

Spis treści (podsumowanie)

Wprowadzenie	xxiii
1 Przełamując zalew początkowych trudności. Szybki skok na głęboką wodę	1
2 Wycieczka do Obiektowa. Klasy i obiekty	27
3 Poznaj swoje zmienne. Typy podstawowe i odwołania	49
4 Jak działają obiekty. Metody wykorzystują zmienne instancyjne	71
5 Supermocne metody. Pisanie programu	95
6 Korzystanie z biblioteki Javy. Poznaj Java API	123
7 Wygodniejsze życie w Obiektowie. Dziedziczenie i polimorfizm	163
8 Poważny polimorfizm. Interfejsy i klasy abstrakcyjne	195
9 Życie i śmierć obiektu. Konstruktory i odśmiecacz	233
10 Liczby mają znaczenie. Liczby oraz składowe statyczne	271
11 Struktury danych. Kolekcje i typy ogólne	305
12 Co, a nie jak. Wyrażenia lambda i strumienie	365
13 Ryzykowne zachowanie. Obsługa wyjątków	417
14 Historia bardzo graficzna. Tworzenie graficznego interfejsu użytkownika	455
15 Popracuj nad Swingiem. Stosowanie biblioteki Swing	503
16 Zapisywanie obiektów (i tekstu). Serializacja i operacje wejścia-wyjścia na plikach	531
17 Nawiaź połączenie. Zagadnienia sieciowe i wątki	579
18 Rozwiązywanie problemów współbieżności. Warunki współzawodnictwa i dane niemodyfikowalne	631
A Dodatek A. Ostatnie poprawianie kodu	665
B Dodatek B. Jedenaście najważniejszych tematów, które nie zmieściły się w głównej części książki	675
Skorowidz	693

Spis treści (z prawdziwego zdarzenia)

W

Wprowadzenie

Twój mózg myśli o Javie. W tym rozdziale Ty próbujesz się czegoś dowiedzieć, natomiast Twój mózg wyświadcza Ci uprzejmość i stara się, aby te informacje nie zostały zapamiętane na długo. Myśli sobie: „Lepiej zostawić miejsce na ważniejsze rzeczy, takie jak dzikie zwierzęta, których należy się wystrzegać, lub rozważania, czy jeżdżenie na snowboardzie w stroju Adama to dobry pomysł”. Jak zatem można oszukać własny mózg i przekonać go, że od znajomości Javy zależy nasze życie?

Dla kogo jest przeznaczona ta książka?	xxiv
Wiemy, co sobie myślisz	xxv
Metapoznanie — myślenie o myśleniu	xxvii
Oto co zrobiliśmy	xxviii
Oto co możesz zrobić, aby zmusić swój mózg do posłuszeństwa	xxix
Czego potrzebujesz, aby skorzystać z tej książki?	xxx
Kilka ostatnich spraw, o których musisz wiedzieć	xxxi

1 Przełamując zalew początkowych trudności

Java zabiera nas w nowe miejsca. Od momentu pojawienia się pierwszej, skromnej wersji o numerze 1.02 Java pociągała programistów ze względu na przyjazną składnię, cechy obiektowe, zarządzanie pamięcią, a przede wszystkim obietnicę przenośności. Wskoczmy od razu na głęboką wodę: napiszemy prosty program w Javie, skompilujemy go i uruchomimy. Pokażemy składnię Javy, instrukcje warunkowe, pętle i inne rzeczy, które sprawiają, że Java jest super. A więc do dzieła!



Jak działa Java?	2
Co będziesz robić w Javie?	3
Krótką historia Javy	4
Struktura kodu w Javie	7
Tworzenie klasy z metodą main()	9
Pętle i pętle, i...	13
Rozgałęzienia warunkowe	15
Tworzenie poważnej aplikacji biznesowej	16
Program krasomówczy	19
Ćwiczenia	20
Rozwiązania ćwiczeń	24

2 Wycieczka do Obiektowa

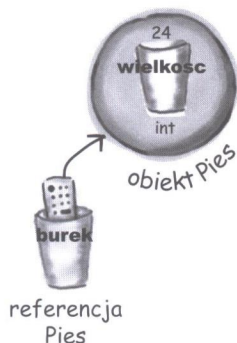
Mówiono mi, że będą obiekty. W rozdziale 1. cały tworzony kod był umieszczany w metodzie `main()`. Nie jest to poprawne rozwiązanie obiektowe. Teraz musimy więc zostawić świat programowania proceduralnego i zacząć tworzyć własne obiekty. Przyjrzymy się czynnikom, które sprawiają, że programowanie zorientowane obiektowo w języku Java jest takie fajne. Przedstawimy różnice pomiędzy klasą a obiektem. Przekonamy się, w jaki sposób obiekty mogą ułatwić nam życie.



