

Jacek Matulewski

Visual Studio 2013

Podręcznik programowania w C# z zadaniami



Ucz się, projektuj, zarabiaj!

- Poznaj język C# 5.0 i platformę .NET 4.5.1
— podstawy nowoczesnego projektowania aplikacji Windows.
- Dowiedz się, jak projektować aplikacje Windows Forms i efektywnie używać kontrolek.
- Wybierz optymalny sposób przechowywania informacji w bazie danych w aplikacjach dla platformy .NET.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Ewelina Burska
Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/vs12pa>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-6856-4

Copyright © Helion 2014

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Wstęp	11
Część I Język C# i platforma .NET	13
Rozdział 1. Pierwsze spotkanie ze środowiskiem Visual Studio i językiem C#	15
Klasa Console	15
Projekt aplikacji konsolowej	15
Drukowanie napisów w konsoli i podpowiadanie kodu	18
Czekanie na akceptację użytkownika	19
Odczytywanie danych z klawiatury	20
Korzystanie z bibliotek DLL. Komunikat „okienkowy” w aplikacji konsolowej	21
Informacje o środowisku aplikacji	23
Podstawowe informacje o systemie i profilu użytkownika	23
Katalogi specjalne zdefiniowane w bieżącym profilu użytkownika	24
Odczytywanie zmiennych środowiskowych	25
Lista dysków logicznych	26
Rozdział 2. O błędach i ich tropieniu	27
Program z błędem logicznym — pole do popisu dla debugera	27
Kontrolowane uruchamianie aplikacji w Visual C#	28
Śledzenie wykonywania programu krok po kroku (F10 i F11)	28
Run to Cursor (Ctrl+F10)	30
Breakpoint (F9)	30
Okna Locals i Watch	31
Stan wyjątkowy	33
Zgłaszanie wyjątków	33
Przechwytywanie wyjątków w konstrukcji try..catch	35
Wymuszenie kontroli zakresu zmiennych	37
Rozdział 3. Język C# 5.0	39
Platforma .NET	40
Środowisko uruchomieniowe	40
Kod pośredni i podwójna kompilacja	40
Wersje	41
Skróty, które warto poznać	43
Podstawowe typy danych	44
Deklaracja i zmiana wartości zmiennej	44
Typy liczbowe oraz typ znakowy	45

Określanie typu zmiennej przy inicjacji (pseudotyp var)	47
Operatory	47
Konwersje typów podstawowych	49
Operatory is i as	50
Łańcuchy	51
Typ wyliczeniowy	54
Leniwe inicjowanie zmiennych	55
Metody	56
Przeciążanie metod	57
Domyślne wartości argumentów metod — argumenty opcjonalne	58
Argumenty nazwane	59
Wartości zwracane przez metody	59
Zwracanie wartości przez argument metody	59
Delegacje i zdarzenia	61
Wyrażenia lambda	64
Typy wartościowe i referencyjne	66
Nullable	67
Pudełkowanie	68
Typy dynamiczne	69
Sterowanie przepływem	72
Instrukcja warunkowa if..else	72
Instrukcja wyboru switch	73
Pętle	74
Wyjątki	75
Dyrektywy preprocesora	77
Kompilacja warunkowa — ostrzeżenia	78
Definiowanie stałych preprocesora	78
Bloki	79
Atrybuty	80
Kolekcje	81
„Zwykłe” tablice	81
Pętla foreach	83
Sortowanie	84
Kolekcja List	85
Kolekcja SortedList i inne słowniki	87
Kolejka i stos	88
Tablice jako argumenty metod oraz metody z nieokreśloną liczbą argumentów	89
Słowo kluczowe yield	90
Nowa forma inicjacji obiektów i tablic	92
Caller Information	93

Rozdział 4. Programowanie obiektowe w C# 95

Przykład struktury (Ułamek)	96
Przygotowywanie projektu	96
Konstruktor i statyczne obiekty składowe	98
Modyfikatory const i readonly	99
Pierwsze testy	99
Konwersje na łańcuch (metoda ToString) i na typ double	100
Metoda upraszczająca ułamek	101
Właściwości	102
Domyślnie implementowane właściwości (ang. auto-implemented properties)	104
Operatory arytmetyczne	105
Operatory porównania oraz metody Equals i GetHashCode	106
Operatory konwersji	108

Implementacja interfejsu (na przykładzie IComparable)	109
Definiowanie typów parametrycznych	110
Definiowanie typów ogólnych	111
Określanie warunków, jakie mają spełniać parametry	113
Implementacja interfejsów przez typ ogólny	114
Definiowanie aliasów	115
Typy ogólne z wieloma parametrami	116
Rozszerzenia	117
Typy anonimowe	119
Dziedziczenie	119
Klasy bazowe i klasy potomne	120
Nadpisywanie a przesłanianie metod	123
Klasy abstrakcyjne	124
Metody wirtualne	126
Polimorfizm	127
Konstruktory a dziedziczenie	129
Singleton	131
Interfejsy	134
Rozdział 5. Biblioteki DLL	137
Tworzenie zarządzanej biblioteki DLL	138
Dodawanie do aplikacji referencji do biblioteki DLL	139
Przenośne biblioteki DLL	140
Rozdział 6. Testy jednostkowe	143
Projekt testów jednostkowych	144
Przygotowania do tworzenia testów	144
Pierwszy test jednostkowy	145
Uruchamianie testów	146
Dostęp do prywatnych pól testowanej klasy	147
Testowanie wyjątków	148
Kolejne testy weryfikujące otrzymane wartości	148
Test ze złożoną weryfikacją	149
Powtarzane wielokrotnie testy losowe	151
Niepowodzenie testu	152
Nieuniknione błędy	154
Rozdział 7. Elementy programowania współbieżnego	159
Równoległa pętla for	159
Przerywanie pętli	161
Programowanie asynchroniczne. Modyfikator async i operator await (nowość języka C# 5.0)	162
Część II Projektowanie aplikacji Windows Forms	169
Rozdział 8. Pierwszy projekt aplikacji Windows Forms	171
Projektowanie interfejsu aplikacji	172
Tworzenie projektu	172
Dokowanie palety komponentów Toolbox	174
Tworzenie interfejsu za pomocą komponentów Windows Forms	175
Przełączanie między podglądem formy a kodem jej klasy	176
Analiza kodu pierwszej aplikacji Windows Forms	176
Metody zdarzeniowe	182
Metoda uruchamiana w przypadku wystąpienia zdarzenia kontrolki	182
Testowanie metody zdarzeniowej	183

