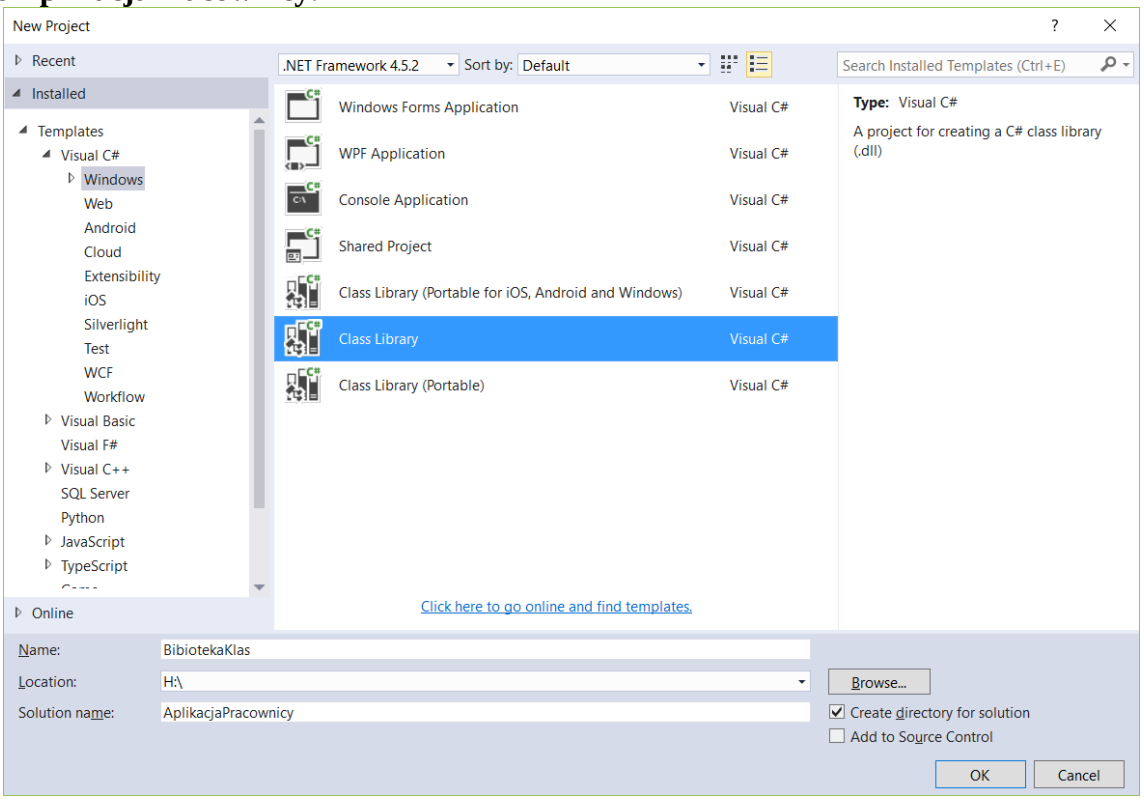
Zajęcia 1. Stworzenie projektu typu biblioteka klas (zawierającego klasy `Data`, `Adres` `Pracownik`) oraz projektu działającego w trybie wiersza poleceń

1.1. Wstęp

Celem pierwszej części ćwiczenia jest stworzenie projektu o szablonie **Class Library**. Szablon ten stanowić będzie bibliotekę klas **dll** dla aplikacji działającej w trybie tekstowym wiersza poleceń oraz docelowo – aplikacji graficznego interfejsu użytkownika opartego na formularzach. Projekt typu **Class Library** będzie składową kontenera (*ang. solution*), który w przyszłości zostanie rozbudowany o moduły komunikacji z użytkownikiem, co rozdzieli logikę (funkcjonalność) aplikacji od jej wyglądu. W skład tworzonego szablonu projektu wchodzić będą klasy (reprezentowane poprzez odpowiednie pliki), 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.
Celem drugiej części ćwiczenia jest stworzenie projektu o szablonie **Console Application** i dodanie go do kontenera aplikacji. W projekcie tym stworzona zostanie jedna klasa o nazwie `Program` z metodą główną `Main`. W metodzie tej konieczne będzie zaprezentowanie sposobu tworzenia obiektów klas `Data`, `Adres` oraz `Pracownik`.

