

Spis treści

Wstęp (13)

Podziękowania (19)

1. Wprowadzenie do Javy (21)

1.1. Java jako platforma programistyczna (21)

1.2. Słowa klucze białej księgi Javy (22)

1.2.1. Prostota (23)

1.2.2. Obiektowość (23)

1.2.3. Sieciowość (24)

1.2.4. Niezawodność (24)

1.2.5. Bezpieczeństwo (24)

1.2.6. Niezależność od architektury (25)

1.2.7. Przenośność (26)

1.2.8. Interpretacja (26)

1.2.9. Wysoka wydajność (27)

1.2.10. Wielowątkowość (27)

1.2.11. Dynamiczność (27)

1.3. Aplety Javy i internet (28)

1.4. Krótka historia Javy (29)

1.5. Główne nieporozumienia dotyczące Javy (31)

2. Środowisko programistyczne Javy (35)

2.1. Instalacja oprogramowania Java Development Kit (36)

2.1.1. Pobieranie pakietu JDK (36)

2.1.2. Instalacja pakietu JDK (38)	
2.1.3. Instalacja plików źródłowych i dokumentacji (40)	
2.2. Używanie narzędzi wiersza poleceń (41)	
2.3. Praca w zintegrowanym środowisku programistycznym (43)	
2.4. Uruchamianie aplikacji graficznej (46)	
2.5. Tworzenie i uruchamianie apletów (48)	
3. Podstawowe elementy języka Java (55)	
3.1. Prosty program w Javie (56)	
3.2. Komentarze (59)	
3.3. Typy danych (60)	
3.3.1. Typy całkowite (60)	
3.3.2. Typy zmiennoprzecinkowe (61)	
3.3.3. Typ char (62)	
3.3.4. Unicode i typ char (63)	
3.3.5. Typ boolean (64)	
3.4. Zmienne (65)	
3.4.1. Inicjalizacja zmiennych (66)	
3.4.2. Stałe (67)	
3.5. Operatory (67)	
3.5.1. Funkcje i stałe matematyczne (68)	
3.5.2. Konwersja typów numerycznych (70)	
3.5.3. Rzutowanie (71)	
3.5.4. Łączenie przypisania z innymi operatorami (72)	
3.5.5. Operatory inkrementacji i dekrementacji (72)	
3.5.6. Operatory relacyjne i logiczne (73)	
3.5.7. Operatory bitowe (73)	

- 3.5.8. Nawiasy i priorytety operatorów (74)
- 3.5.9. Typ wyliczeniowy (75)
- 3.6. Łącuchy (76)
 - 3.6.1. Podłańcuchy (76)
 - 3.6.2. Konkatenacja (76)
 - 3.6.3. Łącuchów nie można modyfikować (77)
 - 3.6.4. Porównywanie łańcuchów (78)
 - 3.6.5. Łącuchy puste i łańcuchy null (79)
 - 3.6.6. Współrzędne kodowe znaków i jednostki kodowe (80)
 - 3.6.7. API String (81)
 - 3.6.8. Dokumentacja API w internecie (84)
 - 3.6.9. Składanie łańcuchów (85)
- 3.7. Wejście i wyjście (88)
 - 3.7.1. Odbieranie danych wejściowych (88)
 - 3.7.2. Formatowanie danych wyjściowych (91)
 - 3.7.3. Zapis i odczyt plików (95)
- 3.8. Sterowanie wykonywaniem programu (97)
 - 3.8.1. Zasięg blokowy (97)
 - 3.8.2. Instrukcje warunkowe (98)
 - 3.8.3. Pętle (100)
 - 3.8.4. Pętle o określonej liczbie powtórzeń (104)
 - 3.8.5. Wybór wielokierunkowy - instrukcja switch (108)
 - 3.8.6. Instrukcje przerywające przepływ sterowania (110)
- 3.9. Wielkie liczby (113)
- 3.10. Tablice (115)
 - 3.10.1. Pętla typu for each (116)
 - 3.10.2. Inicjowanie tablic i tworzenie tablic anonimowych (117)

- 3.10.3. Kopiowanie tablicy (118)
- 3.10.4. Parametry wiersza poleceń (119)
- 3.10.5. Sortowanie tablicy (120)
- 3.10.6. Tablice wielowymiarowe (123)
- 3.10.7. Tablice postrzępione (126)

