

| | |
|---|-----------|
| Słowo wstępne | 13 |
| Wstęp | 19 |
| 1. Czysty kod | 23 |
| Niech stanie się kod... | 24 |
| W poszukiwaniu doskonałego kodu... | 24 |
| Całkowity koszt bałaganu | 25 |
| Rozpoczęcie wielkiej zmiany projektu | 26 |
| Postawa | 27 |
| Największa zagadka | 28 |
| Sztuka czystego kodu? | 28 |
| Co to jest czysty kod? | 28 |
| Szkoły myślenia | 34 |
| Jesteśmy autorami | 35 |
| Zasada skautów | 36 |
| Poprzednik i zasady | 36 |
| Zakończenie | 36 |
| Bibliografia | 37 |
| 2. Znaczące nazwy | 39 |
| Wstęp | 39 |
| Używaj nazw przedstawiających intencje | 40 |
| Unikanie dezinformacji | 41 |
| Tworzenie wyraźnych różnic | 42 |
| Tworzenie nazw, które można wymówić | 43 |
| Korzystanie z nazw łatwych do wyszukania | 44 |
| Unikanie kodowania | 45 |
| Notacja węgierska | 45 |
| Przedrostki składników | 46 |
| Interfejsy i implementacje | 46 |
| Unikanie odwzorowania mentalnego | 47 |
| Nazwy klas | 47 |
| Nazwy metod | 47 |
| Nie bądź dowcipny | 48 |
| Wybieraj jedno słowo na pojęcie | 48 |
| Nie twórz kalamburów! | 49 |
| Korzystanie z nazw dziedziny rozwiązania | 49 |
| Korzystanie z nazw dziedziny problemu | 49 |
| Dodanie znaczącego kontekstu | 50 |
| Nie należy dodawać nadmiarowego kontekstu | 51 |
| Słowo końcowe | 52 |

| | |
|--|-----------|
| 3. Funkcje | 53 |
| Małe funkcje! | 56 |
| Bloki i wcięcia | 57 |
| Wykonuj jedną czynność | 57 |
| Sekcje wewnątrz funkcji | 58 |
| Jeden poziom abstrakcji w funkcji | 58 |
| Czytanie kodu od góry do dołu — zasada zstępująca | 58 |
| Instrukcje switch | 59 |
| Korzystanie z nazw opisowych | 61 |
| Argumenty funkcji | 62 |
| Często stosowane funkcje jednoargumentowe | 62 |
| Argumenty znacznikowe | 63 |
| Funkcje dwuargumentowe | 63 |
| Funkcje trzyargumentowe | 64 |
| Argumenty obiektowe | 64 |
| Listy argumentów | 65 |
| Czasowniki i słowa kluczowe | 65 |
| Unikanie efektów ubocznych | 65 |
| Argumenty wyjściowe | 66 |
| Rozdzielanie poleceń i zapytań | 67 |
| Stosowanie wyjątków zamiast zwracania kodów błędów | 67 |
| Wyodrębnienie bloków try-catch | 68 |
| Obsługa błędów jest jedną operacją | 69 |
| Przyciąganie zależności w Error.java | 69 |
| Nie powtarzaj się | 69 |
| Programowanie strukturalne | 70 |
| Jak pisać takie funkcje? | 70 |
| Zakończenie | 71 |
| SetupTearDownIncluder | 71 |
| Bibliografia | 73 |
| 4. Komentarze | 75 |
| Komentarze nie są szminką dla złego kodu | 77 |
| Czytelny kod nie wymaga komentarzy | 77 |
| Dobre komentarze | 77 |
| Komentarze prawne | 77 |
| Komentarze informacyjne | 78 |
| Wyjaśnianie zamierzeń | 78 |
| Wyjaśnianie | 79 |
| Ostrzeżenia o konsekwencjach | 80 |
| Komentarze TODO | 80 |
| Wzmocnienie | 81 |
| Komentarze Javadoc w publicznym API | 81 |
| Złe komentarze | 81 |
| Bełkot | 81 |
| Powtarzające się komentarze | 82 |
| Mylące komentarze | 84 |
| Komentarze wymagane | 85 |
| Komentarze dziennika | 85 |