Przypisywanie istniejącej metody do zdarzeń komponentów	185
Edycja metody zdarzeniowej	185
Modyfikowanie własności komponentów	186
Wywoływanie metody zdarzeniowej z poziomu kodu	186
Reakcja aplikacji na naciskanie klawiszy	187
Rozdział 9. Przegląd komponentów biblioteki Windows Forms	189
Notatnik.NET	189
Projektowanie interfejsu aplikacji i menu główne	189
Okna dialogowe i pliki tekstowe	196
Edycja i korzystanie ze schowka	204
Drukowanie	205
Elektroniczna kukułka	214
Ekran powitalny (splash screen)	214
Przygotowanie ikony w obszarze powiadamiania	217
Odtwarzanie pliku dźwiękowego	220
Ustawienia aplikacji	222
Dywan graficzny	225
Rozdział 10. Przeciągnij i upuść	231
Podstawy	231
Interfejs przykładowej aplikacji	232
Inicjacja procesu przeciągania	233
Akceptacja upuszczenia elementu	235
Reakcja na upuszczenie elementu	236
Czynności wykonywane po zakończeniu procesu przenoszenia i upuszczania	237
Przenoszenie elementów między różnymi aplikacjami	238
Zagadnienia zaawansowane	238
Opóźnione inicjowanie procesu przenoszenia	238
Przenoszenie wielu elementów	241
Przenoszenie plików	242
Część III Dane w aplikacjach dla platformy .NET	245
Rozdział 11. LINQ	247
Operatory LINQ	247
Pobieranie danych (filtrowanie i sortowanie)	249
Analiza pobranych danych	250
Wybór elementu	250
Weryfikowanie danych	251
Prezentacja w grupach	251
Łączenie zbiorów danych	252
Łączenie danych z różnych źródeł w zapytaniu LINQ — operator join	252
Możliwość modyfikacji danych źródła	253
Rozdział 12. Przechowywanie danych w plikach XML. LINQ to XML	255
Podstawy języka XML	255
Deklaracja	255
Elementy	256
Atrybuty	256
Komentarze	256
LINQ to XML	257
Tworzenie pliku XML za pomocą klas XDocument i XElement	257
Pobieranie wartości z elementów o znanej pozycji w drzewie	259
Odwzorowanie struktury pliku XML w kontrolce TreeView	261

Przenoszenie danych z kolekcji do pliku XML	262
Zapytania LINQ, czyli tworzenie kolekcji na bazie danych z pliku XML	263
Modyfikacja pliku XML	264
Rozdział 13. Baza danych SQL Server w projekcie Visual Studio	267
Odwzorowanie obiektowo-relacyjne	267
Szalenie krótki wstęp do SQL	269
Select	269
Insert	270
Delete	270
Projekt aplikacji z bazą danych	270
Dodawanie bazy danych do projektu aplikacji	270
Łańcuch połączenia (ang. connection string)	271
Dodawanie tabeli do bazy danych	272
Edycja danych w tabeli	274
Druga tabela	274
Procedura składowana — pobieranie danych	276
Procedura składowana — modyfikowanie danych	276
Procedura składowana — dowolne polecenia SQL	277
Widok	277
Rozdział 14. LINQ to SQL	279
Klasa encji	280
Pobieranie danych	281
Prezentacja danych w siatce DataGridView	283
Aktualizacja danych w bazie	283
Modyfikacje istniejących rekordów	284
Dodawanie i usuwanie rekordów	285
Inne operacje	286
Wizualne projektowanie klasy encji	286
O/R Designer	287
Współpraca z kontrolkami tworzącymi interfejs aplikacji	290
Korzystanie z widoków	291
Łączenie danych z dwóch tabel — operator join	292
Relacje (Associations)	292
Korzystanie z procedur składowanych	294
Pobieranie danych za pomocą procedur składowanych	294
Modyfikowanie danych za pomocą procedur składowanych	295
Rozdział 15. Kreator źródeł danych	297
Kreator źródła danych	297
Zautomatyzowane tworzenie interfejsu użytkownika	300
Prezentacja tabeli w siatce	300
Klasa BindingSource	301
Sortowanie	301
Filtrowanie	302
Prezentacja rekordów tabeli w formularzu	302
Dwie siatki w układzie master-details	304
Rozdział 16. Tradycyjne ADO.NET (DataSet)	307
Konfiguracja źródła danych DataSet	307
Edycja klasy DataSet i tworzenie relacji między tabelami	309
Podgląd danych udostępnianych przez komponent DataSet	310
Prezentacja danych w siatce	311
Zapisywanie zmodyfikowanych danych	314

Prezentacja danych w formularzu	316
Sortowanie i filtrowanie	319
Odczytywanie z poziomu kodu wartości przechowywanych w komórkach tabeli	319
Zapytania LINQ do danych z DataSet	322
Rozdział 17. Entity Framework	323
Tworzenie modelu danych EDM dla istniejącej bazy danych	324
Użycie klasy kontekstu z modelu danych EF	327
LINQ to Entities	329
Prezentacja i edycja danych w siatce	330
Asynchroniczne wczytywanie danych	332
Użycie widoku i procedur składowanych	333
Połączenie między tabelami	334
Tworzenie źródła danych	336
Automatyczne tworzenie interfejsu	338
Edycja i zapis zmian	341
Część IV Dodatki	343
Zadania	345
Skorowidz	355

Rozdział 5.

Biblioteki DLL

Biblioteki DLL (ang. *dynamic link library*) są głównym sposobem dystrybucji klas i komponentów zarówno w aplikacjach dla platformy Win32, jak i dla platformy .NET. Oczywiście biblioteki obu platform różnią się w takim samym stopniu jak ich pliki wykonywalne *.exe*¹. Niemniej jednak idea stojąca za umieszczaniem modułów aplikacji w osobnych bibliotekach jest podobna.