4. Obiekty i klasy (129)

- 4.1. Wstęp do programowania obiektowego (130)
 - 4.1.1. Klasy (131)
 - 4.1.2. Obiekty (131)
 - 4.1.3. Identyfikacja klas (132)
 - 4.1.4. Relacje między klasami (133)
- 4.2. Używanie klas predefiniowanych (134)
 - 4.2.1. Obiekty i zmienne obiektów (135)
 - 4.2.2. Klasa LocalDate (137)
 - 4.2.3. Metody udostępniające i zmieniające wartość elementu (139)
- 4.3. Definiowanie własnych klas (142)
 - 4.3.1. Klasa Employee (143)
 - 4.3.2. Używanie wielu plików źródłowych (145)
 - 4.3.3. Analiza klasy Employee (146)
 - 4.3.4. Pierwsze kroki w tworzeniu konstruktorów (147)
 - 4.3.5. Parametry jawne i niejawne (148)
 - 4.3.6. Korzyści z hermetyzacji (149)
 - 4.3.7. Przywileje klasowe (151)
 - 4.3.8. Metody prywatne (152)
 - 4.3.9. Stałe jako pola klasy (152)
- 4.4. Pola i metody statyczne (153)

- 4.4.1. Pola statyczne (153)
- 4.4.2. Stałe statyczne (154)
- 4.4.3. Metody statyczne (155)
- 4.4.4. Metody fabryczne (156)
- 4.4.5. Metoda main (156)
- 4.5. Parametry metod (159)
- 4.6. Konstruowanie obiektów (165)
 - 4.6.1. Przeciążanie (165)
 - 4.6.2. Domyślna inicjalizacja pól (166)
 - 4.6.3. Konstruktor bezargumentowy (167)
 - 4.6.4. Jawna inicjalizacja pól (167)
 - 4.6.5. Nazywanie parametrów (168)
 - 4.6.6. Wywoływanie innego konstruktora (169)
 - 4.6.7. Bloki inicjalizujące (170)
 - 4.6.8. Niszczenie obiektów i metoda finalize (174)
- 4.7. Pakiety (174)
 - 4.7.1. Importowanie klas (175)
 - 4.7.2. Importowanie statyczne (177)
 - 4.7.3. Dodawanie klasy do pakietu (177)
 - 4.7.4. Zasięg pakietów (180)
- 4.8. Ścieżka klas (181)
 - 4.8.1. Ustawianie ścieżki klas (184)
- 4.9. Komentarze dokumentacyjne (184)
 - 4.9.1. Wstawianie komentarzy (185)
 - 4.9.2. Komentarze do klas (186)
 - 4.9.3. Komentarze do metod (186)
 - 4.9.4. Komentarze do pól (187)

- 4.9.5. Komentarze ogólne (187)
- 4.9.6. Komentarze do pakietów i ogólne (188)
- 4.9.7. Generowanie dokumentacji (189)
- 4.10. Porady dotyczące projektowania klas (190)

5. Dziedziczenie (193)

- 5.1. Klasy, nadklasy i podklasy (194)
 - 5.1.1. Definiowanie podklas (194)
 - 5.1.2. Przesłanianie metod (195)
 - 5.1.3. Konstruktory podklas (197)
 - 5.1.4. Hierarchia dziedziczenia (201)
 - 5.1.5. Polimorfizm (201)
 - 5.1.6. Zasady wywoływania metod (203)
 - 5.1.7. Wyłączanie dziedziczenia - klasy i metody finalne (205)
 - 5.1.8. Rzutowanie (206)
 - 5.1.9. Klasy abstrakcyjne (209)
 - 5.1.10. Ograniczanie dostępu (214)
- 5.2. Kosmiczna klasa wszystkich klas - Object (215)
 - 5.2.1. Metoda equals (215)
 - 5.2.2. Porównywanie a dziedziczenie (217)
 - 5.2.3. Metoda hashCode (220)
 - 5.2.4. Metoda toString (222)
- 5.3. Generyczne listy tablicowe (228)
 - 5.3.1. Dostęp do elementów listy tablicowej (231)
 - 5.3.2. Zgodność pomiędzy typowanymi a surowymi listami tablicowymi (234)
- 5.4. Opakowania obiektów i automatyczne pakowanie (235)
- 5.5. Metody ze zmienną liczbą parametrów (238)