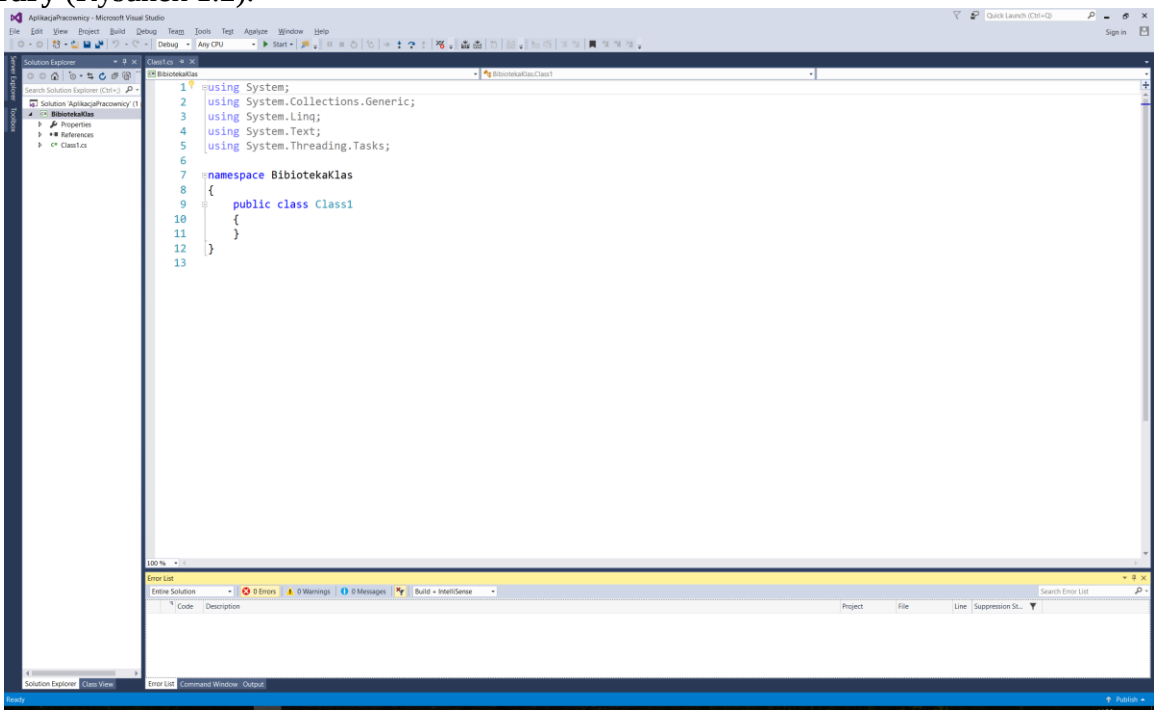
1.2. Stworzenie szablonu Class Library w ramach projektu typu Windows

Aby w środowisku **Visual Studio** utworzyć nowy projekt dla biblioteki klas, należy z menu **File** wybrać opcję **New|Project**. Następnie w panelu **Project types** rozwinąć rodzaj projektu **Visual C#**, wybrać typ **Windows** i w oknie szablonów (**Templates**) zaznaczyć: **Class Library**. W polu tekstowym **Location** należy wpisać ścieżkę, w której zostanie umieszczony szablon: **H:**, a w polu **Name** podać dowolną jego nazwę, np. **BibiotekaKlas**. Aby projekt był składową kontenera, konieczne jest podanie nazwy kontenera (pole **Solution Name**) oraz zaznaczenie opcji **Create directory for solution** w celu umieszczenia go w odrębnym katalogu. Rysunek 1.1 przedstawia sposób tworzenia szablonu biblioteki klas w ramach projektu typu **Windows** w kontenerze **AplikacjaPracownicy**.



Rysunek 1.1: Tworzenie projektu typu **Windows** o wybranym szablonie **Class Library**.

Kliknięcie na przycisk **OK** spowoduje stworzenie projektu typu **Windows** z jednym szablonem typu **ClassLibrary** (Rysunek 1.2).



Rysunek 1.2: Widok projektu typu **ClassLibrary** wraz z automatycznie wygenerowaną klasą `Class1`.

