

Spis treści

O autorze	15
O recenzentach	15
Wprowadzenie	17
CZĘŚĆ 1. Jak doszliśmy do TDD?	
ROZDZIAŁ 1.	
Dlaczego potrzebujemy TDD?	23
Pisanie złego kodu	23
Dlaczego piszemy zły kod?	25
Rozpoznawanie złego kodu	26
Złe nazwy zmiennych	26
Złe nazwy funkcji, metod i klas	27
Struktury podatne na błędy	29
Zależność i spójność	30
Obniżenie wydajności zespołu	32
Pogorszenie wyników biznesowych	33
Podsumowanie	35
Pytania i odpowiedzi	35
Polecane źródła	36
ROZDZIAŁ 2.	
TDD w służbie dobrego kodu	37
Projektowanie wysokiej jakości kodu	37
Mów to, co masz na myśli, i miej na myśli to, co mówisz	38
Zajmuj się szczegółami w prywatnych blokach kodu	39
Unikaj zbędnej złożoności	40
Wykrywanie wad projektowych	42
Analiza korzyści z pisania testów przed kodem produkcyjnym	43
Zapobieganie błędom w logice	46
Zabezpieczanie się przed przyszłymi defektami	47
Dokumentowanie kodu	48

Podsumowanie	49
Pytania i odpowiedzi	49
Polecane źródła	50
ROZDZIAŁ 3.	
Obalamy mity na temat TDD	51
„Pisanie testów mnie spowalnia”	51
Zrozumienie korzyści ze spowolnienia	51
Przełamywanie zarzutów dotyczących spowolnienia wynikającego z testów	53
„Testy nie zapobiegają wszystkim błędom”	54
Dlaczego ludzie twierdzą, że testy nie wychwytyją wszystkich błędów?	54
Przełamywanie zarzutów, że testy nie wychwytyją wszystkich błędów	55
„Skąd wiemy, że testy są poprawne?”	55
Rozumienie obaw dotyczących pisania złych testów	55
Dla pewności testujemy testy	56
„TDD to gwarancja dobrego kodu”	57
Zrozumienie problemu nadmiernych oczekiwań	57
Zarządzanie oczekiwaniami względem TDD	57
„Kod jest zbyt skomplikowany, aby go testować”	58
Rozumienie przyczyn nietestowalnego kodu	58
Nowe podejście do związku między dobrym projektem a prostymi testami	59
Radzenie sobie z odziedziczonym kodem pozbawionym testów	59
„Przed napisaniem kodu nie wiadomo, co testować”	60
Rozumienie trudności związanych z rozpoczynaniem pracy od pisania testów	60
Co zrobić, żeby nie musieć pisać kodu produkcyjnego na początku?	61
Podsumowanie	62
Pytania i odpowiedzi	62

Polecane źródła	62
CZĘŚĆ 2. Techniki TDD	
ROZDZIAŁ 4.	
Tworzymy aplikację z użyciem TDD	65
Wymagania techniczne	65
Przygotowanie środowiska programistycznego	66
Instalowanie programu IntelliJ	66
Konfiguracja projektu i bibliotek	66
Omówienie aplikacji Wordz	68
Zasady gry Wordz	68
Poznanie metodyk zwinnych	69
Czytamy historyjki użytkowników — elementy składowe planowania	70
Łączymy programowanie zwinne i TDD	72
Podsumowanie	72
Pytania i odpowiedzi	73
Polecane źródła	73
ROZDZIAŁ 5.	
Piszemy pierwszy test	74
Wymagania techniczne	74
Początek pracy z TDD — wzorzec przygotowania, działania i asercji	74
Poznajemy strukturę testu	75
Zaczynamy pracę od rezultatów	77
Podniesienie wydajności pracy	77
Cechy dobrego testu	78
Stosowanie zasad FIRST	78
Używamy jednej asercji na test	80
Decydujemy o zakresie testu jednostkowego	80
Wykrywanie częstych błędów	81
Sprawdzanie wyjątków	82
Testujemy tylko metody publiczne	83
Zachowanie hermetyzacji	83

Uczenie się na podstawie testów	84
Chaotyczne przygotowania	84
Chaotyczne wywołanie	84
Chaotyczne asercje	84
Ograniczenia testów jednostkowych	85
Pokrycie kodu — często bezużyteczny wskaźnik	85
Pisanie złych testów	86
Rozpoczęcie pracy nad aplikacją Wordz	86
Podsumowanie	91
Pytania i odpowiedzi	92
ROZDZIAŁ 6.	
Rytm pracy w TDD	93
Wymagania techniczne	93
Stosowanie cyklu czerwone, zielone i refaktoryzacja	93
Czerwone na początek	94
Prostota nade wszystko — przejście do etapu zielonego	95
Refaktoryzacja do czystego kodu	96
Piszemy kolejne testy do gry Wordz	97
Podsumowanie	108
Pytania i odpowiedzi	108
Polecane źródła	109
ROZDZIAŁ 7.	
TDD i SOLID wspierają projektowanie	110
Wymagania techniczne	111
Testowe i pozatestowe wsparcie w projektowaniu	111
Zasada jednej odpowiedzialności — proste elementy składowe	112
Zbyt wiele odpowiedzialności sprawia, że z kodem trudniej się pracuje	113
Możliwość ponownego użycia kodu	115
Prostsze utrzymanie w przyszłości	115
Przykład kodu nieprzestrzegającego zasady jednej odpowiedzialności	115