5.6. Klasy wyliczeniowe (240)

5.7. Refleksja (242)

5.7.1. Klasa Class (242)

5.7.2. Podstawy przechwytywania wyjątków (244)

5.7.3. Zastosowanie refleksji w analizie funkcjonalności klasy (246)

5.7.4. Refleksja w analizie obiektów w czasie działania programu (251)

5.7.5. Zastosowanie refleksji w generycznym kodzie tablicowym (255)

5.7.6. Wywoływanie dowolnych metod (258)

5.8. Porady projektowe dotyczące dziedziczenia (261)

6. Interfejsy, wyrażenia lambda i klasy wewnętrzne (265)

6.1. Interfejsy (266)

6.1.1. Koncepcja interfejsu (266)

6.1.2. Własności interfejsów (272)

6.1.3. Interfejsy a klasy abstrakcyjne (273)

6.1.4. Metody statyczne (274)

6.1.5. Metody domyślne (275)

6.1.6. Wybieranie między metodami domyślnymi (276)

6.2. Przykłady interfejsów (278)

6.2.1. Interfejsy i wywołania zwrotne (278)

6.2.2. Interfejs Comparator (281)

6.2.3. Klonowanie obiektów (282)

6.3. Wyrażenia lambda (288)

6.3.1. Po co w ogóle są lambdy (288)

6.3.2. Składnia wyrażeń lambda (289)

6.3.3. Interfejsy funkcyjne (292)

6.3.4. Referencje do metod (293)

- 6.3.5. Referencje do konstruktorów (295)
- 6.3.6. Zakres dostępności zmiennych (295)
- 6.3.7. Przetwarzanie wyrażeń lambda (298)
- 6.3.8. Poszerzenie wiadomości o komparatorach (301)
- 6.4. Klasy wewnętrzne (302)
 - 6.4.1. Dostęp do stanu obiektu w klasie wewnętrznej (303)
 - 6.4.2. Specjalne reguły składniowe dotyczące klas wewnętrznych (306)
 - 6.4.3. Czy klasy wewnętrzne są potrzebne i bezpieczne? (307)
 - 6.4.4. Lokalne klasy wewnętrzne (310)
 - 6.4.5. Dostęp do zmiennych finalnych z metod zewnętrznych (310)
 - 6.4.6. Anonimowe klasy wewnętrzne (313)
 - 6.4.7. Statyczne klasy wewnętrzne (316)
- 6.5. Klasy pośredniczące (319)
 - 6.5.1. Kiedy używać klas pośredniczących (319)
 - 6.5.2. Tworzenie obiektów pośredniczących (320)
 - 6.5.3. Właściwości klas pośredniczących (324)
- 7. Wyjątki, asercje i dzienniki (327)
 - 7.1. Obsługa błędów (328)
 - 7.1.1. Klasyfikacja wyjątków (329)
 - 7.1.2. Deklarowanie wyjątków kontrolowanych (331)
 - 7.1.3. Zgłaszanie wyjątków (333)
 - 7.1.4. Tworzenie klas wyjątków (334)
 - 7.2. Przechwytywanie wyjątków (335)
 - 7.2.1. Przechwytywanie wyjątku (335)
 - 7.2.2. Przechwytywanie wielu typów wyjątków (337)
 - 7.2.3. Powtórne generowanie wyjątków i budowanie łańcuchów wyjątków (339)

- 7.2.4. Klauzula finally (340)
- 7.2.5. Instrukcja try z zasobami (343)
- 7.2.6. Analiza danych ze śledzenia stosu (345)
- 7.3. Wskazówki dotyczące stosowania wyjątków (348)
- 7.4. Asercje (350)
 - 7.4.1. Koncepcja asercji (351)
 - 7.4.2. Włączanie i wyłączanie asercji (352)
 - 7.4.3. Zastosowanie asercji do sprawdzania parametrów (352)
 - 7.4.4. Zastosowanie asercji do dokumentowania założeń (353)
- 7.5. Dzienniki (355)
 - 7.5.1. Podstawy zapisu do dziennika (355)
 - 7.5.2. Zaawansowane techniki zapisu do dziennika (356)
 - 7.5.3. Zmiana konfiguracji menedżera dzienników (358)
 - 7.5.4. Lokalizacja (359)
 - 7.5.5. Obiekty typu Handler (360)
 - 7.5.6. Filtry (363)
 - 7.5.7. Formatery (364)
 - 7.5.8. Przepis na dziennik (364)
- 7.6. Wskazówki dotyczące debugowania (372)
- 8. Programowanie generyczne (379)
 - 8.1. Dlaczego programowanie generyczne (380)
 - 8.1.1. Zalety parametrów typów (380)
 - 8.1.2. Dla kogo programowanie generyczne (381)
 - 8.2. Definicja prostej klasy generycznej (382)
 - 8.3. Metody generyczne (384)
 - 8.4. Ograniczenia zmiennych typowych (385)