| | |
|---|------------|
| Komentarze wprowadzające szum informacyjny | 86 |
| Przerażający szum | 87 |
| Nie używaj komentarzy, jeżeli można użyć funkcji lub zmiennej | 88 |
| Znaczniki pozycji | 88 |
| Komentarze w klamrach zamykających | 88 |
| Atrybuty i dopiski | 89 |
| Zakomentowany kod | 89 |
| Komentarze HTML | 90 |
| Informacje nielokalne | 91 |
| Nadmiar informacji | 91 |
| Nieoczywiste połączenia | 91 |
| Nagłówki funkcji | 92 |
| Komentarze Javadoc w niepublicznym kodzie | 92 |
| Przykład | 92 |
| Bibliografia | 95 |
| 5. Formatowanie | 97 |
| Przeznaczenie formatowania | 98 |
| Formatowanie pionowe | 98 |
| Metafora gazety | 99 |
| Pionowe odstępy pomiędzy segmentami kodu | 99 |
| Gęstość pionowa | 101 |
| Odległość pionowa | 101 |
| Uporządkowanie pionowe | 105 |
| Formatowanie poziome | 106 |
| Poziome odstępy i gęstość | 106 |
| Rozmieszczenie poziome | 107 |
| Wcięcia | 109 |
| Puste zakresy | 110 |
| Zasady zespołowe | 110 |
| Zasady formatowania wujka Boba | 111 |
| 6. Obiekty i struktury danych | 113 |
| Abstrakcja danych | 113 |
| Antysymetria danych i obiektów | 115 |
| Prawo Demeter | 117 |
| Wraki pociągów | 118 |
| Hybrydy | 118 |
| Ukrywanie struktury | 119 |
| Obiekty transferu danych | 119 |
| Active Record | 120 |
| Zakończenie | 121 |
| Bibliografia | 121 |
| 7. Obsługa błędów | 123 |
| Użycie wyjątków zamiast kodów powrotu | 124 |
| Rozpoczynanie od pisania instrukcji try-catch-finally | 125 |
| Użycie niekontrolowanych wyjątków | 126 |
| Dostarczanie kontekstu za pomocą wyjątków | 127 |
| Definiowanie klas wyjątków w zależności od potrzeb wywołującego | 127 |

| | |
|---|------------|
| Definiowanie normalnego przepływu | 129 |
| Nie zwracamy null | 130 |
| Nie przekazujemy null | 131 |
| Zakończenie | 132 |
| Bibliografia | 132 |
| 8. Granice | 133 |
| Zastosowanie kodu innych firm | 134 |
| Przeglądanie i zapoznawanie się z granicami | 136 |
| Korzystanie z pakietu log4j | 136 |
| Zalety testów uczących | 138 |
| Korzystanie z nieistniejącego kodu | 138 |
| Czyste granice | 139 |
| Bibliografia | 140 |
| 9. Testy jednostkowe | 141 |
| Trzy prawa TDD | 142 |
| Zachowanie czystości testów | 143 |
| Testy zwiększają możliwości | 144 |
| Czyste testy | 144 |
| Języki testowania specyficzne dla domeny | 147 |
| Podwójny standard | 147 |
| Jedna asercja na test | 149 |
| Jedna koncepcja na test | 150 |
| F.I.R.S.T. | 151 |
| Zakończenie | 152 |
| Bibliografia | 152 |
| 10. Klasy | 153 |
| Organizacja klas | 153 |
| Hermetyzacja | 154 |
| Klasy powinny być małe! | 154 |
| Zasada pojedynczej odpowiedzialności | 156 |
| Spójność | 158 |
| Utrzymywanie spójności powoduje powstanie wielu małych klas | 158 |
| Organizowanie zmian | 164 |
| Izolowanie modułów kodu przed zmianami | 166 |
| Bibliografia | 167 |
| 11. Systemy | 169 |
| Jak budowałbyś miasto? | 170 |
| Oddzielenie konstruowania systemu od jego używania | 170 |
| Wydzielenie modułu main | 171 |
| Fabryki | 172 |
| Wstrzykiwanie zależności | 172 |
| Skalowanie w górę | 173 |
| Separowanie (rozcięcie) problemów | 176 |
| Pośredniki Java | 177 |