Zastosowanie zasady jednej odpowiedzialności do uproszczenia	
utrzymania kodu w przyszłości	116
Organizacja testów o jednej odpowiedzialności	118
Zasada odwrócenia zależności — ukrywanie nieistotnych szczegółów	118
Zastosowanie odwrócenia zależności do kodu rysującego kształty	120
Zasada podstawienia Liskov — wymienne obiekty	122
Aplikujemy zasadę podstawienia Liskov do kodu rysującego kształty ...	124
Zasada otwarte-zamknięte — rozszerzalny projekt	125
Dodajemy nowy typ kształtu	126
Zasada segregacji interfejsów — skuteczne abstrakcje	127
Przegląd użycia zasady segregacji interfejsów w kodzie	
rysującym kształty	128
Podsumowanie	129
Pytania i odpowiedzi	129
ROZDZIAŁ 8.	
Zamienniki testowe — zaślepki i atrapy	131
Wymagania techniczne	131
Problemy z obiektami pomocniczymi w testach	132
Wyzwanie testowania niepowtarzalnych zachowań	132
Wyzwanie testowania obsługi błędów	133
Przyczyny trudności w testowaniu	133
Cel stosowania zamienników testowych	133
Piszemy wersję produkcyjną kodu	135
Używanie zaślepek do zwracania predefiniowanych wyników	137
Kiedy używać zaślepek?	138
Używanie atrap do weryfikowania interakcji	138
Kiedy użycie zamienników testowych jest uzasadnione?	141
Nie nadużywajmy atrap	141
Nie róbmy atrap zewnętrznego kodu	141
Nie róbmy atrap obiektów reprezentujących wartość	142
Nie da się tworzyć atrap bez wstrzykiwania zależności	142

Nie testujemy atrap	143
Kiedy używać atrap?	143
Pracujemy z Mockito — popularną biblioteką do tworzenia zamienników testowych	143
Rozpoczynamy pracę z Mockito	144
Używamy Mockito do utworzenia zaślepki	144
Używamy Mockito do utworzenia atrapy	149
Zacieranie się rozróżnienia między zaślepkami a atrapami	150
Dopasowywanie argumentów — niestandardowe zachowanie zamienników testowych	151
Projektowanie kodu obsługującego błędy z użyciem testów	152
Testowanie obsługi błędów w grze Wordz	153
Podsumowanie	155
Pytania i odpowiedzi	155
Polecane źródła	156

ROZDZIAŁ 9.

Architektura heksagonalna — oddzielenie systemów zewnętrznych ...	157
Wymagania techniczne	157
Dlaczego systemy zewnętrzne są przyczyną trudności?	158
Problemy ze środowiskiem	159
Omyłkowe wywołanie prawdziwych procesów podczas wykonywania testów	159
Jakich danych powinniśmy oczekiwać?	160
Wywołania systemowe i czas systemowy	160
Wyzwania związane z usługami stron trzecich	160
Odwroćcie zależności na ratunek	161
Generalizowanie w kierunku architektury heksagonalnej	162
Przegląd elementów architektury heksagonalnej	163
Złota zasada: domena nigdy nie łączy się bezpośrednio z adapterami	167
Skąd kształt sześcioboku?	167
Tworzenie abstrakcji systemu zewnętrznego	168

Czego potrzebuje model domeny?	168
Pisanie kodu domeny	171
Co powinno się znaleźć w modelu domeny?	172
Użycie bibliotek i frameworków w modelu domeny	172
Wybór stylu programowania	172
Tworzenie testowych zamienników systemów zewnętrznych	173
Zastępowanie adapterów zamiennikami testowymi	173
Tworzenie testów jednostkowych większych jednostek	174
Testy jednostkowe całych historyjek użytkownika	175
Tworzenie abstrakcji bazy danych w aplikacji Wordz	176
Projektowanie interfejsów repozytoriów	176
Projektowanie adapterów bazy danych i generatora liczb losowych	179
Podsumowanie	180
Pytania i odpowiedzi	180
Polecane źródła	181
ROZDZIAŁ 10.	
Testy FIRST i piramida testów	182
Wymagania techniczne	182
Piramida testów	183
Testy jednostkowe zgodne z zasadami FIRST	185
Testy integracyjne	186
Co powinny obejmować testy integracyjne?	187
Testowanie adapterów bazodanowych	188
Testowanie usług sieciowych	189
Testowanie zgodności kontraktu z wymaganiami konsumenta	190
Testy przekrojowe i testy akceptacji przez użytkownika	191
Narzędzia do testów akceptacyjnych	193
Procesy CI/CD i środowiska testowe	194
Co to jest proces CI/CD?	194
Dlaczego potrzebujemy ciągłej integracji?	195
Dlaczego potrzebujemy ciągłego dostarczania?	196

