

Spis treści |

O autorze	17
O recenzentach	18
Przedmowa	19
Wstęp	20

CZĘŚĆ 1. Refaktoryzacja w C# w Visual Studio

ROZDZIAŁ 1

Dług techniczny, zapaszki kodu i refaktoryzacja	27
Dług techniczny i stary kod	27
Skąd się bierze dług techniczny	28
Identyfikacja zapaszków kodu	29
Wprowadzenie do refaktoryzacji	30
Narzędzia do refaktoryzacji w Visual Studio	31
Studium przypadku — linie lotnicze Cloudy Skies	32
Podsumowanie	34
Pytania	34
Dalsza lektura	35

ROZDZIAŁ 2

Wprowadzenie do refaktoryzacji	36
Wymagania techniczne	36
Refaktoryzacja kalkulatora cen bagażu	36
Konwersja własności na własności automatyczne	38
Wprowadzanie zmiennych lokalnych	39
Wprowadzanie stałych	42
Wprowadzanie parametrów	43
Usuwanie nieużywanego i nieosiągalnego kodu	45

Wyodrębnianie metod	48
Ręczna refaktoryzacja	50
Testowanie kodu po refaktoryzacji	51
Refaktoryzacja w innych edytorach	52
Refaktoryzacja w Visual Studio Code z dodatkiem C# Dev Kit	52
Refaktoryzacja w środowisku JetBrains Rider	53
Refaktoryzacja w Visual Studio z dodatkiem ReSharper	54
Podsumowanie	54
Pytania	55
Dalsza lektura	56

ROZDZIAŁ 3

Refaktoryzacja przepływu sterowania i iteracji	57
Wymagania techniczne	57
Refaktoryzacja aplikacji do obsługi przyjmowania na pokład	57
Kontrola przepływu sterowania	58
Odwracanie instrukcji if	60
Opuszczanie instrukcji else po instrukcjach return	61
Restrukturyzacja instrukcji if	62
Operator trójargumentowy	64
Zamiana instrukcji if na instrukcje switch	66
Konwersja na wyrażenia switch	70
Tworzenie obiektów	72
Zamiana var na konkretne określenia typów	72
Prostsze tworzenie przy użyciu słowa kluczowego new z określeniem typu docelowego	74
Inicjalizatory obiektów	75
Iterowanie kolekcji	76
Pętla foreach	77
Konwersja na pętlę for	79
Konwersja na LINQ	79
Refaktoryzacja instrukcji LINQ	81
Wybór odpowiedniej metody LINQ	81
Łączenie metod LINQ	83
Przekształcanie przy użyciu metody Select	84
Przegląd i testowanie kodu po refaktoryzacji	85
Podsumowanie	86
Pytania	87
Dalsza lektura	87

ROZDZIAŁ 4

Refaktoryzacja na poziomie metod 88

- Wymagania techniczne 88
- Refaktoryzacja rejestratora lotów 88
- Refaktoryzacja metod 90
 - Zmiana modyfikatorów dostępu do metod 90
 - Zmienianie nazw metod i parametrów 91
 - Przeciążanie metod 93
 - Łącuchy wywołań metod 94
- Refaktoryzacja konstruktorów 95
 - Generowanie konstruktorów 96
 - Łącuchy konstruktorów 98
- Refaktoryzacja parametrów 99
 - Zmiana kolejności parametrów 100
 - Dodawanie parametrów 101
 - Wprowadzanie parametrów opcjonalnych 104
 - Usuwanie parametrów 104
- Refaktoryzacja do funkcji 106
 - Składowe z wyrażeniem w treści 106
 - Przekazywanie funkcji jako parametrów z akcjami 107
 - Zwracanie danych z akcji przy użyciu struktur typu Func 110
- Wprowadzanie metod statycznych i rozszerzających 112
 - Tworzenie statycznych metod 112
 - Przenoszenie statycznych składowych do innych typów 113
 - Tworzenie metod rozszerzających 115
- Przegląd i testowanie kodu po refaktoryzacji 117
- Podsumowanie 118
- Pytania 118
- Dalsza lektura 118

ROZDZIAŁ 5

Refaktoryzacja kodu obiektowego 120

- Wymagania techniczne 120
- Refaktoryzacja systemu wyszukiwania lotów 120
- Organizowanie klas przez refaktoryzację 121
 - Przenoszenie klas do osobnych plików 122
 - Zmienianie nazw plików i klas 123
 - Zmiana przestrzeni nazw 124
 - Unikanie klas częściowych i regionów 125