| | |
|---|------------|
| Czyste biblioteki Java AOP | 178 |
| Aspekty w AspectJ | 181 |
| Testowanie architektury systemu | 182 |
| Optymalizacja podejmowania decyzji | 183 |
| Korzystaj ze standardów, gdy wnoszą realną wartość | 183 |
| Systemy wymagają języków dziedzinowych | 184 |
| Zakończenie | 184 |
| Bibliografia | 185 |
| 12. Powstawanie projektu | 187 |
| Uzyskiwanie czystości projektu przez jego rozwijanie | 187 |
| Zasada numer 1 prostego projektu — system przechodzi wszystkie testy | 188 |
| Zasady numer 2 – 4 prostego projektu — przebudowa | 188 |
| Brak powtórzeń | 189 |
| Wyrazistość kodu | 191 |
| Minimalne klasy i metody | 192 |
| Zakończenie | 192 |
| Bibliografia | 192 |
| 13. Współbieżność | 193 |
| W jakim celu stosować współbieżność? | 194 |
| Mity i nieporozumienia | 195 |
| Wyzwania | 196 |
| Zasady obrony współbieżności | 196 |
| Zasada pojedynczej odpowiedzialności | 197 |
| Wniosek — ograniczenie zakresu danych | 197 |
| Wniosek — korzystanie z kopii danych | 197 |
| Wniosek — wątki powinny być na tyle niezależne, na ile to tylko możliwe | 198 |
| Poznaj używaną bibliotekę | 198 |
| Kolekcje bezpieczne dla wątków | 198 |
| Poznaj modele wykonania | 199 |
| Producent-konsument | 199 |
| Czytelnik-pisarz | 200 |
| Uczujący filozofowie | 200 |
| Uwaga na zależności pomiędzy synchronizowanymi metodami | 201 |
| Tworzenie małych sekcji synchronizowanych | 201 |
| Pisanie prawidłowego kodu wyłączającego jest trudne | 202 |
| Testowanie kodu wątków | 202 |
| Traktujemy przypadkowe awarie jako potencjalne problemy z wielowątkowością | 203 |
| Na początku uruchamiamy kod niekorzystający z wątków | 203 |
| Nasz kod wątków powinien dać się włączyć | 203 |
| Nasz kod wątków powinien dać się dostrajać | 204 |
| Uruchamiamy więcej wątków, niż mamy do dyspozycji procesorów | 204 |
| Uruchamiamy testy na różnych platformach | 204 |
| Uzbrajamy nasz kod w elementy próbujące wywołać awarie i wymuszające awarie | 205 |
| Instrumentacja ręczna | 205 |
| Instrumentacja automatyczna | 206 |
| Zakończenie | 207 |
| Bibliografia | 208 |

| | |
|--|------------|
| 14. Udane oczyszczanie kodu | 209 |
| Implementacja klasy Args | 210 |
| Args — zgrubny szkic | 216 |
| Argumenty typu String | 228 |
| Zakończenie | 261 |
| 15. Struktura biblioteki JUnit | 263 |
| Biblioteka JUnit | 264 |
| Zakończenie | 276 |
| 16. Przebudowa klasy SerialDate | 277 |
| Na początek uruchamiamy | 278 |
| Teraz poprawiamy | 280 |
| Zakończenie | 293 |
| Bibliografia | 294 |
| 17. Zapachy kodu i heurystyki | 295 |
| Komentarze | 296 |
| C1. Niewłaściwe informacje | 296 |
| C2. Przeszarżane komentarze | 296 |
| C3. Nadmiarowe komentarze | 296 |
| C4. Źle napisane komentarze | 297 |
| C5. Zakomentowany kod | 297 |
| Środowisko | 297 |
| E1. Budowanie wymaga więcej niż jednego kroku | 297 |
| E2. Testy wymagają więcej niż jednego kroku | 297 |
| Funkcje | 298 |
| F1. Nadmiar argumentów | 298 |
| F2. Argumenty wyjściowe | 298 |
| F3. Argumenty znacznikowe | 298 |
| F4. Martwe funkcje | 298 |
| Ogólne | 298 |
| G1. Wiele języków w jednym pliku źródłowym | 298 |
| G2. Oczywiste działanie jest nieimplementowane | 299 |
| G3. Niewłaściwe działanie w warunkach granicznych | 299 |
| G4. Zdjęte zabezpieczenia | 299 |
| G5. Powtórzenia | 300 |
| G6. Kod na nieodpowiednim poziomie abstrakcji | 300 |
| G7. Klasy bazowe zależne od swoich klas pochodnych | 301 |
| G8. Za dużo informacji | 302 |
| G9. Martwy kod | 302 |
| G10. Separacja pionowa | 303 |
| G11. Niespójność | 303 |
| G12. Zaciemnianie | 303 |
| G13. Sztuczne sprzężenia | 303 |
| G14. Zazdrość o funkcje | 304 |
| G15. Argumenty wybierające | 305 |
| G16. Zaciemnianie intencji | 305 |
| G17. Źle rozmieszczona odpowiedzialność | 306 |