Poniżej pokazuję, jak utworzyć bibliotekę DLL zawierającą klasy .NET. Użyjemy do tego przygotowanej w poprzednim rozdziale klasy *Ułamek*. Następnie wykorzystamy ją w aplikacji konsolowej. Ograniczę się przy tym wyłącznie do tzw. statycznego łączenia bibliotek². W efekcie klasa *Ułamek* nie zostanie wkompiłowana do pliku *.exe* tej aplikacji — biblioteka będzie ładowana do pamięci w momencie uruchamiania aplikacji. Taki sposób postępowania ma kilka zalet. Jeżeli napiszemy lepszą wersję klasy *Ułamek*, ale taką, której interfejs nie ulegnie zmianie, to łatwo można zaktualizować cały program bez potrzeby ponownej kompilacji i dystrybucji pliku *.exe*. Wystarczy podmienić plik *.dll*. W większych aplikacjach może to bardzo ułatwić rozpowszechnianie uaktualnień. Ważne jest również to, że klasy umieszczone w bibliotece DLL mogą być wielokrotnie wykorzystywane przez różne aplikacje bez konieczności ich wkompiłowywania w każdą z nich osobno. Dodatkową zaletą jest przenośność bibliotek PCL (ang. *Portable Class Library*). Biblioteki tego typu mogą być używane bez rekompilacji w różnych platformach zarządzanych Microsoftu. Jest to szczególnie interesujące w kontekście niedawno ogłoszonej współpracy Microsoftu i Xamarin, który rozszerzy listę platform obsługiwanych przez te biblioteki o Android i iOS (na bazie platformy Mono).

¹ Zarówno pliki *.dll*, jak i pliki *.exe* są podzespołami (ang. *assembly*), czyli skompilowanymi do kodu pośredniego MSIL samodzielnymi kawałkami kodu. Na poziomie podzespołów realizowana jest główna polityka bezpieczeństwa platformy .NET.

² Zagadnienia dynamicznego ładowania biblioteki już w trakcie działania aplikacji oraz związane z tym zagadnienia dynamicznego typowania (słowo kluczowe *dynamic*) oraz korzystanie z platformy MEF do tworzenia bibliotek DLL funkcjonujących jako wtyczki omówione zostały w książce *Visual Studio 2010 dla programistów C#*, Helion, 2011 (obecnie dostępna w postaci ebooka). Umieszczone w tej książce informacje są nadal aktualne dla Visual Studio 2013.

Tworzenie zarządzanej biblioteki DLL

Jeżeli dysponujemy gotowymi klasami, które chcemy umieścić w zarządzanej bibliotece DLL, stworzenie takiej biblioteki zajmie krótką chwilę. Przeciwicmy to, umieszczając w bibliotece DLL klasę `Ulamek` z poprzedniego rozdziału.

1. Otwórz opisane w poprzednim rozdziale rozwiązanie *UlamekDemo* zawierające projekt *UlamekDemo* z klasą `Ulamek`.
2. Do projektu dodajmy projekt biblioteki. W tym celu:
 - a) Z menu *File* wybierz polecenie *New Project* lub wciśnij kombinację klawiszy *Ctrl+Shift+N*.
 - b) Zaznacz ikonę *Class Library*.
 - c) W polu *Name* wpisz nazwę `UlamekBiblioteka`.
 - d) Ponieważ chcemy nowy projekt dodać do istniejącego rozwiązania, należy w oknie *New Project* przełączyć rozwijaną listę *Solution* z *Create new solution* na *Add to solution*.
 - e) Kliknij *OK*.
3. Po utworzeniu projektu zobaczysz okno edytora kodu z definicją pustej klasy. Jednocześnie w podoknie *Solution Explorer* widoczny będzie nowy projekt w bieżącym rozwiązaniu, a w nim wyświetlany w edytorze plik *Class1.cs*. Usuńmy ten plik, zaznaczając go w podoknie *Solution Explorer* i naciskając klawisz *Delete*. Zostaniemy zapytani o potwierdzenie.
4. W jego miejsce skopiujemy plik *Ulamek.cs* z projektu *UlamekDemo*. Wystarczy go przeciągnąć w podoknie *Solution Explorer* i upuścić na pozycję *UlamekBiblioteka*.
5. W pliku *Ulamek.cs* musimy wprowadzić trzy zmiany (listing 5.1). Tylko pierwsza jest naprawdę ważna, dwie następne to zmiany czysto kosmetyczne.
 - a) Pierwszą jest dodanie do deklaracji klasy `Ulamek` modyfikatora `public`. Bez tego nie będzie ona widoczna poza biblioteką.
 - b) Drugą zmianą jest zmiana nazwy przestrzeni nazw. Wystarczy w kodzie zmienić nazwę za słowem kluczowym `namespace` z `UlamekDemo` na `Helion`.
 - c) Trzecia to usunięcie niepotrzebnych poleceń `using` w nagłówku pliku. Z zadeklarowanych tam przestrzeni nazw używamy tak naprawdę tylko przestrzeni `System`. Co ciekawe, nie musimy sami sprawdzać, które polecenia `using` są potrzebne, a które zbędne. Wystarczy, że z menu kontekstowego w edytorze wybierzemy pozycję *Organize Usings*, a z rozwiniętego w ten sposób podmenu — polecenie *Remove Unused Usings*.

Listing 5.1. Zmiany w pliku *Ulamek.cs*

```
using System;  
using System.Collections.Generic;  
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace Helion
{
    public struct Ułamek : IComparable<Ułamek>
    {
        private int licznik, mianownik;
        ...
    }
}
```

6. Aby sprawdzić, czy wszystko jest w porządku, skompilujmy projekt biblioteki. W tym celu z menu kontekstowego pozycji *UłamekBiblioteka* wybierzmy polecenie *Build*. Po chwili na pasku stanu Visual Studio powinniśmy zobaczyć komunikat *Build succeeded*.



Wskazówka

Efektom kompilacji jest biblioteka *UłamekBiblioteka.dll*, którą znajdziemy w podkatalogu *UłamekDemo\UłamekBiblioteka\bin\Debug* (jeżeli kompilujemy projekt w trybie *Debug*).