8.5. Kod generyczny a maszyna wirtualna (387)

8.5.1. Wymazywanie typów (388)

8.5.2. Translacja wyrażeń generycznych (389)

8.5.3. Translacja metod generycznych (389)

8.5.4. Używanie starego kodu (392)

8.6. Ograniczenia i braki (393)

8.6.1. Nie można podawać typów prostych jako parametrów typowych (393)

8.6.2. Sprawdzanie typów w czasie działania programu jest możliwe tylko dla typów surowych (393)

8.6.3. Nie można tworzyć tablic typów generycznych (394)

8.6.4. Ostrzeżenia dotyczące zmiennej liczby argumentów (394)

8.6.5. Nie wolno tworzyć egzemplarzy zmiennych typowych (395)

8.6.6. Nie można utworzyć egzemplarza generycznej tablicy (396)

8.6.7. Zmiennych typowych nie można używać w statycznych kontekstach klas generycznych (398)

8.6.8. Obiektów klasy generycznej nie można generować ani przechwytywać (398)

8.6.9. Można wyłączyć sprawdzanie wyjątków kontrolowanych (399)

8.6.10. Uważaj na konflikty, które mogą powstać po wymazaniu typów (401)

8.7. Zasady dziedziczenia dla typów generycznych (402)

8.8. Typy wieloznaczne (404)

8.8.1. Koncepcja typu wieloznacznego (404)

8.8.2. Ograniczenia nadtypów typów wieloznacznych (405)

8.8.3. Typy wieloznaczne bez ograniczeń (408)

8.8.4. Chwytywanie typu wieloznacznego (408)

8.9. Refleksja a typy generyczne (411)

8.9.1. Generyczna klasa Class (411)

8.9.2. Zastosowanie parametrów Class do dopasowywania typów (412)

8.9.3. Informacje o typach generycznych w maszynie wirtualnej (412)

9. Kolekcje (419)

9.1. Architektura kolekcji Javy (419)

9.1.1. Oddzielenie warstwy interfejsów od warstwy klas konkretnych (420)

9.1.2. Interfejs Collection (422)

9.1.3. Iteratory (423)

9.1.4. Generyczne metody użytkowe (425)

9.1.5. Interfejsy w systemie kolekcji Javy (428)

9.2. Konkretnie klasy kolekcyjne (430)

9.2.1. Listy powiązane (431)

9.2.2. Listy tablicowe (440)

9.2.3. Zbiór HashSet (440)

9.2.4. Zbiór TreeSet (444)

9.2.5. Kolejki Queue i Deque (448)

9.2.6. Kolejki priorytetowe (450)

9.3. Słowniki (451)

9.3.1. Podstawowe operacje słownikowe (451)

9.3.2. Modyfikowanie wpisów w słowniku (454)

9.3.3. Widoki słowników (456)

9.3.4. Klasa WeakHashMap (457)

9.3.5. Klasy LinkedHashSet i LinkedHashMap (458)

9.3.6. Klasy EnumSet i EnumMap (459)

9.3.7. Klasa IdentityHashMap (460)

9.4. Widoki i opakowania (462)

9.4.1. Lekkie obiekty opakowujące kolekcje (462)

9.4.2. Przedziały (463)

9.4.3. Widoki niemodyfikowalne (464)

- 9.4.4. Widoki synchronizowane (465)
- 9.4.5. Widoki kontrolowane (465)
- 9.4.6. Uwagi dotyczące operacji opcjonalnych (466)
- 9.5. Algorytmy (469)
 - 9.5.1. Sortowanie i tasowanie (470)
 - 9.5.2. Wyszukiwanie binarne (473)
 - 9.5.3. Proste algorytmy (474)
 - 9.5.4. Operacje zbiorowe (476)
 - 9.5.5. Konwersja pomiędzy kolekcjami a tablicami (477)
 - 9.5.6. Pisanie własnych algorytmów (478)
- 9.6. Stare kolekcje (479)
 - 9.6.1. Klasa Hashtable (479)
 - 9.6.2. Wyliczenia (480)
 - 9.6.3. Słowniki własności (481)
 - 9.6.4. Stosy (482)
 - 9.6.5. Zbiory bitów (482)
- 10. Grafika (487)
 - 10.1. Wprowadzenie do pakietu Swing (488)
 - 10.2. Tworzenie ramki (492)
 - 10.3. Pozycjonowanie ramki (494)
 - 10.3.1. Własności ramek (496)
 - 10.3.2. Określanie rozmiaru ramki (497)
 - 10.4. Wyświetlanie informacji w komponencie (500)
 - 10.5. Figury 2D (505)
 - 10.6. Kolory (513)
 - 10.7. Czcionki (517)