| | |
|---|------------|
| G18. Niewłaściwe metody statyczne | 306 |
| G19. Użycie opisowych zmiennych | 307 |
| G20. Nazwy funkcji powinny informować o tym, co realizują | 307 |
| G21. Zrozumienie algorytmu | 308 |
| G22. Zamiana zależności logicznych na fizyczne | 308 |
| G23. Zastosowanie polimorfizmu zamiast instrukcji if-else lub switch-case | 309 |
| G24. Wykorzystanie standardowych konwencji | 310 |
| G25. Zamiana magicznych liczb na stałe nazwane | 310 |
| G26. Precyzja | 311 |
| G27. Struktura przed konwencją | 312 |
| G28. Hermetyzacja warunków | 312 |
| G29. Unikanie warunków negatywnych | 312 |
| G30. Funkcje powinny wykonywać jedną operację | 312 |
| G31. Ukryte sprzężenia czasowe | 313 |
| G32. Unikanie dowolnych działań | 314 |
| G33. Hermetyzacja warunków granicznych | 314 |
| G34. Funkcje powinny zagłębiać się na jeden poziom abstrakcji | 315 |
| G35. Przechowywanie danych konfigurowalnych na wysokim poziomie | 316 |
| G36. Unikanie nawigacji przechodnich | 317 |
| Java | 317 |
| J1. Unikanie długich list importu przez użycie znaków wieloznacznych | 317 |
| J2. Nie dziedziczymy stałych | 318 |
| J3. Stałe kontra typy wyliczeniowe | 319 |
| Nazwy | 320 |
| N1. Wybór opisowych nazw | 320 |
| N2. Wybór nazw na odpowiednich poziomach abstrakcji | 321 |
| N3. Korzystanie ze standardowej nomenklatury tam, gdzie jest to możliwe | 322 |
| N4. Jednoznaczne nazwy | 322 |
| N5. Użycie długich nazw dla długich zakresów | 323 |
| N6. Unikanie kodowania | 323 |
| N7. Nazwy powinny opisywać efekty uboczne | 323 |
| Testy | 324 |
| T1. Niewystarczające testy | 324 |
| T2. Użycie narzędzi kontroli pokrycia | 324 |
| T3. Nie pomijaj prostych testów | 324 |
| T4. Ignorowany test jest wskazaniem niejednoznaczności | 324 |
| T5. Warunki graniczne | 324 |
| T6. Dokładne testowanie pobliskich błędów | 324 |
| T7. Wzorce błędów wiele ujawniają | 324 |
| T8. Wzorce pokrycia testami wiele ujawniają | 325 |
| T9. Testy powinny być szybkie | 325 |
| Zakończenie | 325 |
| Bibliografia | 325 |
| A Współbieżność II | 327 |
| Przykład klient-serwer | 327 |
| Serwer | 327 |
| Dodajemy wątki | 329 |
| Uwagi na temat serwera | 329 |
| Zakończenie | 331 |

| | |
|---|------------|
| Możliwe ścieżki wykonania | 331 |
| Liczba ścieżek | 332 |
| Kopiemy głębiej | 333 |
| Zakończenie | 336 |
| Poznaj używaną bibliotekę | 336 |
| Biblioteka Executor | 336 |
| Rozwiązania nieblokujące | 337 |
| Bezpieczne klasy nieobstugujące wątków | 338 |
| Zależności między metodami mogą uszkodzić kod współbieżny | 339 |
| Tolerowanie awarii | 340 |
| Blokowanie na kliencie | 340 |
| Blokowanie na serwerze | 342 |
| Zwiększanie przepustowości | 343 |
| Obliczenie przepustowości jednowątkowej | 344 |
| Obliczenie przepustowości wielowątkowej | 344 |
| Zakleszczenie | 345 |
| Wzajemne wykluczanie | 346 |
| Blokowanie i oczekiwanie | 346 |
| Brak wywłaszczania | 346 |
| Cykliczne oczekiwanie | 346 |
| Zapobieganie wzajemnemu wykluczaniu | 347 |
| Zapobieganie blokowaniu i oczekiwaniu | 347 |
| Umożliwienie wywłaszczania | 348 |
| Zapobieganie oczekiwaniu cyklicznemu | 348 |
| Testowanie kodu wielowątkowego | 349 |
| Narzędzia wspierające testowanie kodu korzystającego z wątków | 351 |
| Zakończenie | 352 |
| Samouczek. Pełny kod przykładów | 352 |
| Klient-serwer bez wątków | 352 |
| Klient-serwer z użyciem wątków | 355 |
| B org.jfree.date.SerialDate | 357 |
| C Odwołania do heurystyk | 411 |
| Epilog | 413 |
| Skorowidz | 415 |