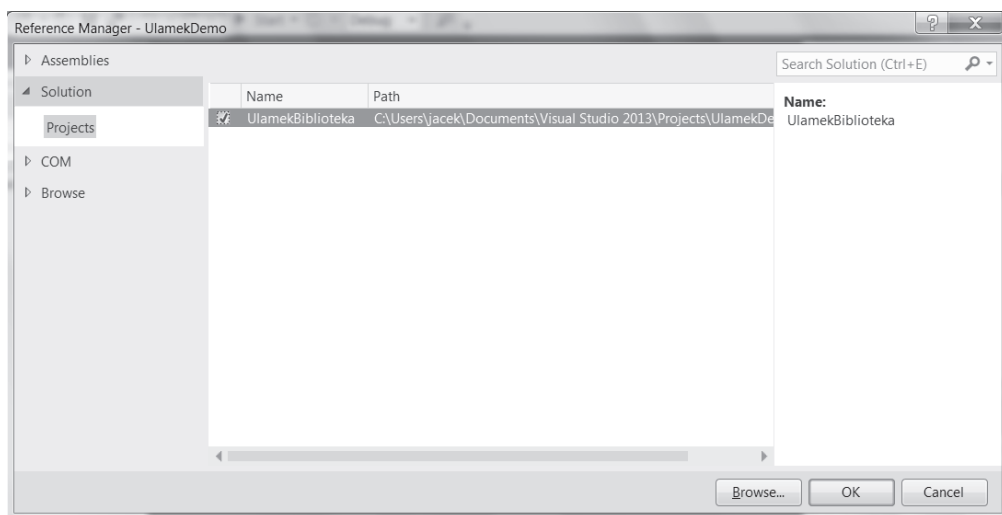
7. Jeżeli kompilacja się powiodła, możemy śmiało skasować plik *Ułamek.cs* z projektu *UłamekDemo*. Projekt nie będzie chciał się skompilować, ale niedługo to naprawimy.

Jeśli chcielibyśmy zachować konwencję nazewnictwa proponowaną w platformie .NET, nasza biblioteka powinna przejąć nazwę po przestrzeni nazw, tj. powinna nazywać się *Helion.dll*. Skoro jednak nasza biblioteka zawiera tylko jedną klasę, uważam, że większy sens ma wyeksponowanie nazwy tej klasy.

Dodawanie do aplikacji referencji do biblioteki DLL

W tej chwili projekt *UłamekDemo* nie chce się skompilować, ponieważ nie widzi klasy *Ułamek*, do której odwołuje się w statycznej metodzie *Program.Main*. Musimy podpowiedzieć kompilatorowi, żeby klasy *Ułamek* szukał w bibliotece *UłamekBiblioteka.dll*.

1. W podoknie *Solution Explorer* z menu kontekstowego projektu *UłamekDemo* wybierz pozycję *Add, Reference....*
2. Pojawi się okno *Reference Manager — UłamekDemo* (rysunek 5.1). W lewym panelu zaznaczmy pozycję *Solution*. Ogranicz to wyświetlane biblioteki do tych, które są zdefiniowane w bieżącym rozwiązaniu, a więc tylko do jednej.
3. Z lewej strony tej biblioteki na liście znajduje się słabo widoczne pole opcji, które należy zaznaczyć.
4. Następnie klikamy *OK*.



Rysunek 5.1. Dodawanie do projektu referencji do biblioteki DLL

5. Dodanie biblioteki jednak nie wystarczy. Znajduje się ona bowiem w innej przestrzeni nazw niż przestrzeń nazw projektu aplikacji (jeżeli zmieniliśmy nazwę przestrzeni nazw na *Helion*). Musimy wobec tego zadeklarować użycie tej przestrzeni nazw, dodając na początku pliku *Program.cs* instrukcję `using Helion;`.
6. Po tym oba projekty powinny się kompilować bez problemów. Klasa *Ulamek* jest dostępna i jej metody mogą być swobodnie wywoływane z metody *Program.Main*.

To najprostszy, a jednocześnie najczęstszy sposób tworzenia i użycia zarządzanych bibliotek DLL. Jest to tzw. statyczne łączenie bibliotek. Oprócz tego możliwe jest ich ładowanie dynamiczne już w trakcie działania programu. Wówczas konieczna jest czasem analiza zawartości biblioteki i rozpoznawanie umieszczonych w niej typów, weryfikowanie, czy biblioteka zawiera konkretną klasę, której chcemy użyć, i czy ta z kolei zawiera potrzebną nam metodę. Konieczne jest wówczas korzystanie z technologii odzwierciedlania typów (ang. *reflection*). Jest to odpowiednik RTTI z C++. Kolejnym zagadnieniem jest uruchomienie znalezionych metod. Wysiłek włożony w przygotowanie odpowiedniego kodu może się opłacać, np. w sytuacji, w której bibliotek DLL chcemy używać jako znajdujących już w trakcie działania programu pluginów. Wówczas jednak lepiej użyć technologii MEF (zobacz książkę polecaną w przypisie nr 2).

Przenośne biblioteki DLL

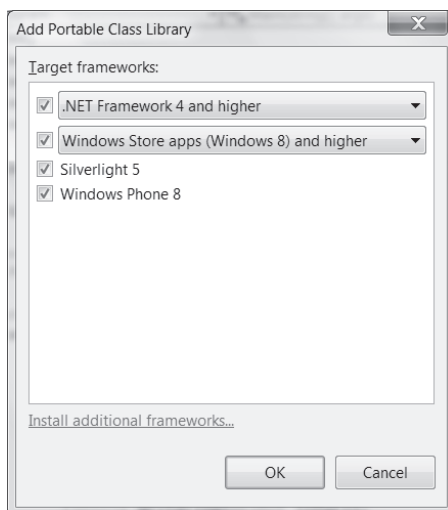
W Visual Studio 2012 pojawiła się możliwość tworzenia bibliotek przenośnych PCL (ang. *Portable Class Library*), których można używać na kilku zarządzanych platformach oferowanych przez Microsoft. Pośród nich są: platforma .NET, WinRT, XNA,

platforma instalowana w Xbox czy urządzeniach z Windows Phone. Ma to ogromną zaletę — raz przygotowany kod nie musi być modyfikowany w siostrzanych projektach dla różnego typu urządzeń. A tym samym ułatwia zarządzanie zmianami, które są zmorem projektów wieloplatformowych.

Tworzenie tego typu bibliotek nie różni się zbytnio od tworzenia zwykłej biblioteki klas:

1. W podoknie *Solution Explorer* zaznaczmy pozycję *UlamekDemo* odpowiadającą całemu rozwiązaniu (nie projektowi aplikacji).
2. Z jej menu kontekstowego wybierzmy *Add*, a następnie *New Project...*
3. Pojawi się znajome okno *Add New Project*. Zaznaczmy w nim pozycję *Portable Class Library*.
4. W polu *Name* wpisujemy *UlamekBibliotekaPrzenosna* i klikamy *OK*.
5. Zanim utworzony zostanie nowy projekt, zostaniemy spytani o to, z jakimi platformami nowa biblioteka ma być zgodna (rysunek 5.2). Ja obniżyłem tylko wymóg co do wersji platformy .NET do wersji 4. Dzięki temu uzyskaliśmy zgodność wsteczną z Visual Studio 2010.