10.8. Wyświetlanie obrazów (525)

11. Obsługa zdarzeń (529)

11.1. Podstawy obsługi zdarzeń (529)

11.1.1. Przykład - obsługa kliknięcia przycisku (531)

11.1.2. Zwięzłe definiowanie procedur nasłuchowych (536)

11.1.3. Przykład - zmiana stylu (538)

11.1.4. Klasy adaptacyjne (542)

11.2. Akcje (546)

11.3. Zdarzenia generowane przez mysz (553)

11.4. Hierarchia zdarzeń w bibliotece AWT (560)

11.4.1. Zdarzenia semantyczne i niskiego poziomu (561)

12. Komponenty Swing interfejsu użytkownika (565)

12.1. Swing a wzorzec projektowy Model-View-Controller (566)

12.1.1. Wzorce projektowe (566)

12.1.2. Wzorzec Model-View-Controller (568)

12.1.3. Analiza MVC przycisków Swing (571)

12.2. Wprowadzenie do zarządzania rozkładem (572)

12.2.1. Rozkład brzegowy (574)

12.2.2. Rozkład siatkowy (576)

12.3. Wprowadzanie tekstu (580)

12.3.1. Pola tekstowe (580)

12.3.2. Etykiety komponentów (582)

12.3.3. Pola haseł (584)

12.3.4. Obszary tekstowe (584)

12.3.5. Panele przewijane (585)

12.4. Komponenty umożliwiające wybór opcji (587)

12.4.1. Pola wyboru (587)

12.4.2. Przełączniki (590)

12.4.3. Obrazowanie (593)

12.4.4. Listy rozwijalne (597)

12.4.5. Suwaki (600)

12.5. Menu (606)

12.5.1. Tworzenie menu (606)

12.5.2. Ikony w elementach menu (609)

12.5.3. Pola wyboru i przełączniki jako elementy menu (610)

12.5.4. Menu podręczne (611)

12.5.5. Mnemoniki i akceleratory (612)

12.5.6. Aktywowanie i dezaktywowanie elementów menu (614)

12.5.7. Paski narzędzi (618)

12.5.8. Dymki (620)

12.6. Zaawansowane techniki zarządzania rozkładem (623)

12.6.1. Rozkład GridBagLayout (624)

12.6.2. Rozkład grupowy (634)

12.6.3. Rezygnacja z zarządców rozkładu (643)

12.6.4. Niestandardowi zarządcy rozkładu (643)

12.6.5. Kolejka dostępu (647)

12.7. Okna dialogowe (648)

12.7.1. Okna dialogowe opcji (649)

12.7.2. Tworzenie okien dialogowych (659)

12.7.3. Wymiana danych (663)

12.7.4. Okna dialogowe wyboru plików (669)

12.7.5. Okna dialogowe wyboru kolorów (679)

12.8. Rozwiązywanie problemów z programami z graficznym interfejsem użytkownika (684)

12.8.1. Wskazówki dotyczące debugowania (684)

12.8.2. Zaprzęganie robota AWT do pracy (686)

13. Przygotowywanie apletów i aplikacji do użytku (693)

13.1. Pliki JAR (694)

13.1.1. Tworzenie plików JAR (694)

13.1.2. Manifest (695)

13.1.3. Wykonywalne pliki JAR (696)

13.1.4. Zasoby (697)

13.1.5. Pieczętowanie pakietów (700)

13.2. Zapisywanie preferencji użytkownika (701)

13.2.1. Słowniki własności (701)

13.2.2. API Preferences (706)

13.3. Moduły ładowania usług (712)

13.4. Aplety (714)

13.4.1. Prosty aplet (715)

