**[KURS C#] Zakładamy pierwsze konto!**



**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 nauczyliśmy się tworzyć zmienne, a więc pudełka na dane. Wiemy też jak wygląda klasa, czyli schemat obiektów, jakie będziemy robić. Aplikacja bankowa dla naszego kolegi Jacka nabiera rozpędu. Pora abyśmy założyli w niej **pierwsze konto**.

**Jedno konto proszę**

W ćwiczeniu do poprzedniej części dodałeś **zmienne** w klasie konta oszczędnościowego (**SavingsAccount**) i teraz wygląda ona tak:

namespace Bank

{

class SavingsAccount

{

public string AccountNumber;

public decimal Balance;

public string FirstName;

public string LastName;

public long Pesel;

}

}

Skoro jest szablon to warto z niego skorzystać. Teraz mamy możliwość utworzenia pierwszego konta oszczędnościowego! Skorzystajmy więc z niej.

W pliku ***Program.cs*** dopiszmy po linijkach odpowiedzialnych za wyświetlanie nazwy aplikacji i autora linijkę, która robi coś bardzo ważnego – **tworzy nowe konto**:

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)

{

string name = "Nazwa: Bank";

string author = "Autor: Marek Zajac";

Console.WriteLine(name);

Console.WriteLine(author);

SavingsAccount savingsAccount = new SavingsAccount();

Console.ReadKey();

}

}

}

Właśnie **program bankowy**, nad którym tak pracujemy **założył pierwsze konto**. „Jak to?” zapytasz. Już odpowiadam.

**Budujemy**

Przyjrzyjmy się tej **jednej linijce kodu**:

SavingsAccount savingsAccount = new SavingsAccount();

Dzieją się tutaj **trzy rzeczy** i przynajmniej dwie z nich poznaliśmy już poprzednio:

* **Definiujemy zmienną** o nazwie **savingsAccount**, która jest typu **SavingsAccount**, co oznacza, że będzie mogła przechowywać **obiekty tego typu**.
* Tworzymy nowy **obiekt** klasy **SavingsAccount**. To jest nowość, o której za moment.
* **Przypisujemy** utworzony **obiekt** do naszej zmiennej, a więc chowamy go od pudełka podpisanego jako **savingsAccount**.

Skoro **pierwszy** i **trzeci** punkt są nam znane to przejdźmy do omówienia **punktu drugiego**.

Po pierwsze pojawia się tutaj **nowe słowo kluczowe** (listę słów kluczowych znajdziesz w jednym z wcześniejszych rozdziałów) – new . Mówi ono, że chcemy **utworzyć nowy obiekt**, co z resztą sugeruje sama nazwa.

Następnie mamy nazwę klasy, której obiekt tworzymy.

Ale z jakiegoś powodu na końcu są jeszcze **nawiasy okrągłe**. Są one dlatego, że używamy tutaj pewnej **funkcji** każdej klasy. Jednak co to dokładnie jest poznamy [przy okazji omawiania funkcji](https://zajacmarek.com/kurs-c-zrobisz-cos-dla-mnie/). W każdym razie taka konstrukcja czyli new  + NazwaKlasy  + ();  umożliwia nam **utworzenie**, wyprodukowanie nowego egzemplarza **obiektu** takiego typu jaki podamy.

**Proszę wypełnić formularz**

Założyliśmy pierwsze konto. Jednak do tej pory nie podaliśmy nawet do kogo ono należy! Przyszedł czas **uzupełnienia pustych miejsc** jakie przewidzieliśmy w obiekcie kiedy pisaliśmy **klasę**.

Nauczyliśmy się jak wkładać wartość do zmiennej. Ale tam zmienna była **typu tekstowego** więc po prostu przypisywaliśmy do niej tekst. Ale jak to zrobić kiedy mamy zmienną typu **SavingsAccount**? Przypisaliśmy wartość takiego typu do tej zmiennej, jednak w żaden sposób nie uzupełnialiśmy pustych przestrzeni, które w tym typie zaprojektowaliśmy. Co robić?

**Dostęp do wnętrza**

Tutaj jest pora na dowiedzenie się jak dostać się do tych przestrzeni, do tych zmiennych, które przewidzieliśmy w klasie.

Skoro do zmiennej utworzonej bezpośrednio w funkcji ***Main*** dostawaliśmy się po nazwie to tutaj też powinno być to możliwe. Jednak wpisanie samej nazwy zmiennej znajdującej się w klasie, np. **FirstName** niewiele pomoże. Bo skąd program ma wiedzieć, o którą konkretnie zmienną Ci chodzi? Przecież obiektów, które mają tą zmienną może być dużo więcej.