Wojna o fotel	28
Tworzenie pierwszego obiektu	36
Tworzenie i testowanie obiektów Film	37
Szybko! Opuszczamy metodę main()!	38
Uruchamianie zgadywanki	40
Ćwiczenia	42
Rozwiązania ćwiczeń	46

3 Poznaj swoje zmienne

Zmienne można podzielić na dwie kategorie: **zmienne typów podstawowych oraz odwołania**. Ale przecież musi być coś bardziej interesującego niż liczby całkowite, łańcuchy znaków i tablice. Co należy zrobić, gdybyśmy chcieli stworzyć obiekt WłaścicielZwierzaka ze zmienną instancyjną Pies? Albo obiekt Samochód ze zmienną instancyjną Silnik? W tym rozdziale wyjaśnimy tajemnice typów danych w Javie i przyjrzymy się, co można *zadeklarować* jako zmienną, co w takiej zmiennej można *zapisać* i do czego jej można *użyć*. W końcu zobaczymy także, jak *naprawdę* wygląda życie na automatycznie odśmiecanej stercie.



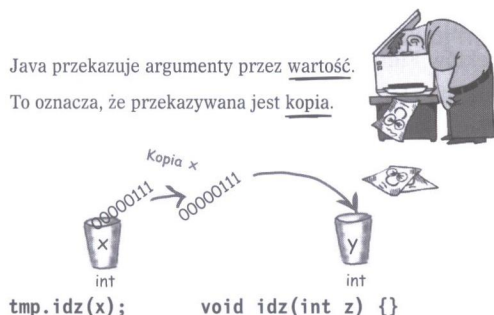
Deklarowanie zmiennej	50
„Proszę podwójną. Albo nie — całkowitą!”	51
Trzymaj się z daleka od tego słowa kluczowego!	53
Kontrolowanie obiektu Pies	54
Odwołanie do obiektu to jedynie inna wartość zmiennej	55
Życie na odśmiecanej stercie	57
Tablica jest jakby tacą z kubkami	59
Przykładowy obiekt Pies	62
Ćwiczenia	63
Rozwiązania ćwiczeń	68

4 Jak działają obiekty?

Stan wpływa na działanie, a działanie wpływa na stan. Wiemy już, że obiekty mają **stan** oraz **działanie**, które to aspekty obiektów są odpowiednio reprezentowane przez **zmienne instancyjne** oraz **metody**. Teraz zajmiemy się zagadnieniem, w jaki sposób stan oraz działanie obiektu są ze sobą *powiązane*. Zachowania obiektu korzystają z jego unikalnego stanu. Innymi słowy, **metody wykorzystują wartości zmiennych instancyjnych**. Na przykład „Jeśli pies waży mniej niż 7 kilogramów, szczekaj piskliwie, w przeciwnym razie...” bądź też: „Powiększ wagę o 2 kilogramy”. **A zatem pozmieniamy stan.**

Java przekazuje argumenty przez wartość.

To oznacza, że przekazywana jest kopia.



Pamiętaj! Klasa opisuje to, co obiekt wie, oraz to, co robi	72
Wielkość ma wpływ na sposób szczekania	73
Do metod można przekazywać informacje	74
Metoda może coś zwrócić	75
Do metody można przekazać więcej niż jedną informację	76
Ciekawe rozwiązania wykorzystujące parametry i wartości wynikowe	79
Hermetyzacja	80
Jak zachowują się obiekty w tablicy?	83
Deklarowanie i inicjalizacja zmiennych instancyjnych	84
Porównywanie zmiennych (typów podstawowych oraz odwołań)	86
Ćwiczenia	88
Rozwiązania ćwiczeń	93

5

Supermocne metody

Dodajmy naszym metodom nieco siły. Igraliśmy ze zmiennymi, zabawialiśmy się z kilkoma obiektami i napisaliśmy parę wierszy kodu. Ale potrzeba nam więcej narzędzi. Takich jak **operatory**. Oraz **pętle**. Może się nam przydać umiejętność **generowania liczb losowych**. Albo **zamieniania łańcucha znaków na liczbę**, ech... to by było świetne. A czy nie można by się nauczyć tego wszystkiego, *pisząc* jakiś program? Tak żeby zobaczyć, jak wygląda pisanie i testowanie normalnego programu od samego początku. **Na przykład jakąś grę...** taką jak gra w okręty.

