**[KURS C#] Zacznijmy od rusztowania**



**UWAGA!**

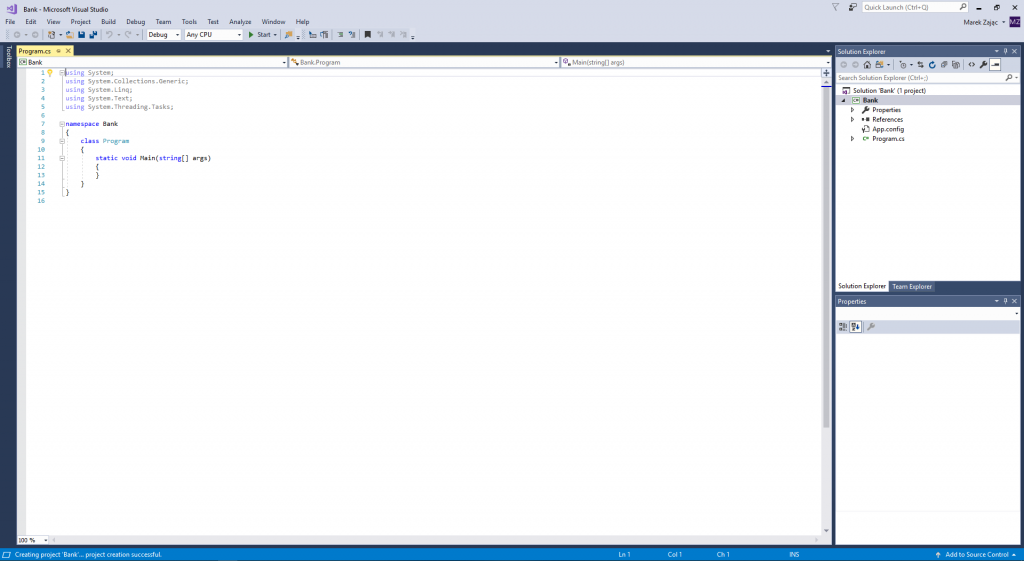
**Nowa wersja kursu dostępna** jest na moim portalu z kursami. Nadal jest on **BEZPŁATNY**! A jednocześnie posiada więcej lekcji, wygodną nawigację i możliwość zapamiętania, w którym miejscu skończyłeś ostatnio!. Znajdziesz go [**POD TYM LINKIEM**](https://kurs-szarpania.pl/nauka/product/szkola-szarpania-intro)!

Poprzednio udało nam się przygotować nasz warsztat do pracy i poznaliśmy podstawowe pojęcia jakimi będziemy się posługiwać w programach. Możemy w końcu przystąpić do pracy! Już nic nas nie powstrzyma przed przygotowaniem dla kolegi Jacka aplikacji do obsługi jego banku.

**Ruszamy!**

Otwórz utworzony wcześniej projekt (możesz to zrobić uruchamiając plik „**Bank.sln**” w folderze z projektem, wybierając ten plik poprzez” **File->Open->Project/Solution…”** albo na ekranie startowym Visual Studio klikając w projekt widoczny na liście **„Recent”**).

Kiedy wszystko się już załaduje to zobaczymy, że zostawiliśmy aplikację w stanie widocznym poniżej:

[](https://zajacmarek.com/wp-content/uploads/2018/11/c_sharp_new_project.png)

Pora więc na wyjaśnienie z czym właściwie mamy tutaj do czynienia.

W tym momencie interesuje nas to co znajduje się w **edytorze**. Zawartość **Solution Explorera** nie jest teraz ważna. Warto jedynie wiedzieć, że otwarty plik to **Program.cs**, który się tam znajduje. Wszystkie pliki z kodem w języku **C#** mają rozszerzenie **\*.cs** więc od razu będziesz wiedział, czy to co jest na liście plików to kod czy np. jakieś pliki z konfiguracją.