Musimy więc powiedzieć aplikacji o **jakie konkretnie konto** nam chodzi. Dlatego musimy je wybrać poprzez podanie **nazwy zmiennej** do jakiej to konto wstawiliśmy. Wtedy za pomocą **znaku kropki** możemy już wybierać zmienne obecne **w tym obiekcie**.

Porównując to do wspominanej kiedyś obudowy komputera. Mamy serwis komputerowy. Potrzebujemy zamontować zasilacz. Chcemy go umieścić w miejscu przeznaczonym na taki właśnie zasilacz. Czy po prostu wchodzimy do serwisu i montujemy ten komponent? Nie. Bo takie miejsce, gdzie będziemy go mogli przykręcić samo w sobie nie istnieje. Istnieje ono w kontekście jakiejś obudowy.

Dlatego najpierw musimy wziąć odpowiednią obudowę (**utworzony z klasy obiekt**), zdjąć panel boczny (**użycie znaku kropki**) i dopiero włożyć zasilacz (**przypisanie wartości do zmiennej** o podanej nazwie). Dlatego wypełnienie danych utworzonego konta będzie wyglądało następująco

SavingsAccount savingsAccount = new SavingsAccount();

savingsAccount.AccountNumber = "940000000001";

savingsAccount.Balance = 0.0M;

savingsAccount.FirstName = "Marek";

savingsAccount.LastName = "Zajac";

savingsAccount.Pesel = 92010133333;

W ten oto sposób nie tylko założyliśmy pierwsze konto, ale też **uzupełniliśmy jego dane**. Jak widać jesteśmy pierwszym klientem banku bo konto o numerze ‚1’ przypadło nam ;)

Być może zdziwi Cię litera *‚M’* przy stanie konta. Jest ona użyta dlatego, że domyślnie jeśli wpisujemy stałe liczby to liczba z przecinkiem jest typu double . Dodanie na końcu litery *‚M’* mówi wprost, że wpisujemy liczbę typu decimal . *‚M’* może kojarzyć się z ***‚Money’*** bo typ decimal  jest powszechnie używany w operacjach związanych **z pieniędzmi**.

**Drukujemy**



Jeżeli będziemy chcieli **wypisać na ekranie** dane konta to dostaniemy się do nich w analogiczny sposób jak robiliśmy to przypisując nowe wartości – skorzystamy ze **znaku kropki**.

Tak więc chcąc wypisać na ekranie wszystkie dane konta, które niedawno utworzyliśmy zrobimy to w ten sposób:

namespace Bank

{

class Program

{

static void Main(string[] args)

{

string name = "Nazwa: Bank";

string author = "Autor: Marek Zajac";

Console.WriteLine(name);

Console.WriteLine(author);

SavingsAccount savingsAccount = new SavingsAccount();

savingsAccount.AccountNumber = "940000000001";

savingsAccount.Balance = 0.0M;

savingsAccount.FirstName = "Marek";

savingsAccount.LastName = "Zajac";

savingsAccount.Pesel = 92010133333;

Console.WriteLine("Dane konta:");

Console.WriteLine(savingsAccount.AccountNumber);

Console.WriteLine(savingsAccount.Balance);

Console.WriteLine(savingsAccount.FirstName);

Console.WriteLine(savingsAccount.LastName);

Console.WriteLine(savingsAccount.Pesel);

Console.ReadKey();

}

}

}

Wszystko fajnie, możemy wypisać w konsoli wszystkie wartości, ale przy nazwie projektu i autorze mieliśmy to ładnie **sformatowane** – była etykieta co to jest, dwukropek i dopiero nazwa. A teraz wyświetla nam się na ekranie jakaś wartość i nawet nie wiadomo co to. Jacek przecież nie będzie się za każdym razem zastanawiał czy to co jest wypisane to tytuł przelewu czy nazwisko klienta.