Napiszemy grę
„Zatopić startup”

A							
B							
C	cabista						
D			poniez				
E							
F							
G				hacqi			
	0	1	2	3	4	5	6

Napiszmy grę przypominającą okręty, o nazwie „Zatopić startup”	96
Tworzenie klasy	99
Pisanie implementacji metod	101
Pisanie kodu testowego dla klasy ProstyStartup	102
Metoda sprawdz()	104
Kod przygotowawczy klasy ProstyStartupGra	108
Metoda main() gry	110
Zagrajmy	113
Trochę więcej o pętlach for	114
Rozszerzone pętle for	116
Rzutowanie wartości typów podstawowych	117
Ćwiczenia	118
Rozwiązania ćwiczeń	121

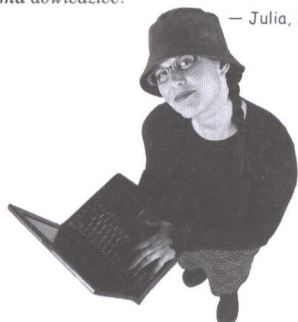
6

Korzystanie z biblioteki Javy

Java jest wyposażona w setki gotowych do użycia klas. Jeśli tylko potrafisz znaleźć w bibliotece Javy, nazywanej **Java API**, to, czego Ci potrzeba, to nie będziesz musiał ponownie wymyślać koła. *W końcu masz ciekawsze rzeczy do roboty.* Jeśli musisz napisać kod, to równie dobrze możesz napisać wyłącznie te jego fragmenty, którą są unikalne dla tworzonej aplikacji. Java API to cała masa klas, które tylko czekają, byś zaczął ich używać jako elementów konstrukcyjnych w swoich programach.

„Dobrze wiedzieć, że w pakiecie java.util istnieje klasa ArrayList. Ale jak mogłabym się o tym sama dowiedzieć?”

— Julia, lat 31, modelka

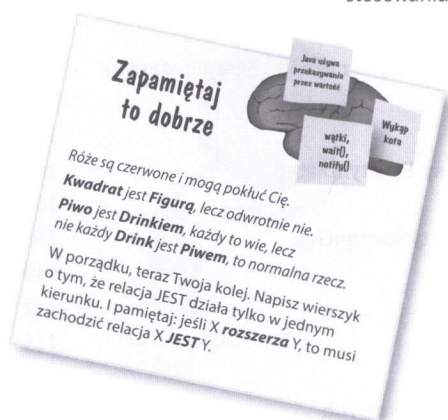


Ostatni rozdział zakończył się w dramatycznych okolicznościach — w programie znaleźliśmy błąd	124
Obudź się i przejrzyj bibliotekę	130
Niektóre możliwości klasy ArrayList	131
Porównanie klasy ArrayList ze zwyczajną tablicą	135
Napiszmy WŁAŚCIWĄ wersję gry „Zatopić startup”	138
Kod przygotowawczy właściwej klasy StartupGraMax	142
Ostateczna wersja klasy Startup	148
Wyrażenia logiczne o bardzo dużych możliwościach	151
Stosowanie biblioteki (Java API)	152
Ćwiczenia	160
Rozwiązania ćwiczeń	161

7

Wygodniejsze życie w Obiekcie

Planuj swoje programy, myśląc perspektywicznie. Co by było, gdybyś mógł tworzyć swój kod w Javie w taki sposób, by ktoś *inny* mógł go **łatwo** rozszerzać? I gdybyś mógł pisać kod bardzo elastyczny, pozwalający na wprowadzanie tych denerwujących zmian specyfikacji zgłaszanych w ostatniej chwili. Gdy zdobędziesz Plan Polimorfizmu, poznasz pięć kroków do lepszego projektowania klas, trzy sztuczki polimorficzne i osiem sposobów tworzenia elastycznego kodu, a jeśli zadzwonisz już teraz — otrzymasz dodatkową lekcję na temat czterech sposobów stosowania dziedziczenia.