Rysunek 5.2.
*Wybór platform,
z którymi nowa
biblioteka
ma być zgodna*



6. Następnie klikamy *OK*. Powstanie biblioteka PCL.

Dalsze postępowanie jest identyczne jak w przypadku zwykłej biblioteki, tj. kasujemy plik *Class1.cs*, a w zamian kopiujemy do nowego projektu plik *Ulamek.cs*. Dołączanie referencji do bibliotek przenośnych PCL również odbywa się identycznie jak zwykłych bibliotek DLL. Jedyna różnica polega na tym, że tak utworzonej biblioteki PCL możemy użyć nie tylko w aplikacjach konsolowych, Windows Forms, WPF lub aplikacjach internetowych ASP.NET, ale również w aplikacjach dla Windows 8 na ekran Start (tzw. aplikacje Windows Store) lub w aplikacjach dla smartfonów z systemem Windows Phone.

**Wskazówka**

Oprócz platform, które widoczne są na rysunku 5.2, możliwe jest także zainstalowanie dodatkowych. Zobaczmy je, klikając łącze *Install additional frameworks...* widoczne w oknie *Add Portable Class Library*. Są to m.in. Windows Azure, Xbox czy Kinect for Windows.

Skorowidz

A

ADO.NET, 267, 307, 309
algorytm Euklidesa, 101
alias, 115
aplikacja
 błąd, 27
 debugowanie, 27, 30, 31
 obsługa wyjątków, 33
Entity Framework, 162
ikona w zasobniku, 217, 219
inicjacja asynchroniczna, 332
interfejs, *Patrz:* interfejs
 aplikacji
konsolowa, 15, 26, 76
menu
 Edycja, 204
 główne, 194
 Plik, 194, 196, 202, 205
 Widok, 203
projekt, 190
punkt wejściowy, 17
środowisko, 23
tryb pojedynczego wątku, 181
uruchamianie
 bez debugowania, 18
 breakpoint, 30
 do kursora, 30
 obsługa wyjątków, 35
 w trybie śledzenia, 28
 z debugowaniem, 19
ustawienia, 222, 223, 224
Windows Forms, 26, 76, 171,
 176, 257
 interfejs, *Patrz:* interfejs
 aplikacji
 Windows Forms
Windows Phone, 39, 141

Windows Store, 39, 40,
 162, 165
WPF, 26, 39
z bazą danych SQL Server,
 270
zamykanie, 196, 197
auto-implemented properties,
 Patrz: właściwość domyślnie
 implementowana

B

balloon, *Patrz:* dymek
baza danych, 40, 267, 323
 aktualizacja, 283
 Microsoft Access, 309
 obiektowa, 267
 odzworowanie
 obiektowo-relacyjne,
 Patrz: ORM
 w klasie .NET, 267
rekord, 285
SQL Server, 267, 271, 279,
 309
SQL Server Compact, 279
biblioteka, 72
ASP.NET, 193
DLL, 137, 144
 przenośna, *Patrz:*
 biblioteka PCL
 referencja, 139
Entity Framework, 193, 326
EntityFramework.SqlServer.d
 ll, 326
Forms.dll, 21
kontrolki, 171
łączenie
 dynamiczne, 140
 statyczne, 137, 140
PCL, 137, 140, 144

STL, 111
System.Data.Linq.dll, 280
TPL, 159
WCF, 193
Windows Forms, 171, 193
 kontrolka, 189
WPF, 171, 193
biblioteka DLL, 138
blok, 79
boxing, *Patrz:* pudełkowanie
buforowanie podwójne, 227

C

callback, *Patrz:* funkcja zwrotna
Caller Information, 93
CAS, 43
CIL, 41, 43
CLR, 39, 40, 43
CLS, 43
Code Access Security, *Patrz:*
 CAS
Common Intermediate
 Language, *Patrz:* CIL
Common Language Runtime,
 Patrz: CLR
Common Language
 Specification, *Patrz:* CLS
Common Type System, *Patrz:*
 CTS
connection string, *Patrz:*
 łańcuch konfiguracyjny
 połączenie, *Patrz:* łańcuch
 połączenia
CTS, 43
czcionka, 203

D

dane
 baza, *Patrz:* baza danych
 filtrowanie, 302, 319
 łączenie zbiorów, 252
 modyfikacja, 276, 295
 pobieranie, 276, 283
 sortowanie, 301, 319
 typ, *Patrz:* zmienna typ
 źródło, 298, 299, 304, 307, 334, 336
 kreator, 297
 data source, *Patrz:* dane źródło
 delegacja, 61, 64, 95, 118
 DLR, 40, 43
 drag & drop, *Patrz:*
 przeciągnij i upuść
 drukowanie, 205, 208, 209
 długich linii, 210, 212
 w tle, 213
 drzewo, 81
 dymek, 219
 Dynamic Language Runtime,
Patrz: DLR
 dyrektywa
 #define, 79
 #endregion, 79, 80
 #if, 78
 #region, 79, 80
 dyrektywa preprocesora, 77, 78
 dziedziczenie, 117, 120, 122, 129, 131
 wielokrotne, 134

E

edytor
 Create Unit Test, 144
 kodu, 17, 80
 O/R Designer, 281, 286, 287
 relacji, 292
 EF, *Patrz:* Entity Framework
 ekran powitalny, 214, 215
 entity class, *Patrz:* klasa encji
 Entity Framework, 268, 307, 323, 326, 341
 entry point, *Patrz:* aplikacja
 punkt wejściowy
 Euklidesa algorytm, *Patrz:*
 algorytm Euklidesa
 exception, *Patrz:* wyjątek
 extension method, *Patrz:*
 rozszerzenie

F

FIFO, 88
 FILO, 88
 formularz, 302, 316
 funkcja, 56
 Average, 250
 GetSystemMetrics, 238
 haszująca, 108
 Max, 250
 Min, 250
 Sum, 250
 trygonometryczna, 159
 zwrotna, 64

G

garbage collector, *Patrz:*
 odśmiecacz
 generator liczb pseudolosowych,
 73
 generic types, *Patrz:* zmienna
 typ ogólny
 graphical user interface, *Patrz:*
 GUI
 GUI, 171, 231