Jak można zauważyć całe okno projektu składa się z kilku mniejszych paneli, m. in.: edycyjnego (tj. do pisania kodu) na środku – z zakładką o nazwie **Class1.cs**, podłużnego u dołu – które wyświetla listę błędów, ostrzeżeń lub informacji w programie i używane jest do podglądu wartości obiektów, zmiennych podczas procesu *debugowania* (tj. pracy krokowej). Po lewej stronie znajduje się okno przeglądania zawartości całego kontenera z zakładkami **Toolbox**, **Class View** oraz **Solution Explorer**. Warto zwrócić uwagę, że (przy aktywnej zakładce **Solution Explorer**) w drzewie katalogów w tym oknie znajduje się plik **Class1.cs**. Na tym etapie tworzenia projektu nie jest konieczne przywiązywanie uwagi do funkcjonalności pozostałych zakładek czy okien.
W oknie edycyjnym (plik **Class1.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 `BibliotekaKlas`. W przestrzeni tej zdefiniowana jest klasa o nazwie `Class1`. Warto mieć na uwadze fakt, iż wygląd okna projektu może różnić się w zależności od domyślnych ustawień środowiska **Visual Studio**.

1.3. Definicja klasy `Data`

Ze względu na to, iż środowisko Visual Studio wygenerowało już jedną klasę (`Class1`) w stworzonym szablonie `BibliotekaKlas`, plik tej klasy można wykorzystać do definicji typu `Data`. Należy jednak zmienić domyślną nazwę `Class1` na `Data`. W tym celu wystarczy kliknąć prawym klawiszem myszy na plik **Class1.cs** w oknie przeglądania zawartości kontenera przy aktywnej zakładce **Solution Explorer**, wybrać z podręcznego menu opcję **Rename**, w zaznaczone pole wpisać **Data.cs** i zatwierdzić operację klawiszem **Enter**. Spowoduje to wyświetlenie oka dialogowego z zapytaniem czy aktualizację zmiany nazwy pliku wprowadzić w całym projekcie. Kliknięcie na przycisk **OK** zmodyfikuje nazwę pliku oraz klasy – od tej pory w przestrzeni `BibliotekaKlas` szablonu klas istniała będzie tylko jedna klasa o nazwie `Data`.
W klasie `Data` należy zdefiniować następujące składowe:

- Prywatne pola:
  - `int dzien;`

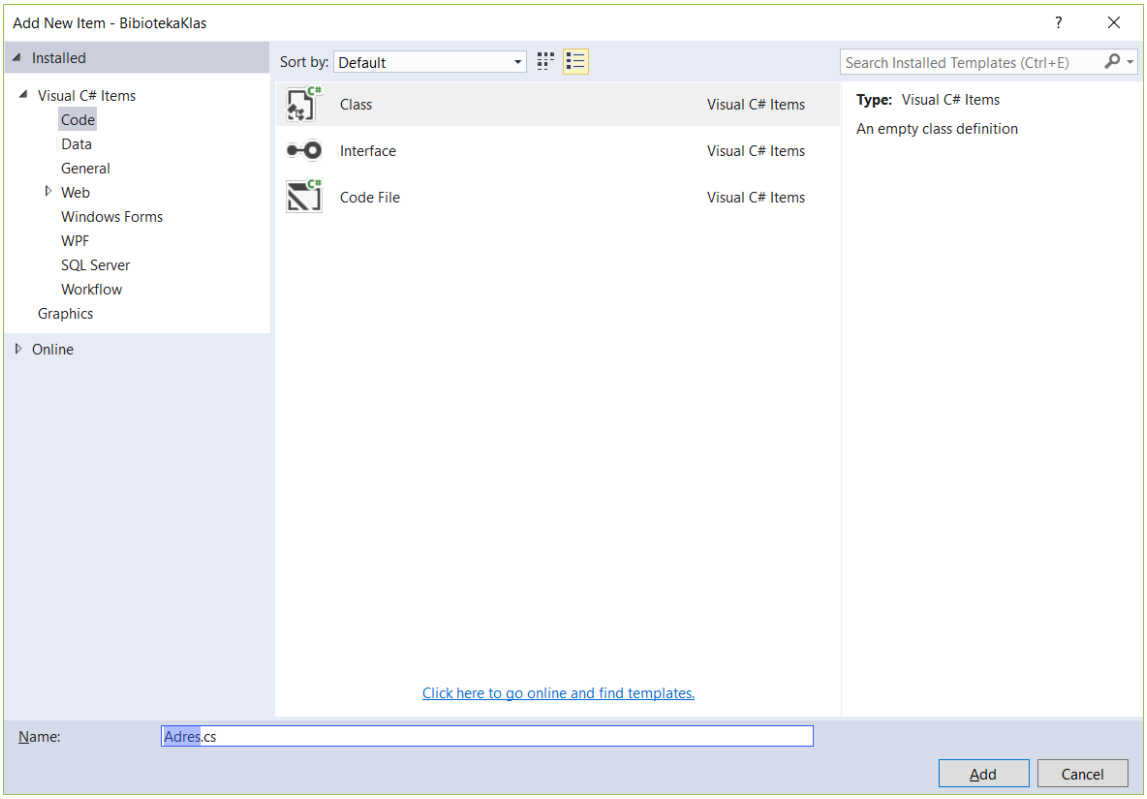
  - `string miesiac;`
  - `int rok;`
  - `static readonly string[] miesiace,` którą należy wypełnić łańcuchami nazw poszczególnych miesięcy.
- Publiczne właściwości:
  - `Dzien` zwracającą i ustawiającą wartość pola `dzien`.
  - `Miesiac` zwracającą i ustawiającą wartość pola `miesiac`.
  - `Rok` zwracającą i ustawiającą wartość pola `rok`.
- Publiczne metody:
  - bezargumentowy konstruktor domyślny.
  - konstruktor inicjalizujący wszystkie pola składowe na podstawie odpowiednio dobranych argumentów.
  - konstruktor kopiujący inicjalizujący wszystkie pola składowe na podstawie argumentu wzorcowego typu `Data`.
  - `override string ToString()` zwracającą łańcuch opisujący datę urodzenia pracownika w formie: dzień miesiąc rok.
  - `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.