Wojna o fotel raz jeszcze...	164
Zrozumienie dziedziczenia	166
Zaprojektujmy drzewo dziedziczenia dla programu symulacji zwierząt	168
Poszukiwanie dalszych możliwości zastosowania dziedziczenia	171
Relacje JEST oraz MA	175
Jak możesz określić, czy dobrze zaprojektowałeś hierarchię dziedziczenia?	177
Czy korzystanie z dziedziczenia przy projektowaniu klas jest „używaniem”, czy „nadużywaniem”?	179
Jak dotrzymać kontraktu — reguły przesłaniania	188
Przeciążanie metody	189
Ćwiczenia	190
Rozwiązania ćwiczeń	193

8

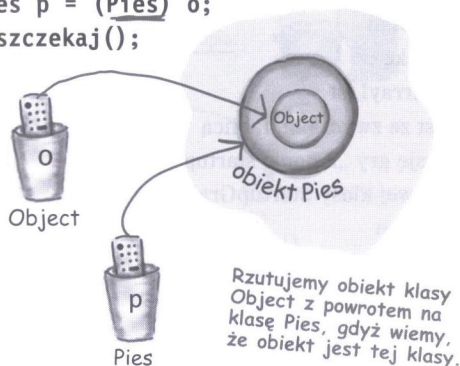
Poważny polimorfizm

Dziedziczenie to jedynie początek. Aby w pełni wykorzystać możliwości, jakie daje polimorfizm, będziemy potrzebować interfejsów. Musimy zostawić proste dziedziczenie i pójść dalej — dotrzeć do poziomu elastyczności i możliwości rozbudowy kodu, jakie dają jedynie projektowanie i programowanie z wykorzystaniem specyfikacji interfejsów. Jednak czym jest interfejs? Otóż interfejs to całkowicie — w stu procentach — abstrakcyjna klasa. A co to jest abstrakcyjna klasa? To klasa, która nie daje możliwości tworzenia obiektów. A do czego taka klasa może nam się przydać? O tym dowiesz się z tego rozdziału.

Object o = lista.get(indeks);

Pies p = (Pies) o;

p.szczekaj();



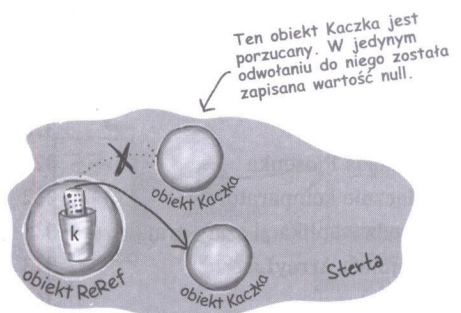
Czy projektując przedstawioną obok hierarchię klas, o czymś zapomnieliśmy?	196
Kompilator nie pozwoli utworzyć obiektów klasy abstrakcyjnej	199
Abstrakcyjne kontra konkretne	200
MUSISZ zaimplementować wszystkie metody abstrakcyjne	202
Polimorfizm w działaniu	204
A co z obiektami innych klas? Dlaczego nie stworzyć listy, w której można by zapisać cokolwiek?	206
Kiedy Pies nie zachowuje się jak Pies	210
Rozważmy różne możliwości projektowe	217
Tworzenie i implementowanie interfejsu ZwierzakDomowy	223
Wywoływanie wersji metody zdefiniowanej w klasie bazowej	226
Ćwiczenia	228
Rozwiązania ćwiczeń	231



9

Życie i śmierć obiektu

Obiekty się rodzą i obiekty umierają. To Ty odpowiadasz za ich cykl życia. Ty decydujesz, kiedy obiekt należy *utworzyć*. Również Ty decydujesz, kiedy obiekt należy *porzucić*. **Odśmiecacz sterty** odzyskuje pamięć. W tym rozdziale opisujemy, w jaki sposób obiekty są tworzone, gdzie są przechowywane podczas swojego istnienia oraz jak można je efektywnie zachować lub porzucić. Oznacza to, że będziemy pisać o stercie, stosie, zasięgu, konstruktorach, konstruktorach bazowych, odwołaniach pustych oraz nadawaniu się do usunięcia.



W zmiennej instancyjnej „k” jest zapisywana wartość null, co można porównać z pilotem, który nie jest zaprogramowany do obsługi żadnego urządzenia. Dopóki zmienna ta nie zostanie ponownie zaprogramowana (czyli do momentu, gdy zapiszemy w niej jakiś obiekt), nie możemy jej nawet używać wraz z operatorem kropki.