Ciągłe dostarczanie czy ciągłe wdrażanie?	197
Praktyczne aspekty procesów CI/CD	197
Środowiska testowe	198
Testowanie na produkcji	200
Test integracyjny bazy danych dla gry Wordz	201
Pobieranie słowa z bazy danych	202
Podsumowanie	204
Pytania i odpowiedzi	204
Polecane źródła	205
ROZDZIAŁ 11.	
TDD jako część zapewnienia jakości	206
Metodyka TDD i jej miejsce w szerszym kontekście zapewnienia jakości	206
Rozumienie ograniczeń TDD	206
Czy to oznacza, że testy manualne nie są potrzebne?	207
Manualne testy eksploracyjne i odkrywanie nieoczekiwanego	209
Przeglądy kodu i programowanie zespołowe	211
Testowanie interfejsu i doświadczeń użytkownika	213
Testowanie interfejsu użytkownika	213
Ocena doświadczeń użytkownika	214
Testowanie bezpieczeństwa i monitorowanie utrzymania	215
Włączanie manualnych elementów w procesy CI/CD	216
Podsumowanie	218
Pytania i odpowiedzi	219
Polecane źródła	219
ROZDZIAŁ 12.	
Testy na początku, później czy nigdy?	221
Dodawanie testów na początku	221
Podejście „testy najpierw” jako narzędzie do projektowania	222
Testy stanowią wykonywalną specyfikację	223
Podejście „testy najpierw” przekłada się na wartościowe wskaźniki pokrycia kodu	224

Wystrzegajmy się określania progów pokrycia kodu	224
Nie pisz wszystkich testów na początku	225
Zaczynanie pracy od testów pomaga w ciągłym dostarczaniu	226
„Zawsze możemy napisać testy później, prawda?”	226
Dla początkujących podejście „testy później” jest łatwiejsze niż TDD	227
W podejściu z późniejszym testowaniem trudniej przetestować każdą ścieżkę	227
Podejście z późniejszym testowaniem ma mniejszy wpływ na projektowanie	228
Późniejsze testowanie może nigdy się nie wydarzyć	229
„Testy? Testy są dla ludzi, którzy nie potrafią programować!”	229
Co się dzieje, jeżeli nie testujemy w trakcie rozwoju oprogramowania?	230
Testowanie od wewnątrz na zewnątrz	231
Testowanie od zewnątrz do wewnątrz	233
Określanie granic testów dzięki architekturze heksagonalnej	234
Podejście dośrodkowe działa dobrze z modelem domeny	235
Podejście dośrodkowe współgra z adapterami	236
Historyjki użytkownika można testować w modelu domeny	237
Podsumowanie	238
Pytania i odpowiedzi	238
Polecane źródła	239
CZĘŚĆ 3. TDD w prawdziwym życiu	
ROZDZIAŁ 13.	
Tworzenie warstwy domeny	243
Wymagania techniczne	243
Rozpoczęcie nowej gry	243
Sterowane testami projektowanie rozpoczęcia nowej gry	244
Śledzenie postępu w grze	245
Triangulacja wyboru słowa	250
Granie w grę	255

Projektowanie interfejsu oceny próby	255
Triangulacja śledzenia postępu gry	257
Koniec gry	259
Reakcja na poprawną próbę	259
Triangulacja zakończenia gry ze względu na liczbę niepoprawnych prób	260
Triangulacja odpowiedzi na próbę po końcu gry	261
Przegląd projektu	263
Podsumowanie	265
Pytania i odpowiedzi	266
Polecane źródła	267
ROZDZIAŁ 14.	
Tworzenie warstwy bazodanowej	268
Wymagania techniczne	268
Instalowanie bazy danych PostgreSQL	268
Tworzenie testu integracji z bazą danych	269
Tworzenie testu bazy danych z użyciem biblioteki DBRider	269
Tworzenie kodu produkcyjnego	272
Implementacja adaptera WordRepository	275
Dostęp do bazy danych	277
Implementacja GameRepository	278
Podsumowanie	279
Pytania i odpowiedzi	279
Polecane źródła	280
ROZDZIAŁ 15.	
Tworzenie warstwy sieciowej	281
Wymagania techniczne	281
Rozpoczęcie nowej gry	282
Dodanie wymaganych bibliotek do projektu	282
Pisanie testu kończącego się niepowodzeniem	282
Utworzenie własnego serwera HTTP	284

Dodawanie ścieżek do serwera HTTP	285
Podłączenie do warstwy domeny	286
Refaktoryzacja kodu rozpoczynającego grę	288
Obsługa błędów podczas rozpoczęcia gry	289
Naprawa testów nieoczekiwanie kończących się niepowodzeniem	291
Granie w grę	292
Połączenie komponentów aplikacji w całość	297
Używanie aplikacji	298
Podsumowanie	301
Pytania i odpowiedzi	302
Polecane źródła	302