**Punkt zaczepienia**

Zawartość edytora, którą dostajemy na początku to **pełnoprawny program**:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Bank

{

class Program

{

static void Main(string[] args)

{

}

}

}

Wracając pamięcią do etapu przygotowywania warsztatu przypominasz sobie zapewne, że wybraliśmy **„Console Application”** jako typ aplikacji jaką tworzyliśmy. Działa ona w **konsoli systemu Windows**.

Na początku jest to najczytelniejszy typ projektu. Działająca całość znajduje się powyżej. Jest to cały kod działającego **programu konsolowego** w języku C#. Czas abyśmy poznali co właściwie się w nim znajduje.

Pojęcia jakie się tutaj pojawią będziemy omawiać w kolejnych częściach naszego projektu. Jeżeli już nie możesz wytrzymać nie rozumiejąc dokładnie co się tutaj dzieje to nie będziesz musiał długo czekać. Omówimy te rzeczy w lekcjach poświęconych [klasom](https://zajacmarek.com/kurs-c-praca-z-klasa/) i [funkcjom](https://zajacmarek.com/kurs-c-zrobisz-cos-dla-mnie/).

**Klasa programu**

Pierwszą rzeczą jaką wspomnę w tym miejscu jest **klasa** Program. Zapewne wszyscy już domyślili się, że jest to coś związanego z klasami, o których trochę teorii przedstawiłem w [poprzedniej części](https://zajacmarek.com/kurs-c-klocki-ukladanki/) ponieważ znajduje się tam słowo kluczowe ***class***.

Każdy **projekt w C#** musi składać się z **przynajmniej jednej klasy**, bo praktycznie wszystko co będziemy dodawać musi znajdować się w jakiejś klasie. Stąd nawet najprostszy program ją zawiera.

Skoro porównywaliśmy klasy do schematów jakichś urządzeń to tutaj możemy powiedzieć, że **klasa** Program **to taki schemat budowy naszej aplikacji**.

Kolejne klasy, które będziemy dodawać będą schematami komponentów tego urządzenia zwanego aplikacją do zarządzania bankiem.

**Główna funkcja**

Kolejna rzecz to ta dziwna linijka zaczynająca się od słowa static . Jest to **główna funkcja** naszego programu.

O funkcjach będziemy się uczyć później, kiedy projekt dla naszego kolegi będzie musiał wykonywać różnego rodzaju operacje. Teraz wystarczy zapamiętać, że ta funkcja Main  (to jest jej nazwa, tak jak **f** jest **nazwą funkcji f(x)**) to główna funkcja naszej aplikacji. To ona rozpoczyna działanie całej maszynerii jaka później powstanie.

**Przestrzeń nazw**



Następnym elementem jest tajemnicze słówko namespace  widoczne ponad klasą. Jest to **przestrzeń nazw** w jakiej znajduje się nasza klasa. Nie omawialiśmy wcześniej tego terminu.

Na początek wystarczy żebyś porównał sobie przestrzeń nazw do folderu. Zwłaszcza, że jeżeli niczego nie będziesz sam zmieniał to przestrzenie nazw będą właśnie odpowiadały folderom w projekcie.

Tutaj widzimy nazwę **Bank**. Jest to nazwa naszego projektu. Porównanie do folderów jest tym bardziej zasadne, że jeżeli otworzymy **eksplorator plików** i przejdziemy do **folderu** z naszym projektem to zobaczymy, że w nim znajduje się jeszcze **folder Bank** i dopiero w nim siedzi plik ***Program.cs***, który zawiera wspomnianą wcześniej klasę.

**Poproszę nazwę z półki…**

Ostatnia rzecz do omówienia to te linijki na początku pliku rozpoczynające się od słowa using . Służą one do **dołączania innych przestrzeni nazw** do pliku tak żebyśmy mogli używać znajdujących się tam **klas**.