Stos i sterta. Gdzie są przechowywane dane?	234
Metody są zapisywane na stosie	235
A co ze zmiennymi lokalnymi, które są obiektami?	236
Cud utworzenia obiektu	238
Tworzenie obiektu Kaczka	240
Czy kompilator zawsze tworzy dla nas konstruktor bezargumentowy?	244
Nanoprzegląd. Cztery rzeczy o konstruktorach, które należy zapamiętać	247
Znaczenie konstruktorów klasy bazowej w życiu obiektu	249
Czy dziecko może istnieć przed rodzicami?	252
A co ze zmiennymi referencyjnymi?	258
Nie podoba mi się kierunek, w jakim to wszystko zmierza.	259
Ćwiczenia	264
Rozwiązania ćwiczeń	268

10

Liczby mają znaczenie

Wykonuj obliczenia matematyczne. Java API udostępnia metody pozwalające wyznaczyć wartość bezwzględną liczby, zaokrąglić ją albo znaleźć większą lub mniejszą z dwóch wartości. A co z formatowaniem? Moglibyśmy chcieć wyświetlać liczby z dokładnie dwoma liczbami dziesiętymi lub z odstępami pomiędzy grupami cyfr. Oprócz tego może się pojawić konieczność wyświetlania dat. Albo zamiany liczb na łańcuchy znaków. Albo łańcucha na liczbę. Zaczniemy od wyjaśnienia, co dla zmiennej lub metody oznacza bycie *statyczną*.

Wszystkie obiekty tej samej klasy wspólnie używają jednej kopii każdej ze składowych statycznych.



Zmienne instancyjne — po jednej w każdej instancji

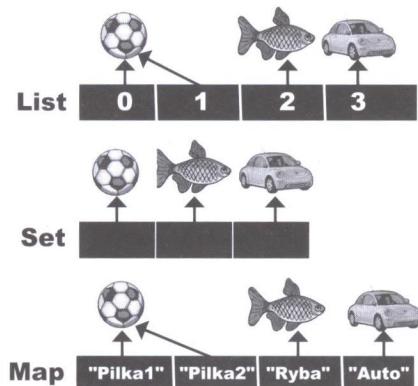
Zmienne statyczne — jedna dla całej klasy

Metody klasy Math — najlepsze z możliwych odpowiedników metod globalnych	272
Różnice pomiędzy metodami zwyczajnymi a statycznymi	273
Inicjalizacja zmiennych statycznych	279
Metody klasy Math	284
Reprezentowanie wartości typów podstawowych w formie obiektów	286
Automatyczna konwersja działa niemal wszędzie	288
A teraz w przeciwnym kierunku... zamiana liczby na łańcuch znaków	291
Formatowanie liczb	292
Specyfikator formatu	296
Ćwiczenia	302
Rozwiązania ćwiczeń	304

11

Struktury danych

Przechowywanie w Javie nie następuje żadnych trudności. Dysponujesz wszystkimi narzędziami związanymi z korzystaniem z kolekcji oraz sortowaniem danych i nie musisz pisać żadnych własnych algorytmów. Biblioteka kolekcji Javy udostępnia struktury danych, które powinny zaspokoić praktycznie wszelkie Twoje potrzeby. Chciałbyś mieć listę, do której będziesz mógł bez problemów dodawać nowe elementy? Chcesz znaleźć coś na podstawie nazwy? Chcesz stworzyć listę, która będzie automatycznie eliminować powtarzające się elementy? Chcesz posortować listę współpracowników według liczby określającej, ile razy znieścą Cię w plecy?



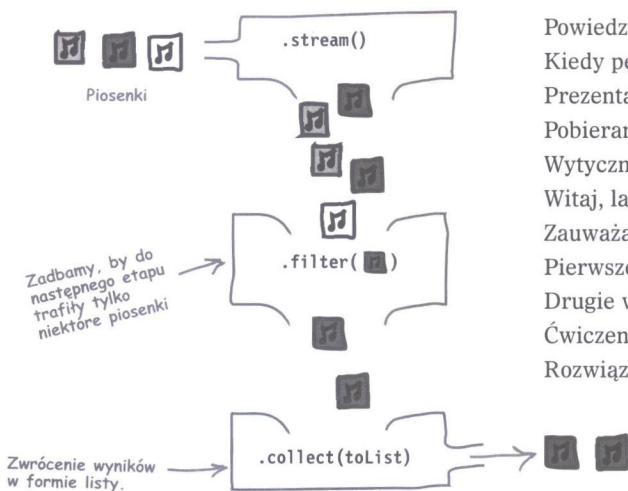
Poznajemy API pakietu java.util, List i Collections	310
Typy ogólne oznaczają większe bezpieczeństwo typów	320
Ponownie odwiedzimy metodę sort()	323
Nowa, poprawiona, porównywalna klasa Piosenka	326
Sortowanie z wykorzystaniem wyłącznie komparatorów	332
Zastosowanie wyrażeń lambda w kodzie aplikacji szafy grającej	338
Wykorzystanie kolekcji HashSet zamiast ArrayList	343
Co MUSIMY wiedzieć o klasie TreeSet...	349
Poznaliśmy już listy i zbiory, teraz poznamy mapy	351
W końcu wracamy do typów ogólnych	354
Rozwiązania ćwiczeń	360