Więc dodajmy też etykiety do tego co wyświetlamy na ekranie. Na szczęście funkcja WriteLine()  pozwala nam też skorzystać z **parametrów w tekście**. Oznacza to ni mniej ni więcej, że mamy możliwość wyświetlić tekst, w którym w odpowiednim miejscu pojawi się **wartość naszej zmiennej**. Robimy to w następujący sposób:

namespace Bank

{

class Program

{

static void Main(string[] args)

{

string name = "Nazwa: Bank";

string author = "Autor: Marek Zajac";

Console.WriteLine(name);

Console.WriteLine(author);

Console.WriteLine();

SavingsAccount savingsAccount = new SavingsAccount();

savingsAccount.AccountNumber = "940000000001";

savingsAccount.Balance = 0.0M;

savingsAccount.FirstName = "Marek";

savingsAccount.LastName = "Zając";

savingsAccount.Pesel = 92010133333;

Console.WriteLine("Dane konta:");

Console.WriteLine("Numer konta: {0}", savingsAccount.AccountNumber);

Console.WriteLine("Saldo: {0}zł", savingsAccount.Balance);

Console.WriteLine("Imię właściciela: {0}", savingsAccount.FirstName);

Console.WriteLine("Nazwisko właściciela: {0}", savingsAccount.LastName);

Console.WriteLine("PESEL właściciela: {0}", savingsAccount.Pesel);

Console.ReadKey();

}

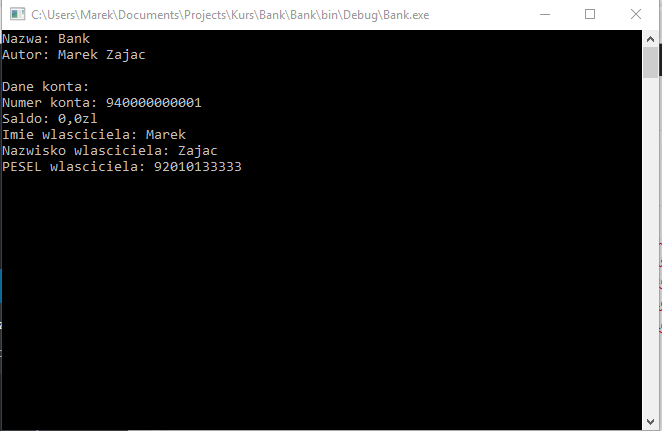
}

}

Teraz ta **funkcja** nie tylko przyjmuje tekst, ale też **po przecinku** przyjmuje to co chcemy wstawić w miejsca oznaczone w specjalny sposób – za pomocą **nawiasów klamrowych**, pomiędzy którymi znajduje się liczba mówiąca, którą wartość z tych podanych dalej ma reprezentować. Tylko liczone to jest od zera. W naszym przypadku mamy **jedną wartość** i jej numer to **0** dlatego w tekście wstawiamy {0} .

Nie przejmuj się jeżeli w tym momencie nie rozumiesz co się tutaj dokładnie dzieje jeśli chodzi o funkcję WriteLine() . Teraz chcę jedynie abyśmy mogli korzystać z jej dobrodziejstw mogąc wyświetlić na ekranie to i owo. Dlatego nie czuj się źle kopiując te fragmenty i tylko zmieniając we wskazanych miejscach wartości. Wiedza o tym co tutaj się dzieje pojawi się w przyszłych lekcjach.

Uruchommy aplikację wciskając klawisz **F5**. Jeżeli wszystko zrobiliśmy poprawnie, **na ekranie** powinniśmy zobaczyć **dane naszego pierwszego konta** wraz z ich opisami:

[](https://zajacmarek.com/wp-content/uploads/2018/11/variables_4.png)

**Ćwiczenie 1**

Utwórz **drugie konto oszczędnościowe** przypisując je do drugiej zmiennej. Uzupełnij jego dane, a następnie wyświetl na ekranie dane obu kont.

**Ćwiczenie 2**

Utwórz **obiekt konta rozliczeniowego**. Uzupełnij dane dotyczące **imienia**, **nazwiska** i numeru **PESEL** właściciela biorąc je z utworzonego wcześniej **konta oszczędnościowego** zamiast wpisywać jako tekst w kodzie. Pozostałe dane wpisz ręcznie.

Wyświetl na ekranie dane wszystkich kont. Następnie zmień nazwisko w **koncie oszczędnościowym**, z którego brałeś dane dla **konta rozliczeniowego** i uruchom program jeszcze raz.

Dane w obu kontach powinny się zmienić.

**Bank rośnie w siłę**

W tym momencie mamy już w aplikacji utworzone **trzy konta**. Całkiem dobry wynik jak na bank, który dopiero powstaje! Kolega powinien być zadowolony z tego, że całość zaczyna przypominać to czego oczekuje i potrzebuje.

W dodatku już za moment, już po przejściu do następnej części, dowiemy się jak dodać pierwsze operacje i zachowania do naszej aplikacji! Chodźmy więc.