H

Hejlsberg Anders, 39, 268

I

indeksator, 95
 inheritance, *Patrz:* dziedziczenie
 instrukcja
 break, 74, 101
 continue, 74
 DELETE, 270
 INSERT, 270
 return, 76
 SELECT, 269
 using
 System.Data.Linq.Mapping,
 280
 warunkowa
 if..else, 72
 wyboru switch, 73
 IntelliSense, 18, 72, 123, 331
 tryb, 18
 interfejs, 95, 111, 134, 135
 aplikacji, 189, 190
 kontrolka, *Patrz:* kontrolka
 Windows Forms, 171, 172

graficzny użytkownika,
Patrz: GUI
 IComparable, 84, 109, 114
 IConvertible, 100
 IDictionary, 87
 IDirectory, 25
 IEnumerable, 247, 248, 250
 implementacja, 109, 131
 przez typ ogólny, 114
 master-details, 334
 Modern UI, 165
 tworzenie, 338
 użytkownika, 298, 300

J

Java, 39
 język
 C#, 17, 39, 40, 95
 wielkość liter, 18
 C++, 17, 39
 dynamiczny, 40
 Java, *Patrz:* Java
 Python, 40
 Ruby, 40
 SQL, *Patrz:* SQL
 Transact SQL, *Patrz:* język
 T-SQL
 T-SQL, 269, 279
 XML, 255
 atrybut, 256
 deklaracja, 255, 258
 dokument, 255
 element, 256
 komentarz, 256, 258
 zapytań, *Patrz:* LINQ
 JIT, 43
 Just-In-Time, *Patrz:* JIT

K

catalog
 domowy, 25
 specjalny, 24
 klasa, 15, 66, 67
 abstrakcyjna, 124, 125, 127, 135
 Array, 65, 81
 bazowa, 98, 100, 112, 113, 115, 120, 123, 127, 134
 bazująca na typie, 110
 BindingSource, 301

DataContext, 268, 279, 281,
 286, 294
 DataSet, 307, 309
 definiowanie, 95
 encji, 279, 280, 281, 286,
 287, 293, 323
 Enum, 54
 Environment, 23, 24
 Graphics, 205, 225
 HttpClient, 165
 instancja, *Patrz:* obiekt
 Lazy, 55
 List, 81
 MessageBox, 21
 opakowująca, 45
 ORM, 287, 309, 323
 Panel, 175
 Parallel, 160
 ParallelLoopState, 161
 pole, *Patrz:* pole
 potomna, 117, 120, 123, 127
 PrivateObject, 147
 Queue, 87
 Random, 73
 SoundPlayer, 221
 Stack, 87
 statyczna, 131
 dziedziczenie, 131
 instancja, 131
 StorageFile, 165
 StreamReader, 165
 StreamWriter, 165
 String, 51, 52
 StringBuilder, 53
 SystemColor, 189
 Task, 159
 Trace, 93
 TrackBar, 175
 WCF, 165
 właściwość, *Patrz:*
 właściwość
 XDocument, 258
 XmlReader, 165
 klawiatura, 187
 odczytywanie danych, 20
 klawisz, 20
 Alt, 187
 Ctrl, 187
 Esc, 187
 F10, 28, 29
 F11, 28, 29
 F4, 174
 F5, 19, 184
 F7, 176

F9, 30
 Shift, 175, 187
 specjalny, 187
 Tab, 180
 klawisze skrótów
 debugera, 29
 edytora, 19
 klucz — wartość, 87
 kod
 maszynowy, 43
 oparty na refleksji, 72
 pośredni, *Patrz:* CIL
 źródłowy, 79, 80
 kolejka, 81, 87, 88
 kolekcja, 44, 75, 81, 85
 Dictionary, 87
 List, 85
 SortedDictionary, 87
 SortedList, 87, 88
 kompilacja, 80
 atrybut, 80
 dwustopniowa, 40
 warunkowa, 78, 79
 kompilator, 40, 43
 jako usługa, 40
 komponent, *Patrz:* kontrolka
 konstruktor, 129, 177
 bezargumentowy, 98, 130
 domyślny, 98, 104, 130, 132
 prywatny, 132
 kontrolka, 175, 189, 290
 ComboBox, 302, 303
 DataGridView, 283, 290,
 291, 300, 311, 330
 DataSet, 267, 307
 Label, 214
 panel, 175
 suwak, 175
 TextBox, 302
 TreeView, 261
 zdarzenie domyślne, 196
 kreator
 modelu danych EDM, 327
 źródła danych, 297
 kursor myszy, 231, 234, 236
 kształt, 235, 236
 położenie, 238

L

Language Integrated Query,
 Patrz: LINQ, zapytanie LINQ
 LINQ, 247, 248, 249
 LINQ to Entities, 329

LINQ to SQL, 268, 279, 280,
 281, 282, 284, 285, 294,
 297, 307
 LINQ to XML, 257, 259
 Linux, 40
 lista, 81, 85
 dwukolumnowa, 87
 dysków logicznych, 26
 szablonów, 100
 z kluczem, 88
 literał liczbowy,
 Patrz: stała liczbowa

Ł

łańcuch, 51, 53
 konfigurujący połączenie, 308
 połączenia, 271
 znaków, 20, 199

M

makro, 78
 metoda, 44, 95
 abstrakcyjna, 125, 127
 anonimowa, 64, 65
 Append, 54
 argument, 57, *Patrz też:*
 metoda parametr
 asynchroniczna, 165
 atrybut
 ClassCleanup, 150
 ClassInitialize, 150
 TestCleanup, 150
 TestInitialize, 150
 Break, 161
 CompareTo, 84
 Console, 18, 21
 CreateDatabase, 286
 DeleteDatabase, 286
 DoDragDrop, 231, 233, 234,
 236, 238
 Equals, 106
 ExecuteCommand, 286
 GetEnvironmentVariable, 25
 GetEnvironmentVariables, 25
 GetHashCode, 106, 107
 GetValueOrDefault, 67
 głowa, 57
 Insert, 53
 LoadAsync, 332
 MessageBox, 200
 Min, 65