Refaktoryzacja i dziedziczenie	126
Przesłanianie metody ToString	127
Generowanie metod równości	129
Wyodrębnianie klasy bazowej	133
Przenoszenie implementacji interfejsów w górę drzewa dziedziczenia	136
Kontrolowanie dziedziczenia za pomocą słowa kluczowego abstract	138
Wyrażanie intencji za pomocą słowa kluczowego abstract	138
Wprowadzanie składowych abstrakcyjnych	138
Konwersja metod abstrakcyjnych na wirtualne	141
Poprawianie hermetyzacji	142
Hermetyzacja pól	143
Pakowanie parametrów do klasy	144
Opakowywanie własności w klasę	147
Kompozycja zamiast dziedziczenia	149
Ulepszanie klas za pomocą interfejsów i polimorfizmu	151
Wyodrębnianie interfejsów	151
Domyślne implementacje interfejsów	153
Wprowadzanie polimorfizmu	154
Przegląd i testowanie zrefaktoryzowanego kodu	157
Podsumowanie	157
Pytania	158
Dalsza lektura	158

CZĘŚĆ 2. Bezpieczna refaktoryzacja

ROZDZIAŁ 6

Testy jednostkowe	161
Wymagania techniczne	161
Testowanie i testy jednostkowe	161
Typy testów i piramida testowania	162
Testy jednostkowe	164
Testowanie kodu przy użyciu xUnit	166
Tworzenie projektu testowego xUnit	166
Łączenie projektu testów xUnit z własnym projektem	168
Pierwszy test jednostkowy	169
Wzorzec Organizacja-Akcja-Asercja	170
Testy i wyjątki	173
Dodawanie kolejnych metod testowych	173

Refaktoryzacja testów jednostkowych	174
Parametryzacja testów za pomocą atrybutów Theory i InlineData	174
Inicjalizacja kodu testów za pomocą konstruktorów i pól	175
Współdzielenie kodu przez metody	178
Inne środowiska testowe	180
Środowisko NUnit	180
Środowisko testowe MSTest	181
Myślenie w kategoriach testów	182
Włączanie testów do codziennego toku pracy	182
Izolowanie zależności	183
Dobre i złe testy	184
Uwagi na temat pokrycia kodu	185
Podsumowanie	186
Pytania	187
Dalsza lektura	187

ROZDZIAŁ 7

Programowanie oparte na testach	188
Wymagania techniczne	188
Czym jest programowanie oparte na testach	188
Programowanie oparte na testach w Visual Studio	190
Ustawianie salda początkowego	191
Dodawanie kilometrów i generowanie metod	195
Wykorzystywanie kilometrów i refaktoryzacja testów	197
Kiedy stosować metodykę TDD	200
Podsumowanie	200
Pytania	201
Dalsza lektura	201

ROZDZIAŁ 8

Unikanie antywzorców dzięki zasadom SOLID	202
Identyfikacja antywzorców w kodzie C#	202
Przestrzeganie zasad SOLID	204
Zasada pojedynczej odpowiedzialności	204
Zasada otwarty-zamknięty	206
Zasada zastępowania Liskov	208
Zasada segregacji interfejsów	209
Zasada odwrócenia zależności	211

Inne zasady architektoniczne	212
Zasada DRY	212
Zasada KISS	214
Wysoka spójność i niski stopień sprzężenia	214
Podsumowanie	215
Pytania	216
Dalsza lektura	216

ROZDZIAŁ 9

Zaawansowane testy jednostkowe	217
Wymagania techniczne	217
Tworzenie czytelnych testów przy użyciu Shouldly	218
Instalowanie pakietu NuGet Shouldly	218
Czytelne asercje z Shouldly	219
Czytelne asercje z FluentAssertions	222
Testowanie wydajności z Shouldly	223
Generowanie danych testowych przy użyciu biblioteki Bogus	225
Imitowanie zależności za pomocą bibliotek Moq i NSubstitute	229
Dlaczego należy korzystać z bibliotek imitacyjnych	229
Tworzenie atrap obiektów przy użyciu biblioteki Moq	231
Programowanie wartości zwrotnych biblioteki Moq	232
Weryfikacja wywołań Moq	233
Tworzenie atrap przy użyciu biblioteki NSubstitute	234
Testy migawkowe z biblioteką Snapper	235
Eksperymentowanie z Scientist .NET	237
Podsumowanie	239
Pytania	240
Dalsza lektura	240

ROZDZIAŁ 10

Defensywne techniki pisanie kodu	241
Wymagania techniczne	241
API Cloudy Skies	241
Sprawdzanie poprawności danych wejściowych	242
Podstawowa weryfikacja poprawności danych	243
Słowo kluczowe nameof	245
Weryfikacja przy użyciu klauzul ochronnych	245
Klauzule ochronne z biblioteki GuardClauses	246
Atrybuty informacyjne	247