12

Wyrażenia lambda i strumienie: co, a nie jak

A co by było, gdybyś... nie musiał mówić komputerowi, jak coś zrobić?

W tym rozdziale zajmiemy się *API strumieni*. Przekonasz się w nim także, jak przydatne mogą być wyrażenia lambda w połączeniu ze strumieniami, i dowiesz się, jak używać API strumieni do wyszukiwania i przekształcania danych w kolekcjach.

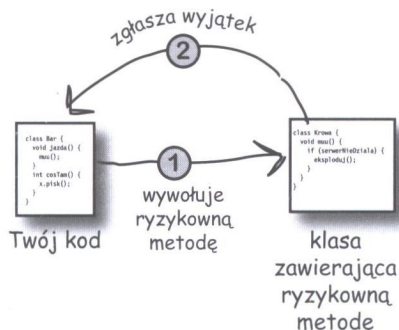


Powiedz komputerowi, CZEGO chcesz	366
Kiedy pętle for schodzą na złą drogę	368
Prezentacja API strumieni	371
Pobieranie wyników ze strumienia	374
Wytczne dotyczące stosowania strumieni	380
Witaj, lambda, nasz (niezbyt) stary przyjacielu	384
Zauważanie interfejsów funkcyjnych	392
Pierwsze wyzwanie Leona: znaleźć wszystkie rockowe utwory	396
Drugie wyzwanie Leona: lista wszystkich gatunków	400
Ćwiczenia	411
Rozwiązania ćwiczeń	413

13

Ryzykowne zachowanie

Różne rzeczy się zdarzają. Pliku nie ma tam, gdzie powinien być. Serwer został wyłączony. Niezależnie od tego, jak dobrym jesteś programistą, nie jesteś w stanie kontrolować *wszystkiego*. Pisząc „ryzykowną” metodę, będziesz potrzebował kodu, który „poradzi” sobie w sytuacji, gdy zdarzy się coś złego. Jednak skąd wiadomo, że metoda jest „ryzykowna”? I gdzie należy umieścić kod przeznaczony do *obsługi* tej **wyjątkowej** sytuacji? W tym rozdziale napiszemy odtwarzacz muzyki MIDI, korzystając z „ryzykownego” JavaSound API, dlatego lepiej będzie, jeśli dowiemy się, jak obsługiwać taki potencjalnie niebezpieczny kod.



Stwórzmy program MuzMachina	418
Przed wszystkim potrzebujemy sekwensera	420
Wyjątek jest obiektem... klasy Exception	424
Sterowanie przepływem w blokach try-catch	428
Czy wspominaliśmy, że metoda może zgłaszać więcej niż jeden wyjątek?	431
W przypadku użycia wielu bloków catch należy je uporządkować ze względu na zakres — od najmniejszego do największego	434
Zrezygnowanie z obsługi wyjątku (poprzez jego zadeklarowanie) jedynie odsuwa w czasie to, co nieuniknione	438
Kod od kuchni	441
Wersja 1. Twój pierwszy odtwarzacz muzyki	444
Wersja 2. Eksperymenty muzyczne z wykorzystaniem argumentów przekazywanych z wiersza poleceń	448
Ćwiczenia	450
Rozwiązania ćwiczeń	453

14

Historia bardzo graficzna

Pogódź się z faktem, że potrzebujesz graficznego interfejsu użytkownika.