Porównując to do przewijających się już schematów możemy powiedzieć, że jest to takie jakby wezwanie pracownika innego działu, żeby poinformował nas o dostępnych u nich projektach, które będziemy mogli używać w naszym schemacie.

Każdy **namespace** (przestrzeń nazw) to taki jakby osobny **dział**, który sam z siebie nic nie wie o tym co jest dostępne gdzieś indziej. W tym celu musi wezwać do siebie pracowników innego działu, którzy powiedzą co oferują albo przejrzeć katalog tych innych działów.

To przedstawienie możliwości jest właśnie ukryte pod słowem using , po którym dopisana jest nazwa „działu” czyli przestrzeni nazw z jakiego będziemy chcieli korzystać.

W powyższym przypadku korzystamy z dobrodziejstw tego co przygotował nam **Microsoft w .NET Frameworku**.

**Witaj świecie!**

Jak już wspomniałem w części gdzie tworzyliśmy ten zalążek projektu, to co mamy już utworzone jest właściwie **działającą aplikacją**, którą można uruchomić.

Kiedy wciśniemy **F5** na klawiaturze ona się włączy, zrobi wszystko co ma zrobić – czyli nic – i się wyłączy. Warto by było jednak dodać już coś od siebie. Zaznaczyć swoją obecność. Podpisać jakoś nasze dzieło. Dlatego bez nadmiaru wyjaśnień wypiszemy na ekranie kto jest Twórcą tego projektu.

Dodajmy więc do kodu aplikacji dwie linijki, które **wyświetlą dwie linie tekstu**:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Bank

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Nazwa: Bank");

Console.WriteLine("Autor: Marek Zajac");

}

}

}

W tym momencie wystarczy żebyś wiedział, że w taki sposób możesz wyświetlić tekst w konsoli po uruchomieniu programu.

Mamy tutaj **przykład użycia funkcji**, konkretnie WriteLine() ,która wyświetla podany tekst i dodaje znak końca linii, tak że następny tekst wyświetli się w kolejnej linii.

Uruchommy nasz kod! W tym celu kliknij **F5** na klawiaturze albo przycisk **„Start”** w Visual Studio (polecana jest jednak opcja ze skrótem na klawiaturze – znajomość skrótów przyśpiesza pracę). Program się uruchomi, iiiii…. Coś mignęło i tyle?! Jak to?!

**Zaczekaj!**

Program zrobił to co mu powiedzieliśmy. Wyświetlił tekst na ekranie i skończył pracę. A że w tym przypadku skończenie pracy oznacza też **zamknięcie konsoli** to niewiele zdążyliśmy zobaczyć.

Jednak chyba ciężko będzie z takim programem pracować, prawda? Kolega Jacek może nie potrafić aż tak szybko czytać tego co się wyświetla na ekranie ;) Teraz to jednak naprawimy.

Pod linijkami wyświetlającymi tekst dodajmy jeszcze jedną, z kolejną dostępną funkcją – ReadKey() :

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Bank

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Nazwa: Bank");

Console.WriteLine("Autor: Marek Zajac");

Console.ReadKey();

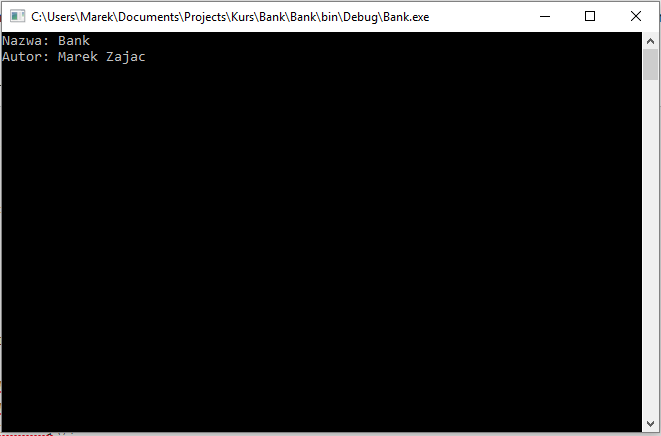
}

}

}

Teraz program wyświetli tekst, a następnym jego zadaniem będzie **odczytanie klawisza wciśniętego przez użytkownika**. Co to powoduje? Że program na to kliknięcie będzie czekał. A dopóki będzie czekał to nie może się zamknąć.