Ochrona przed null 249

 Włączanie analizy dopuszczalności wartości null w C# 250

 Operatory dopuszczalności wartości null 251

Poza granicą klas 252

 Preferowanie klas niezmiennych 252

 Własności wymagane i tylko do inicjalizacji 254

 Konstruktory podstawowe 255

 Konwertowanie klas na rekordy 256

 Klonowanie obiektów przy użyciu wyrażeń with 258

Zaawansowane techniki pracy z typami 258

 Dopasowywanie wzorców 258

 Ograniczanie dublowania za pomocą typów generycznych 260

 Tworzenie aliasów typów za pomocą dyrektywy using 262

Podsumowanie 263

Pytania 263

Dalsza lektura 264

CZĘŚĆ 3. Zaawansowana refaktoryzacja przy użyciu sztucznej inteligencji i analizy kodu

ROZDZIAŁ 11

Refaktoryzacja wspomagana przez sztuczną inteligencję

z GitHub Copilot 267

 Wymagania techniczne 267

 Wprowadzenie do GitHub Copilot 268

 Model predykcyjny GitHub 268

 Rozpoczynanie rozmowy z czatem GitHub Copilot 270

 Rozpoczynanie pracy z GitHub Copilot w Visual Studio 272

 Instalowanie i aktywacja rozszerzenia GitHub Copilot 272

 Uzyskiwanie dostępu do narzędzia GitHub Copilot 273

 Generowanie sugestii przez GitHub Copilot 274

 Interakcja z czatem GitHub Copilot 274

 Refaktoryzacja przy użyciu czatu GitHub Copilot 277

 Czat GitHub Copilot jako recenzent kodu 279

 Refaktoryzacja celowana przy użyciu czatu GitHub Copilot 279

 Tworzenie wstępnej dokumentacji za pomocą czatu GitHub Copilot 282

 Generowanie imitacji obiektów za pomocą czatu GitHub Copilot 284

Ograniczenia narzędzia GitHub Copilot	287
Prywatność danych a GitHub Copilot	287
Wątpliwości związane z kodem publicznym i narzędziem GitHub Copilot	288
Studium przypadku — linie lotnicze Cloudy Skies	289
Podsumowanie	290
Pytania	290
Dalsza lektura	291

ROZDZIAŁ 12

Analiza kodu w Visual Studio	292
Wymagania techniczne	292
Obliczanie metryk kodu w Visual Studio	292
Analiza kodu w Visual Studio	296
Analiza rozwiązania przy użyciu domyślnego zestawu reguł	296
Konfigurowanie zestawów zasad analizy kodu	299
Reagowanie na reguły analizy kodu	300
Traktowanie ostrzeżeń jako błędów	304
Zaawansowane narzędzia do analizy kodu	305
Śledzenie metryk kodu za pomocą SonarCloud i SonarQube	305
Dogłębna analiza na platformie .NET za pomocą NDepend	307
Studium przypadku linii lotniczych Cloudy Skies	311
Podsumowanie	311
Pytania	312
Dalsza lektura	312

ROZDZIAŁ 13

Tworzenie analizatora Roslyn	313
Wymagania techniczne	313
Analizatory Roslyn — informacje podstawowe	314
Instalowanie narzędzi do tworzenia rozszerzeń i edytora DGML	314
Wprowadzenie do wizualizatora składni	316
Tworzenie analizatora Roslyn	317
Dodawanie projektu analizatora do rozwiązania	317
Definiowanie zasady analizy kodu	320
Analizowanie symboli przez analizator Roslyn	322
Wskazówki na temat pisania analizatorów Roslyn	323
Testowanie analizatorów Roslyn za pomocą RoslynTestKit	324
Dodawanie projektu testowego analizatora Roslyn	324
Klasa AnalyzerTestFixture	325

Sprawdzanie, czy analizator Roslyn nie oznacza poprawnego kodu 326
 Sprawdzanie, czy analizator Roslyn znajduje niepoprawny kod 326
 Debugowanie analizatorów Analyzers 327
 Udostępnianie analizatorów jako rozszerzeń do Visual Studio 328
 Tworzenie rozszerzenia Visual Studio (VSIX) dla analizatora Roslyn 328
 Podsumowanie 331
 Pytania 331
 Dalsza lektura 332