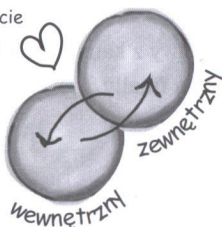
Nawet jeśli wierzysz, że całe życie spędzisz na pisaniu programów wykonywanych po stronie serwera, gdzie funkcję interfejsu użytkownika pełnią strony WWW, to i tak wcześniej czy później będziesz musiał napisać jakiś program narzędziowy i zapewne zechcesz wyposażać go w graficzny interfejs użytkownika. Zagadnieniami związanymi z graficznym interfejsem użytkownika będziemy się zajmować przez dwa rozdziały, a w międzyczasie poznamy także kilka nowych możliwości języka Java, takich jak **obsługa zdarzeń** oraz **klasy wewnętrzne**. Wyświetlimy na ekranie przycisk, narysujemy coś w oknie, wyświetlimy obrazek JPEG, a nawet napiszemy prostą animację.

```

class MojaKlasaZewnetrzna {
    ...
    class MojaKlasaWewnetrzna {
        void doDziela() {
            ...
        }
    }
}
  
```

Klasa wewnętrzna jest całkowicie zawarta w klasie zewnętrznej.

Te dwa obiekty przebywające na stercie łączą szczególna więź. Obiekt wewnętrzny może korzystać ze składowych obiektu zewnętrznego (i na odwrót).



Wszystko zaczyna się od okna	456
Przechwytywanie zdarzeń generowanych przez działania użytkownika	459
Odbiorcy, źródła i zdarzenia	463
Stwórz własny komponent umożliwiający rysowanie	466
Fajne operacje, jakie można realizować w metodzie paintComponent()	467
Układy GUI — wyświetlanie w ramce więcej niż jednego komponentu	472
Klasy wewnętrzne spieszą z pomocą!	478
Wyrażenia lambda spieszą z pomocą (ponownie)!	484
Wykorzystanie klas wewnętrznych do tworzenia animacji	486
Prostszy sposób tworzenia komunikatów-zdarzeń	492
Ćwiczenia	496
Rozwiązania ćwiczeń	501

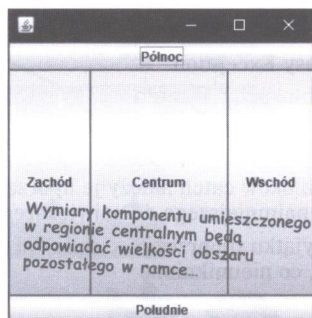
15

Popracuj nad Swingiem

Swing jest łatwy. Chyba że naprawdę *zwracasz uwagę* na to, gdzie zostaną wyświetlone poszczególne elementy interfejsu użytkownika. Kod wykorzystujący bibliotekę Swing *wygląda na prosty*, jednak potem go kompilujesz, uruchamiasz, patrzysz na wyniki i myślisz: „Hej, to nie miało być w tym miejscu”. Elementem, który sprawia, że tworzony kod jest *prosty*, a kontrola położenia elementów *trudna*, jest **menedżer układu**. Jednak przy niewielkim nakładzie pracy można nagiąć menedżer układu do swojej woli. W tym rozdziale „popracujemy nad naszym Swingiem” i dowiemy się także czegoś więcej o komponentach.

Komponenty w regionach wschodnim i zachodnim uzyskują preferowaną szerokość.

Komponenty w regionach północnym i południowym uzyskują preferowaną wysokość.

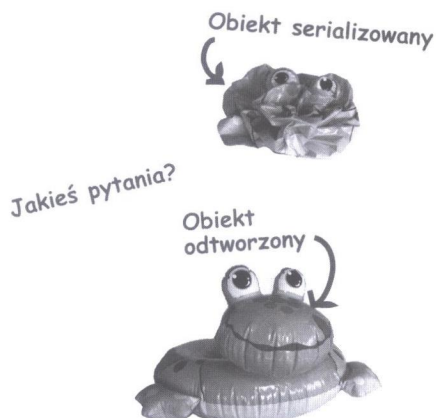


Komponenty biblioteki Swing	504
Menedżery układu	505
Wielka trójka menedżerów układu: BorderLayout, FlowLayout oraz BoxLayout	507
Zabawy z komponentami biblioteki Swing	517
Kod od kuchni	522
Tworzenie aplikacji MuzMachina	523
Ćwiczenia	528
Rozwiązania ćwiczeń	530

16

Zapisywanie obiektów (i tekstu)

Obiekty można „pakować”, a następnie odtwarzać. Obiekty mają swój stan i działanie. Działanie jest określane przez samą *klasę* obiektu, jednak stan określają poszczególne *obiekty*. Jeśli Twój program musi zapisywać stan, to *możesz to zrobić w złożony sposób* — odczytując stan każdego obiektu i pracowicie zapisując wartość każdej zmiennej instancyjnej w pliku w wybranym formacie. Możesz także zrobić to samo w **prosty obiektowy sposób** — po prostu „zapisać-zamrozić-wysuszyć-zachować” sam obiekt, a następnie go „odczytać-odmrozić-namoczyć-odtworzyć”.