metoda
 nadpisywanie, 123, 126,
 127, 180
 OrderBy, 248
 Parallel.For, 161
 parametr, 57, 113, *Patrz też:*
 metoda argument
 nazwany, 59
 opcjonalny, 58
 tablica, 89
 typ referencyjny, 58, 60
 typ wartościowy, 58, 60
 wartość domyślna, 46, 58
 zwracana wartość, 59
 PrivateObject, 147
 przeciągnij i upuść, *Patrz:*
 przeciągnij i upuść
 przeciążanie, 57
 przesłanianie, 124
 Read, 20
 ReadKey, 20
 ReadLine, 20
 Remove, 53
 Replace, 53
 rozszerzająca, 65, 117, 118,
 248, 250, 332
 SaveChangesAsync, 333
 Select, 248
 SetError, 21
 SetIn, 21
 SetOut, 21
 Show, 21
 Single, 250
 Sort, 84
 statyczna, 17, 18, 56, 99
 Stop, 161
 SubmitChanges, 283, 294
 sygnatura, 57
 Where, 248
 witrualna, 126, 127
 WriteLine, 20, 21
 wywołanie, 93
 zdarzeniowa, 95, 182, 183,
 185, 196
 testowanie, 183
 wywoływanie z poziomu
 kodu, 186
 zwracana wartość, 59
 zwrotna, 166
 mock object, *Patrz:* zaślepka
 modyfikator
 async, 165
 const, 99
 event, 63

explicit, 109
 implicit, 109
 internal, 97
 new, 124
 override, 124, 126
 private, 97, 123
 protected, 97
 protected internal, 97
 public, 97, 123
 readonly, 98, 99
 sealed, 123
 static, 56, 98, 99
 virtual, 124, 127
 MSIL, *Patrz:* CIL

N

nadawca, 186
 największy wspólny dzielnik, 101
 namespace, *Patrz:* przestrzeń
 nazw
 nHibernate, 267, 307
 NUnit, 143

O

obiekt, 66, 95
 anonimowy, 249
 formy, 173
 inicjacja, 92
 klonowanie, 53
 kopiowanie, 66
 sortowanie, *Patrz:* sortowanie
 zastępczy, 157
 object-relational mapping,
Patrz: ORM
 obszar powiadamiania, *Patrz:*
 zasobnik
 odświeżacz, 67, 96, 180
 odwzorowanie obiektowo-
 relacyjne, *Patrz:* ORM
 okno, 21
 aplikacji, 173
 ikona, 191
 Locals, 31
 nazwa, 191
 Properties, 174
 Toolbox, 174, 175
 Watch, 31
 własności, 174
 OLE DB, 267
 operator, 47, 48, 49
 !=, 106, 107

+=, 53, 54, 62, 63
 <, 106, 107
 <=, 106
 =, 54
 -=, 62
 ==, 106, 107
 ==>, 64
 >, 106, 107
 >=, 106
 arytmetyczny, 47, 50, 105,
 111, 154, 155
 as, 50
 await, 162, 163, 165, 166,
 167
 bitowy, 47
 from, 248
 is, 50
 join, 252, 253, 292
 konwersji, 49, 108
 LINQ, 247
 new, 66
 orderby, 248
 porównania, 106
 priorytet, 47
 przeciążanie, 105
 przypisania, 44, 53, 63
 select, 248
 where, 248
 oprogramowania testowanie,
Patrz: test
 ORM, 267, 279, 323
 overload, *Patrz:* metoda
 przeciążanie

P

pamięć
 wyciek, 67
 zarządzanie, 39
 Parallel Extensions, 159
 pasek stanu, 192, 198
 pętla, 74
 do..while, 74
 for, 74
 równoległa, 159, 161
 foreach, 75, 83
 while, 74
 platforma, 142
 .NET, 39, 40, 140
 historia, 41
 wersja, 41, 193
 wydajność, 96
 Mono, 40, 137

WinRT, 40, 140
 Xbox, 141
 XNA, 40, 140

plik
 .bmp, 214
 .dll, 41
 .emf, 214
 .exe, 41, 137
 .gif, 214
 .ico, 191
 .jpeg, 214
 .png, 214
 .wav, 220, 221
 .wmf, 214
 AssemblyInfo.cs, 80
 dźwiękowy, 220
 przenoszenie, 242
 tekstowy, 199, 202
 wybór, 200
 XML, 255, 257, 260
 modyfikowanie, 264
 przenośność, 262

pole, 95
 deklaracja, 98
 prywatne, 97, 147, 180
 statyczne, 98
 tylko do odczytu, 98

polimorfizm, 127

Portable Class Library, *Patrz:*
 biblioteka PCL, *Patrz:*
 biblioteka PCL

preprocesor
 dyrektywa, *Patrz:* dyrektywa
 preprocesora
 stała, *Patrz:* stała
 preprocesora

procedura składowana, 276, 277,
 294, 295, 327, 333
 testowanie, 294

programowanie
 asynchroniczne, 40, 162
 dynamiczne, 43
 obiektywne, 95, 119
 wizualne, 172
 współbieżne, 40, 159
 zdarzeniowe, 196

projektowanie wizualne, 308

przeciągnij i upuść, 231, 237, 238
 wiele elementów, 241

przestrzeń nazw, 17
 System.Collections, 81
 System.Collections.Specialized,
 81
 System.Data.Entities, 333

System.Data.Entity, 331
 System.Data.Linq.Mapping,
 280
 System.Entity.Data, 329
 System.Linq, 248
 System.Xml.Linq, 257

pudełkowanie, 68

Q

queue, *Patrz:* kolejka

R

relacja, 292
 jeden do wielu, 304
 ReSharper, 72
 rozszerzenie, 65, 117, 118, 248,
 250, 332

S

schowek, 204
 sender, *Patrz:* nadawca
 siatka DataGridView, *Patrz:*
 kontrolka DataGridView

singleton, 123, 131

słownik, 81, 87, 88

słowo kluczowe
 abstract, 125
 break, 74
 checked, 33, 37
 class, 97
 continue, 74
 default, 46
 delegate, 61
 dynamic, 69, 72
 event, 62, 63
 namespace, 17
 out, 61
 override, 100
 params, 89
 ref, 61
 return, 59
 struct, 66, 97
 this, 98, 118, 180
 throw, 77
 try, 75
 using, 17, 200
 var, 47, 72, 247
 void, 57
 yield, 90

sortowanie, 84

splash screen, *Patrz:* ekran
 powitalny
 SQL, 247, 269, 277

stack, *Patrz:* stos

stała, 66
 liczbowa, 46
 preprocesora, 78

stored procedure, *Patrz:*
 procedura składowana

stos, 66, 87, 88, 96
 wywołań, 33

strongly typed, 280

struktura, 66, 67, 96, 120
 bazująca na typie, 110

strumień
 błędów, 21
 przekierowanie, 21
 standardowy wyjścia, 20
 StringReader, 208

szablon, 100, 111

Ś

środowisko CLR, *Patrz:* CLR

T

tabela, 327
 aktualizacja, 283
 dodawanie do bazy danych,
 272
 edycja danych, 274
 łączenie, 292
 prezentacja w formularzu,
 302
 relacja, 310

tablica, 81, 85, 109
 jako argument metody, 89
 łańcuchów, 199
 wielowymiarowa, 84

