

Spis treści

O autorze	13
O recenzencie	15
Przedmowa	17
Rozdział 1. Standardy i zasady kodowania w języku C#	23
Wymagania techniczne	24
Dobry kod kontra zły kod	24
Zły kod	25
Dobry kod	39
Potrzeba stosowania standardów kodowania, zasad i metodologii	44
Standardy kodowania	44
Zasady kodowania	45
Metodologie kodowania	45
Konwencje kodowania	46
Modułowość	46
KISS	47
YAGNI	47
DRY	48
SOLID	48
Brzytwa Ockhama	49
Podsumowanie	49
Pytania	50
Dalsza lektura	50

Rozdział 2. Przeglądy kodu — procedura i znaczenie	51
Procedura przeglądu kodu	52
Przygotowanie kodu do przeglądu	52
Kierowanie przeglądem kodu	54
Wydawanie żądania ściągnięcia	55
Odpowiadanie na żądanie ściągnięcia	58
Wpływ komentarzy udzielanych podczas przeglądu kodu na programistów przekazujących kod do przeglądu	59
Co należy przejrzeć?	62
Obowiązujące w firmie wytyczne dotyczące kodowania oraz wymagania biznesowe	62
Konwencje nazewnictwa	63
Formatowanie	63
Testowanie	64
Wytyczne dotyczące architektury i wzorce projektowe	65
Wydajność i bezpieczeństwo	66
Kiedy przestać kod do przeglądu?	67
Komentowanie przeglądane kodu i udzielanie odpowiedzi na uwagi	68
Komentowanie recenzowanego kodu	69
Odpowiadanie na komentarze recenzenta	70
Podsumowanie	71
Pytania	71
Dalsza lektura	72
Rozdział 3. Klasy, obiekty i struktury danych	73
Wymagania techniczne	74
Organizowanie klas	74
Klasa powinna mieć tylko jedną odpowiedzialność	76
Wprowadzanie w klasach komentarzy w celu generowania dokumentacji	78
Spójność i sprzężenia	81
Przykład ścisłego sprzężenia	81
Przykład luźnego sprzężenia	82
Przykład kodu o niskiej spójności	84
Przykład kodu o wysokiej spójności	84
Projektowanie z myślą o zmianach	85
Programowanie na bazie interfejsów	86
Wstrzykiwanie zależności i odwracanie sterowania	88
Przykład mechanizmu DI	89
Przykład IoC	91
Prawo Demeter	92
Przykłady stosowania i łamania prawa Demeter	92
Niemutowalne obiekty i struktury danych	94
Przykład niemutowalnego typu	94
Obiekty powinny ukrywać dane i eksponować metody	95
Przykład hermetyzacji	96

Struktury danych powinny eksponować dane i nie powinny mieć metod	96
Przykład struktury danych	97
Podsumowanie	97
Pytania	98
Dalsza lektura	99
Rozdział 4. Pisanie czystych funkcji	101
Podstawy programowania funkcyjnego	102
Pisanie krótkich metod	105
Wcięcia w kodzie	107
Unikanie powielania kodu	108
Unikanie zbyt dużej liczby parametrów	109
Implementacja reguły SRP	110
Podsumowanie	115
Pytania	115
Dalsza lektura	116
Rozdział 5. Obsługa wyjątków	117
Wyjątki sprawdzane i niesprawdzone	118
Unikanie wyjątków <code>NullPointerException</code>	121
Wyjątki dotyczące reguł biznesowych	124
Przykład 1. — obsługa warunków	127
za pomocą wyjątków opisujących reguły biznesowe	127
Przykład 2. — obsługa warunków	128
z wykorzystaniem normalnego przepływu programu	128
Przekazywanie sensownych informacji za pomocą wyjątków	130
Budowanie niestandardowych wyjątków	131
Podsumowanie	134
Pytania	134
Dalsza lektura	135
Rozdział 6. Testy jednostkowe	137
Wymagania techniczne	138
Znaczenie dobrego testu	138
Narzędzia testowe	143
MSTest	144
JUnit	151
Moq	157
SpecFlow	162
Praktyka metodologii TDD — test nie przechodzi, test przechodzi i refaktoryzacja	166
Usuwanie nadmiarowych testów, komentarzy i martwego kodu	172
Podsumowanie	173
Pytania	174
Dalsza lektura	174