1.4. Dodanie klasy `Adres` do biblioteki klas

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

- wybranie opcji **Add Class…** z menu **Project**,
- kliknięcie przycisku **Add New Item** na pasku przycisków (lub ewentualnie rozwinięcie menu przy przycisku i wybranie opcji **Add Class…**),
- kliknięcie prawym przyciskiem myszki na nazwie projektu **BibliotekaKlas** w oknie przeglądania zawartości kontenera (zarówno przy aktywnej zakładce **Class View** jak i **Solution Explorer**) oraz wybranie opcji **Class** z menu **Add**.

Skorzystanie z dowolnej z wyżej wymienionych opcji dodawania klasy spowoduje, że na ekranie pojawi się okno dialogowe dodawania składnika (klasa „widziana” jest w środowisku Visual Studio jako składnik bo reprezentuje kod źródłowy i jest zdefiniowana w odrębnym pliku). Aby zawężyć możliwość wyboru dodawanego składnika do szablonu projektu, wygodnie jest zaznaczyć opcję **Code** w panelu **Categories**, a następnie **Class** w oknie **Templates**. Teraz wystarczy w polu **Name** zmienić nazwę klasy **Class1.cs** na **Adres.cs** a następnie kliknąć przycisk **Add**. Do projektu w przestrzeni `BibliotekaKlas` dodana zostanie klasa `Adres`. Rysunek 1.3 prezentuje omawiane okno dialogowe.



Rysunek 1.3: Okno dodawania składnika (klasy) do szablonu projektu.

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.
  - konstruktor kopiujący inicjalizujący wszystkie pola składowe na podstawie argumentu wzorcowego typu `Adres`.
  - `override string ToString()` zwracającą łańcuch opisujący adres pracownika w formie: ulica numer-domu miasto.

1.5. Dodanie typu wyliczeniowego do biblioteki klas

Definicja typu wyliczeniowego w ramach przestrzeni `BibliotekaKlas` może zostać umieszczona w oddzielnym pliku lub w pliku klasy poza jej definicją. Definicja wyliczenia w osobnym pliku wymaga dodania do biblioteki klas kolejnego składnika w identyczny sposób jak podczas dodawania klasy, z tym wyjątkiem, że w panelu **Templates** kategorii konieczne jest zaznaczenie szablonu **Code File** (Rysunek 1.3). Nazwa pliku dla wyliczenia może być dowolna, ale z rozszerzeniem **cs**. Po zatwierdzeniu operacji przyciskiem **OK**, w

nowostworzonym pliku źródłowym należy umieścić definicję typu wyliczeniowego, którą umieszczono w Listingu 1.1.

```
namespace BibliotekaKlas
{
    public enum Zawody : short
    {
        Pracownik, Informatyk, Lekarz, Nauczyciel
    }
}
```

Listing 1.1: Definicja typu wyliczeniowego `Zawody`.