Mamy rozwiązanie naszego problemu! Uruchommy więc jeszcze raz aplikację. Tym razem na ekranie pojawi się widok podobny do tego:

[](https://zajacmarek.com/wp-content/uploads/2018/11/first_program_text.png)

Oczywiście zamiast mojego imienia i nazwiska wyświetli się Twoje jeżeli je zmieniłeś w kodzie.

Teraz program czeka aż **wciśniemy jakiś klawisz na klawiaturze**. Dopiero wtedy się zamknie.

**Warto zapamiętać**

Cała nasza wyprawa nastawiona jest na **praktykę**. Szybko przeszliśmy do pisania pierwszych **linijek kodu**. Jednak bez **solidnego przygotowania** nie ma co wyruszać. Bo czy poszedłbyś w góry nie wiedząc jaka jest przewidywana pogoda? Albo wsiadł do samochodu nie znając przepisów drogowych?

Ci, którzy tego próbują często marnie kończą. My nie chcemy marnie skończyć dlatego mam dla Ciebie jeszcze kilka słów wyjaśnienia niektórych kwestii.

**To jest kluczowe**

W kodzie naszego projektu dostrzec możemy różne dziwne, angielskie słowa takie jak ***using***, ***namespace***, ***class*** czy ***static***. Są to **słowa kluczowe**.

Są one **zarezerwowane przez język** i nie możemy ich używać jako nazw naszych klas czy funkcji. Listę zarezerwowanych słów znaleźć możemy na [stronie Microsoftu.](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/)

**W języku C# rozróżniana jest wielkość liter** dlatego ***Class*** nie jest tym samym co ***class****,*także to pierwsze słowo jak najbardziej możemy zastosować jako nazwa tego co piszemy.

**Zamknięte bloki**

W pierwszym kodzie jaki dostaliśmy poza nazwami rzucają nam się także w oczy **nawiasy klamrowe**. Za pomocą tych nawiasów wyznaczamy granice naszych klas czy funkcji. Tak jak powierzchnia kartki wyznacza granice narysowanych projektów.

Kiedy będziemy mówić o zawartości klasy to będziemy mieć na myśli to co znajduje się w nawiasach, z których otwierający jest najbliżej nazwy klasy:

class NaszaKlasa

{

// To jest wnętrze klasy

}

// To już nie jest wnętrze klasy

**Średnik na drogę**

Kiedy używamy czegoś to na końcu dajemy **znak średnika**. Oznacza on, że skończyliśmy daną komendę. Jest czymś w rodzaju **kropki w zdaniu**.

**Nie skomentuję tego**

W pokazanym powyżej przykładzie zobaczyć możemy również linijki zaczynające się od znaków // . W ten sposób dodajemy **w C# jednolinijkowe komentarze** w kodzie.

Wszystko co znajdzie się po podwójnym slashu **będzie zignorowane**. Możemy dzięki temu komentować kod, który piszemy aby np. opisać co robi linijka, którą napisaliśmy. Ważne aby nie przesadzić. Lepiej dawać wszystko mówiące nazwy klas i funkcji niż musieć opisywać do czego służą.

**W drogę!**

Projekt zaczyna nabierać rozpędu! Pierwszy kod jaki dopisaliśmy działa. Już krok dzieli nas od zawładnięcia światem! Tzn. od pomocy Jackowi w rozkręceniu bankowego biznesu ;)

Następny przystanek to już mocne uderzanie w klawisze. Do dzieła!