ROZDZIAŁ 14

Refaktoryzacja kodu z analizatorami Roslyn 333
 Wymagania techniczne 333
 Studium przypadku — linie lotnicze Cloudy Skies 333
 Budowa poprawki kodu analizatora Roslyn 334
 Tworzenie dostawcy poprawki kodu 334
 Rejestrowanie poprawki kodu 336
 Modyfikowanie dokumentu przez poprawkę kodu 337
 Testowanie poprawek kodu za pomocą RoslynTestKit 339
 Publikowanie analizatorów Roslyn jako pakietów NuGet 341
 Wdrażanie za pomocą pakietów NuGet 341
 Tworzenie pakietu NuGet 342
 Wdrażanie pakietu NuGet 345
 Dodawanie pakietu NuGet 347
 Pakowanie dostawcy poprawki kodu jako rozszerzenia 348
 Podsumowanie 349
 Pytania 349
 Dalsza lektura 350

CZĘŚĆ 4. Refaktoryzacja w firmie

ROZDZIAŁ 15

Informowanie o długu technicznym 353
 Pokonywanie barier w refaktoryzacji 353
 Pilne terminy 354
 „Nie ruszać kodu wysokiego ryzyka” 355
 „Ten kod zniknie, nie marnuj na niego czasu” 355
 Aplikacje kończące cykl życia 357
 „Zrób tylko to, co jest wymagane” 358
 „Refaktoryzacja nie daje wartości biznesowej” 359

Informowanie o długu technicznym	359
Dług techniczny jako ryzyko	359
Tworzenie rejestru ryzyka	360
Co zamiast rejestru ryzyka	361
Ustalanie priorytetów długu technicznego	362
Obliczanie priorytetów ryzyka za pomocą oceny ryzyka	362
Podejście oparte na przeczuciu	363
Uzyskiwanie poparcia organizacyjnego	363
Przygotowywanie się do rozmowy	364
Przewidywanie pytań i zastrzeżeń	365
Różne podejścia dla różnych liderów	366
Znaczenie komunikacji	366
Studium przypadku — linie lotnicze Cloudy Skies	367
Podsumowanie	369
Pytania	370
Dalsza lektura	370

ROZDZIAŁ 16

Wdrażanie standardów kodowania	371
Wymagania techniczne	371
Czym są standardy kodowania	371
Znaczenie standardów kodowania	372
Jak standardy kodu wpływają na refaktoryzację	372
Stosowanie standardów kodowania do istniejącego kodu	373
Ustanowienie standardów kodowania	373
Zbiorowe standardy kodowania	374
Wybór tego, co ważne	374
Źródła standardów kodowania	375
Ewolucja standardów kodowania	376
Włączanie standardów do procesów	377
Formatowanie i czyszczenie kodu w Visual Studio	378
Formatowanie dokumentów	378
Automatyczne formatowanie dokumentów	379
Konfigurowanie ustawień stylu kodowania	380
Stosowanie standardów kodowania za pomocą plików EditorConfig	382
Kod początkowy do przeglądu	382
Dodawanie pliku EditorConfig	383
Dostosowywanie pliku EditorConfig	384

Podsumowanie	386
Pytania	386
Więcej informacji	386

ROZDZIAŁ 17

Zwinna refaktoryzacja 388

Refaktoryzacja w zwinnym środowisku	388
Kluczowe elementy zwinnych zespołów	389
Czynniki przeszkadzające w refaktoryzacji	390
Strategie dające sukces w zwinnej refaktoryzacji	390
Zadania poświęcone refaktoryzacji	390
Refaktoryzacja kodu, który jest zmieniany	392
Sprinty refaktoryzacji	392
Urlopy refaktoryzacyjne	393
Wykonywanie refaktoryzacji na dużą skalę	394
Dlaczego duże refaktoryzacje są trudne	394
Pułapka przepisywania	395
Lekcje z okrętu Tezeusza	395
Aktualizowanie projektów za pomocą .NET Upgrade Assistant	396
Refaktoryzacja i wzorzec dusiciela	398
Odzyskiwanie sprawności, gdy refaktoryzacja pójdzie źle	399
Wpływ nieudanych refaktoryzacji	400
Zapewnienie bezpieczeństwa w zwinnych środowiskach	400
Wdrażanie refaktoryzacji na dużą skalę	401
Korzystanie z flag funkcji	402
Wdrożenia etapowe lub niebiesko-zielone	402
Wartość ciągłej integracji i ciągłego dostarczania	404
Studium przypadku — linie lotnicze Cloudy Skies	405
Podsumowanie	406
W kierunku bardziej zrównoważonego oprogramowania	406
Pytania	407
Więcej informacji	407