Kod źródłowy przedstawiony na Listingu 1.1 definiuje wyliczenie `Zawody` typu `short` o czterech składowych `Pracownik`, `Informatyk`, `Lekarz`, `Nauczyciel` zainicjalizowanych domyślnie odpowiednio wartościami: `0`, `1`, `2`, `3`. Warto zwrócić uwagę na to, że wyliczenie `Zawody` umieszczone jest w zakresie przestrzeni `BibliotekaKlas`, przez co widoczne jest w całej strukturze projektu.

1.6. Dodanie klasy `Pracownik` do biblioteki klas

Postępując analogicznie jak w punkcie **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`.
  - `Zawod` zwracającą składową `Pracownik` wyliczenia `Zawody`. Właściwość ta ma być wirtualna.
- 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 odpowiednio dobranych argumentów.
  - konstruktor kopiujący inicjalizujący wszystkie pola składowe na podstawie argumentu wzorcowego typu `Pracownik`.
  - `virtual 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.
  - `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
  - `virtual string SzczegolyZawodu()` zwracającą łańcuch `"brak"`.
  - `string DataToString()` zwracającą łańcuch opisujący datę pracownika w formie: dzień miesiąc rok

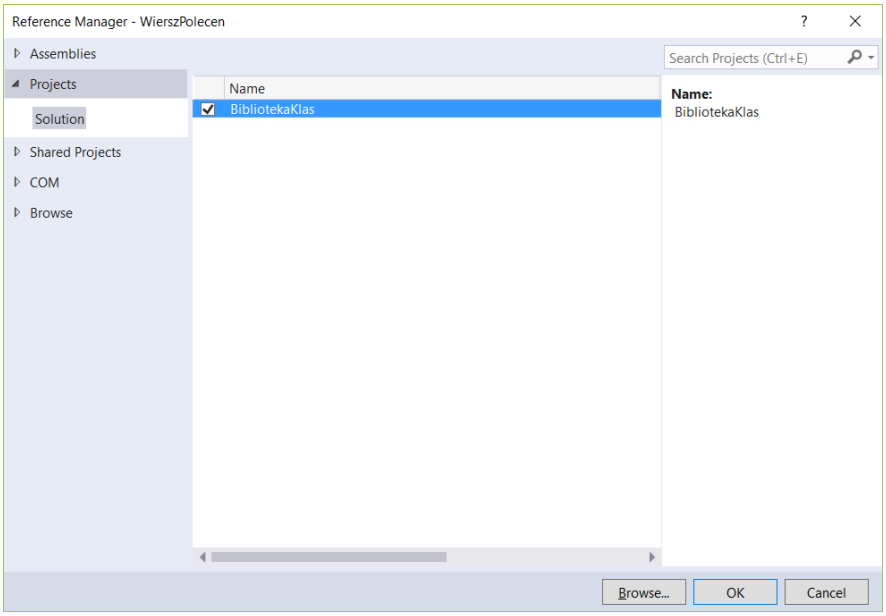
- `string AdresToString()` zwracającą łańcuch opisujący adres pracownika w formie: ulica numer-domu miasto
- `virtual void OdczytConsole()`, której zadaniem jest wczytanie z klawiatury wszystkich danych dla pracownika.
- `virtual void ZapisConsole()`, której zadaniem jest wypisanie na ekran wszystkich danych pracownika.
- `virtual void OdczytXml(DataRow dr)` – ciało metody należy na chwilę obecną zostawić puste. Uwaga: Ze względu na to, iż argumentem metody `OdczytXml` jest obiekt typu `DataRow`, w pliku definicji klasy (**Pracownik.cs**) konieczne jest dołączenie przestrzeni nazw `System.Data` za pomocą dyrektywy `using`.