Target Framework, 193

Task Parallel Library, *Patrz:*
 biblioteka TPL

technologia LINQ, *Patrz:* LINQ

test
 funkcjonalny aplikacji, 143
 integracyjny, 143
 jednostkowy, 143, 144, 152
 konstruktora, 147
 pola prywatnego, 147
 projekt, 144
 tworzenie, 145
 uruchamianie, 146
 wyjątków, 148

test

losowy, 151
 metody zdarzeniowej, 183
 operatorów arytmetycznych, 155
 procedury składowanej, 294
 systemowy, 143
 wydajnościowy, 143
 typ, *Patrz:* zmienna typ

U

użytkownika profil
 katalog domowy, 25
 katalog specjalny, 24

W

widok, 16, 277, 291, 327, 333
 Windows, 39
 Windows Presentation
 Foundation, *Patrz:* biblioteka
 WPF
 właściwość, 95, 102
 domyślnie implementowana, 103, 104
 wyjątek, 75
 DivideByZeroException, 50
 filtrowanie, 200
 InvalidOperationException, 109
 nieobsłużony, 76
 obsługa, 76
 OverflowException, 37
 zgłaszanie, 77
 wyrażenie lambda, 64, 65, 248, 250

Z

zapytanie
 LINQ, 39, 65, 72, 90, 248, 257, 263, 279, 329
 SQL, 248, 267, 277
 T-SQL, 279
 zintegrowane z językiem programowania, *Patrz:* LINQ, zapytanie LINQ
 zasobnik, 217
 zaślepka, 157
 zdarzenie, 62, 95, 186, 196
 domyślne, 196
 DragDrop, 231, 234
 DragEnter, 231
 DragOver, 231, 234, 235
 FormClosed, 257
 KeyPress, 187
 MouseDown, 231
 Paint, 208, 225
 zintegrowany język zapytań, *Patrz:* LINQ
 zmienna, 44
 całkowita, 44, 49
 deklaracja, 44
 globalna, 131
 inicjowanie leniwe, 55
 int, 33
 łańcuchowa, 44
 null, 67, 68
 obiektoowa, 180
 środowiskowa
 USERPROFILE, 25
 typ, 44, 45, 47
 anonimowy, 119
 Delegacja, 61
 dynamiczny, 47, 69, 71
 Graphics, 226
 konwersja, 49, 100, 108, 127
 Nullable, 67

object, 71
 ogólny, 110, 111, 116
 Panel, 180
 parametryczny, *Patrz:* zmienna typ ogólny
 referencyjny, 47, 58, 60, 66, 67, 71, 83, 96, 120, 180
 Task, 165
 wartościowy, 47, 55, 58, 60, 66, 67, 68, 83, 96, 120
 wartość domyślna, 46, 58
 wyliczeniowy, 54
 XComment, 258
 XDeclaration, 258
 znakowy, 46
 zmiennoprzecinkowa, 44, 49, 50
 znak
 !=, 106, 107
 +=, 53, 54, 62
 <, 106, 107
 <=, 106
 -=, 62
 =, 44
 ==, 106, 107
 =>, 64
 >, 106, 107
 >=, 106
 \b, 51
 backslash, *Patrz:* znak
 lewego ukośnika
 cudzysłów, 18
 końca linii, 18, 51
 lewego ukośnika, 51
 łańcuch, *Patrz:* łańcuch znaków
 \n, *Patrz:* znak końca linii
 spacji, 18
 \u, 51
 zapytania, 68

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

Visual Studio 2013

Podręcznik programowania w C# z zadaniami

Współczesne oprogramowanie musi sprostać wysokim standardom — powinno być funkcjonalne, bezawaryjne i szybkie. Pisanie i kompilowanie takich programów znacznie ułatwiają Visual Studio 2013 oraz platforma .NET. Warto wykorzystać także obiektowy język programowania C# 5.0, sprawdzone rozwiązanie zaprojektowane i wykorzystywane przez Microsoft. Ta wiedza pomoże Ci odnieść sukces na elitarnym rynku programistów Windows.

Jeśli chcesz dowiedzieć się więcej o praktycznych aspektach działania Visual Studio, platformy .NET i programowania w języku C#, koniecznie przeczytaj tę książkę! Nauczysz się, jak pisać programy w C#, jak je debugować i kompilować. Zobaczysz, do czego służy biblioteka kontrolki i gdzie możesz bezpiecznie przechowywać dane. Przede wszystkim jednak będziesz miał możliwość rozwiązania wielu zadań, bo nic nie sprzyja nauce tak bardzo, jak samodzielne zmaganie się z ciekawymi zadaniami.

- Pierwsze spotkanie ze środowiskiem Visual Studio i językiem C#
- Błędy i ich tropienie
- Język C# 5.0 i programowanie obiektowe w C#
- Biblioteki DLL, testy jednostkowe i elementy programowania współbieżnego
- Pierwszy projekt aplikacji Windows Forms i komponenty biblioteki Windows Forms
- Technologia LINQ
- Przechowywanie danych w plikach XML (LINQ to XML)
- Tworzenie i rozbudowa bazy danych SQL Server i kreator źródeł danych
- Tradycyjne ADO.NET (DataSet) i nowoczesne Entity Framework
- Przykładowe zadania

Twórz nowoczesne aplikacje w najnowszym Visual Studio i z platformą .NET!

helion.pl
księgarnia
internetowa

Nr katalogowy: 20177



Księgarnia internetowa:

<http://helion.pl>



Zamówienia telefoniczne:

0 801 339900



0 601 339900



Helion

Sprawdź najnowsze promocje:

• <http://helion.pl/promocje>

Książki najchętniej czytane:

• <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

• <http://helion.pl/novosci>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

ISBN 978-83-246-6856-4



9 788324 668564

Cena: 59,00 zł

Informatyka w najlepszym wydaniu