Rozdział 7. Testowanie systemu „od końca do końca”	175
Testowanie E2E	175
Moduł logowania (podsystem)	177
Moduł administratora (podsystem)	180
Moduł sprawdzianów (podsystem)	182
Testowanie E2E trójmodułowego systemu	183
Fabryki	186
Wstrzykiwanie zależności	193
Modularyzacja	198
Podsumowanie	200
Pytania	201
Dalsza lektura	201
Rozdział 8. Wątki i współbieżność	203
Cykl życia wątku	204
Dodawanie parametrów wątku	205
Korzystanie z puli wątków	207
Biblioteka TPL	207
ThreadPool.QueueUserWorkItem()	210
Korzystanie z muteksów dla wątków synchronicznych	210
Praca z wątkami równoległymi z wykorzystaniem semaforów	212
Ograniczanie liczby procesorów i wątków w puli wątków	215
Zapobieganie zakleszczeniom	216
Przykład zakleszczenia	217
Zapobieganie wyścigom	221
Statyczne konstruktory i metody	224
Dodawanie statycznych konstruktorów do kodu	225
Dodawanie metod statycznych	226
Mutowalność, niemutowalność i bezpieczeństwo wątków	229
Pisanie kodu, który jest mutowalny, ale nie jest bezpieczny w kontekście wątków	230
Pisanie kodu, który jest niemutowalny i bezpieczny w kontekście wątków	232
Bezpieczeństwo wątków	233
Zależności metod zsynchronizowanych	237
Korzystanie z klasy Interlocked	238
Ogólne zalecenia	241
Podsumowanie	242
Pytania	243
Dalsza lektura	243
Rozdział 9. Projektowanie i tworzenie API	245
Wymagania techniczne	246
Czym jest API?	246
Proxy interfejsów API	248
Wytyczne projektowe dla interfejsów API	249
Dobrze zdefiniowane granice oprogramowania	252
Znaczenie dobrej jakości dokumentacji interfejsu API	255
Przekazywanie niemutowalnych struktur zamiast mutowalnych obiektów	257

Testowanie zewnętrznych API	260
Testowanie własnych API	261
Projektowanie API za pomocą RAML	264
Instalacja oprogramowania Atom i API Workbench firmy MuleSoft	264
Tworzenie projektu	265
Generowanie API w języku C#	
na podstawie niezależnej od języka specyfikacji w języku RAML	268
Podsumowanie	272
Pytania	272
Dalsza lektura	273
Rozdział 10. Zabezpieczanie API za pomocą kluczy API i usługi Azure Key Vault	275
Wymagania techniczne	276
Projekt API — kalendarz dywidend	276
Dostęp do Morningstar API	277
Przechowywanie klucza Morningstar API w Azure Key Vault	278
Tworzenie w Azure aplikacji webowej ASP.NET Core kalendarza dywidend	280
Publikowanie aplikacji webowej	281
Korzystanie z klucza API do zabezpieczenia interfejsu API kalendarza dywidend	286
Konfigurowanie repozytorium	286
Konfiguracja uwierzytelniania i autoryzacji	288
Testowanie zabezpieczeń z wykorzystaniem klucza API	295
Dodanie kodu kalendarza dywidend	297
Ustawianie przepustowości interfejsu API	304
Podsumowanie	308
Pytania	308
Dalsza lektura	309
Rozdział 11. Rozwiązywanie problemów przekrojowych	311
Wymagania techniczne	312
Wzorzec projektowy Dekorator	312
Wzorzec projektowy Proxy	315
AOP z wykorzystaniem PostSharp	317
Rozszerzanie frameworka aspektów	318
Rozszerzanie frameworka architektury	320
Biblioteka wielokrotnego użytku do obsługi przekrojowych problemów w projekcie	321
Buforowanie	321
Rejestrowanie w plikach	323
Logowanie	324
Obsługa wyjątków	325
Zabezpieczenia	326
Walidacja parametrów	329
Obsługa transakcji	334
Obsługa puli zasobów	334
Obsługa ustawień konfiguracji	335
Oprządkowanie	336
Podsumowanie	336
Pytania	337
Dalsza lektura	337