1.7. Dodanie projektu typu Console Application do kontenera AplikacjaPracownicy

Sposób dołączania projektu o szablonie typu **Console Application** do kontenera aplikacji przeprowadza się w analogiczny sposób jak w przypadku dodawania szablonu biblioteki klas (podrozdział 1.1). Można to zrealizować wybierając z menu **File** opcję **New|Project** lub zaznaczając opcję **New Project** w menu **Add** po kliknieciu prawym przyciskiem myszki na nazwie kontenera (**AplikacjaPracownicy**) w panelu Solution Explorer. Następnie w panelu **Project types** wystarczy tylko rozwinąć rodzaj projektu **Visual C#**, wybrać typ **Windows** i w oknie szablonów (**Templates**) zaznaczyć: **Console Application**. W polu tekstowym **Location** należy wpisać ścieżkę, w której zostanie umieszczony szablon: **h:**, a w polu **Name** podać dowolną jego nazwę, np. **WierszPolecen**. Projekt będzie automatycznie składową wcześniej utworzonego kontenera.

Klasa `Program`

Po dodaniu drugiego projektu do kontenera, automatycznie w katalogu tego projektu, zostanie wygenerowana klasa `Program` w ramach przestrzeni `WierszPolecen`. Będzie ona również widoczna w formie pliku (**Program.cs**) w panelu **Solution Explorer** oraz jako typ w panelu **Class View**.
Jak można zaobserwować, kreator środowiska programistycznego wyposażył klasę `Program` w statyczną metodę `Main`. W ciele tej metody należy umieścić kod źródłowy, który umożliwi przeprowadzanie różnych operacji na danych pracowników. Ze względu na to, iż klasa `Program` znajduje się w innej przestrzeni nazw niż klasy odpowiedzialne za całościowe funkcjonowanie aplikacji, w metodzie `Main` nie będzie możliwe definiowanie obiektów wcześniej stworzonych typów (`Data`, `Adres` oraz `Pracownik`) i wywołanie na ich rzecz poszczególnych metod. Aby to umożliwić, projekt **WierszPolecen** musi posiadać referencję do wszystkich klas projektu **BibliotekaKlas**. Taką referencję należy dodać do projektu poprzez kliknięcie prawym klawiszem myszki na nazwę projektu **WierszPolecen** w widoku **Solution Explorer** lub **Class View** i wybranie z podręcznego menu **Add** opcji **Reference**. Na ekranie pojawi się wówczas okno dialogowe przedstawione na Rysunku 1.4, w którym w panelu (po lewej) **Projects-Solution** należy zaznaczyć dodawany projekt **BibliotekaKlas** i kliknąć przycisk **OK**.



Rysunek 1.4: Okno dodawania referencji do projektu **BibliotekaKlas**.
Uwaga: ścieżka widoczna na rysunku w zakładce **Projects** będzie zależna od lokalizacji kontenera aplikacji.

Następnie, w pliku **Program.cs**, przed definicją metody `Main`, należy za pomocą dyrektywy `using` umożliwić automatyczny import wszystkich wymaganych typów zdefiniowanych w przestrzeni **BibliotekaKlas**. W środowisku Visual Studio 2015, instrukcja `using BibliotekaKlas;` powinna zostać dodana automatycznie. Powyższe dwa zabiegi umożliwią wykorzystanie wszystkich klas w odrębnej przestrzeni w ramach jednego kontenera.

Metoda `Main`

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

Uruchomienie aplikacji

Stworzona do tej pory aplikacja, składa się z dwóch odrębnych projektów. Pierwszy z nich, **BibliotekaKlas**, definiuje w swojej przestrzeni zbiór typów wymaganych do kompletnej funkcjonalności tworzonego programu. Drugi projekt, o nazwie **WierszPolecen**, udostępnia metodę `Main`, która jest odpowiedzialna za formę, styl i sposób działania listy pracowników. Ze względu na to, iż projekt **WierszPolecen** posiada metodę `Main`, aplikację można uruchomić z poziomu tego projektu – wystarczy tylko, aby był on ustawiony jako projekt uruchomieniowy. W tym celu z menu **Project** należy wybrać opcję **Set StartUp Projects…** i w oknie dialogowym wybrać **WierszPolecen** jako pojedynczy projekt uruchomieniowy (**Single startup project**). Jeżeli opcja ta jest niewidoczna (lub niedostępna), wystarczy kliknąć prawym klawiszem myszki na nazwę **WierszPolecen** w panelu **Solution Explorer** i z podręcznego menu wybrać **Set as StartUp Project**. Od tej pory aplikację można już uruchomić (opcja **Start Without Debugging Ctrl+F5** z menu **Debug**) a efekt jej działania będzie bazował na kodzie źródłowym umieszczonym w metodzie głównej `Main` klasy `Program`.