13.4.2. Znacznik applet i jego atrybuty (718)

13.4.3. Parametry przekazujące informacje do apletów (720)

13.4.4. Dostęp do obrazów i plików audio (725)

13.4.5. Środowisko działania apletu (726)

13.4.6. Komunikacja pomiędzy apletami (727)

13.4.7. Wyświetlanie elementów w przeglądarce (727)

13.4.8. Piaskownica (729)

13.4.9. Podpisywanie kodu (730)

13.5. Java Web Start (732)

13.5.1. Wdrażanie aplikacji Java Web Start (732)

13.5.2. API JNLP (735)

14. Współbieżność (745)

14.1. Czym są wątki (746)

14.1.1. Wykonywanie zadań w osobnych wątkach (751)

14.2. Przerywanie wątków (755)

14.3. Stany wątków (758)

14.3.1. Wątki NEW (758)

14.3.2. Wątki RUNNABLE (758)

14.3.3. Wątki BLOCKED i WAITING (759)

14.3.4. Zamykanie wątków (759)

14.4. Własności wątków (761)

14.4.1. Priorytety wątków (761)

14.4.2. Wątki demony (762)

14.4.3. Procedury obsługi nieprzechwyconych wyjątków (762)

14.5. Synchronizacja (764)

14.5.1. Przykład sytuacji powodującej wyścig (764)

14.5.2. Wyścigi (768)

14.5.3. Obiekty klasy Lock (769)

14.5.4. Warunki (772)

14.5.5. Słowo kluczowe synchronized (777)

14.5.6. Bloki synchronizowane (781)

14.5.7. Monitor (783)

14.5.8. Pola ulotne (783)

14.5.9. Zmienne finalne (785)

14.5.10. Zmienne atomowe (785)

14.5.11. Zakleszczenia (787)

- 14.5.12. Zmienne lokalne wątków (790)
- 14.5.13. Testowanie blokad i odmierzenie czasu (791)
- 14.5.14. Blokada odczytu-zapisu (793)
- 14.5.15. Dlaczego metody stop i suspend są wycofywane (794)
- 14.6. Kolejki blokujące (795)
- 14.7. Kolekcje bezpieczne wątkowo (802)
 - 14.7.1. Szybkie słowniki, zbiory i kolejki (802)
 - 14.7.2. Atomowe modyfikowanie elementów słowników (804)
 - 14.7.3. Operacje masowe na współbieżnych słownikach skrótów (806)
 - 14.7.4. Współbieżne widoki zbiorów (808)
 - 14.7.5. Tablice kopiowane przy zapisie (808)
 - 14.7.6. Równoległe algorytmy tablicowe (808)
 - 14.7.7. Starsze kolekcje bezpieczne wątkowo (809)
- 14.8. Interfejsy Callable i Future (810)
- 14.9. Klasa Executors (815)
 - 14.9.1. Pule wątków (815)
 - 14.9.2. Planowanie wykonywania (820)
 - 14.9.3. Kontrolowanie grup zadań (821)
 - 14.9.4. Szkielet rozgałęzienie-złączenie (822)
 - 14.9.5. Klasa CompletableFuture (824)
- 14.10. Synchronizatory (827)
 - 14.10.1. Semafor (827)
 - 14.10.2. Klasa CountdownLatch (828)
 - 14.10.3. Bariery (829)
 - 14.10.4. Klasa Exchanger (830)
 - 14.10.5. Kolejki synchroniczne (830)
- 14.11. Wątki a biblioteka Swing (830)

14.11.1. Uruchamianie czasochłonnych zadań (831)

14.11.2. Klasa SwingWorker (835)

14.11.3. Zasada jednego wątku (841)

A. Słowa kluczowe Javy (843)

Skorowidz (845)