Rozdział 12. Narzędzia do poprawy jakości kodu	339
Wymagania techniczne	340
Definicja dobrej jakości kodu	340
Porządkowanie kodu i obliczanie jego metryk	342
Wykonywanie analizy kodu	345
Korzystanie z narzędzia Quick Action	347
Korzystanie z narzędzia JetBrains dotTrace	348
Korzystanie z narzędzia JetBrains ReSharper	352
Korzystanie z narzędzia Telerik JustDecompile	361
Podsumowanie	362
Pytania	363
Dalsza lektura	363
Rozdział 13. Refaktoryzacja kodu C# — identyfikacja zapachów kodu	365
Wymagania techniczne	366
Zapachy kodu na poziomie aplikacji	366
Ślepotą danych typu Boolean	366
Eksplozja kombinatoryczna	368
Wymyślna złożoność	369
Kępy danych	370
Komentarze-dezodoranty	370
Powielony kod	370
Utracony zamiar	371
Mutacje zmiennych	371
Rozwiązanie-dziwak	373
Chirurgia strzelby	375
Rozrzucanie rozwiązań	377
Niekontrolowane skutki uboczne	377
Zapachy kodu na poziomie klasy	378
Złożoność cyklomatyczna	378
Rozbieżna zmiana	382
Rzutowanie w dół	382
Nadmierne używanie literałów	382
Zazdrość o kod	383
Nieodpowiednia intymność	384
Nieprzystojne obnażanie	384
Rozbudowana klasa (obiekt-Bóg)	385
Klasa leniwa	385
Klasa-pośrednik	386
Klasa osierocona złożona z samych zmiennych i stałych	386
Obsesja na punkcie prymitywów	386
Odrzucony spadek	386
Spekulatywna ogólność	387
Stwierdzaj, nie pytaj	387
Tymczasowe pola	387

Zapachy na poziomie metod	387
Metoda „czarna owca”	387
Złożoność cyklomatyczna	388
Wymyślna złożoność	388
Martwy kod	388
Zbyt duża ilość zwracanych danych	388
Zazdrość o kod	388
Rozmiar identyfikatora	389
Nieodpowiednia intymność	389
Długie wiersze kodu (wiersze-Bogowie)	389
Leniwe metody	389
Długie metody (metody-Bogowie)	389
Długa lista parametrów	390
Łańcuchy komunikatów	390
Metoda-pośrednik	390
Rozwiązanie-dziwak	390
Spekulatywna ogólność	390
Podsumowanie	391
Pytania	391
Dalsza lektura	393
 Rozdział 14. Refaktoryzacja kodu C# — Implementacja wzorców projektowych	 395
Wymagania techniczne	396
Implementacja kreacyjnych wzorców projektowych	396
Implementacja wzorca Singleton	397
Implementacja wzorca Metoda wytwórcza	398
Implementacja wzorca projektowego Fabryka abstrakcyjna	399
Implementacja wzorca Prototyp	402
Implementacja wzorca projektowego Budowniczy	404
Implementacja strukturalnych wzorców projektowych	409
Implementacja wzorca projektowego Most	410
Implementacja wzorca Kompozyt	412
Implementacja wzorca projektowego Fasada	414
Implementacja wzorca projektowego Pylek	416
Przegląd behawioralnych wzorców projektowych	419
Końcowe wnioski	420
Podsumowanie	422
Pytania	423
Dalsza lektura	423
 Odpowiedzi	 425