Zapisywanie serializowanego obiektu do pliku	534
Jeśli chcesz, aby klasa zapewniała możliwość serializacji, zaimplementuj interfejs Serializable	539
Deserializacja — odtwarzanie obiektów	543
Identyfikator wersji — wielki problem serializacji	548
Zapisywanie łańcucha znaków w pliku tekstowym	551
Odczyt zawartości pliku tekstowego	558
Kwiz — gra (zarys kodu)	559
Typy Path, Paths i Files (zabawy z katalogami)	565
Finalnie, bliższe spojrzenie na finally	566
Zapisywanie kompozycji	571
Ćwiczenia	574
Rozwiązania ćwiczeń	576

17

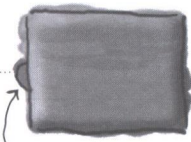
Nawiąż połączenie

Nawiąż połączenie ze światem zewnętrznym. To całkiem łatwe. Wszystkie szczegóły niskiego poziomu związane z komunikacją sieciową są realizowane przez klasy należące do wbudowanej biblioteki Javy. Jedną z ogromnych zalet Javy jest to, że komunikacja sieciowa bardzo przypomina zwyczajną obsługę wejścia-wyjścia, a obie operacje mogą się różnić jedynie obiektami umieszczanymi na samym końcu łańcucha strumieni. W tym rozdziale utworzymy gniazdo klienta. Utworzymy także gniazdo serwera. Napišemy również kod klienta oraz kod serwera. Zanim ten rozdział się skończy, będziesz dysponować w pełni funkcjonalnym, wielowątkowym klientem do obsługi pogawędek internetowych. Czy użyliśmy słowa *wielowątkowy*?

Połączenie
z portem 5000
na serwerze
o adresie
196.164.1.103



Klient



Serwer

Połączenie zwrotne
z klientem
pod adresem
196.164.1.100 na
porcie 4242

Nawiązywanie połączenia, wysyłanie i odbieranie danych	582
Program CodziennePoradyKlient	590
Tworzenie prostej aplikacji serwera	593
Możliwość stosowania wielu wątków w Javie zapewnia jedna klasa — Thread	602
Trzy stany nowego wątku	608
Usypianie wątku	614
Tworzenie i uruchamianie <i>dwóch</i> (lub większej liczby!) wątków	618
Koniec pracy w puli wątków	621
Nowa i poprawiona wersja programu ProstyKlientPogawedek	624
Ćwiczenia	626
Rozwiązania ćwiczeń	628

18

Rozwiązywanie problemów współbieżności

Jednoczesne robienie dwóch lub większej liczby rzeczy jest trudne. Pisanie kodu wielowątkowego nie przysparza problemów. Jednak pisanie kodu wielowątkowego, który będzie działał w oczekiwany sposób, może być znacznie trudniejsze. W tym ostatnim rozdziale książki pokażemy Ci kilka przykładowych problemów, które mogą występować w przypadku korzystania z dwóch lub większej liczby jednocześnie działających wątków. Przy okazji poznasz także wybrane z narzędzi dostępnych w pakiecie `java.util.concurrent`, ułatwiających pisanie poprawnie działającego kodu wielowątkowego. Dowiesz się tu, jak tworzyć obiekty niemodyfikowalne (czyli takie, których stan nie może się zmieniać), których można bezpiecznie używać w wielu jednocześnie działających wątkach. Pod koniec rozdziału będziesz dysponował wieloma różnorodnymi narzędziami do pisania kodu współbieżnego.



iteracja

referencja
do odczytu

935 34 173

zapis
w kopii

935 34 173 5

CopyOnWriteArraySet

Problem Moniki i Roberta w formie kodu	634
Stosowanie blokady obiektu	639
Prerażający problem „utraconej modyfikacji”	642
Zadeklaruj metodę <code>inkrementuj()</code> jako metodę atomową. Synchronizuj ją!	644
Wzajemna blokada, mroczna strona wątków	646
Operacje CAS z użyciem zmiennych atomowych	648
Stosowanie niezmiennych obiektów	651
Więcej problemów ze współużytkowanymi danymi	654
Stosowanie struktur danych bezpiecznych pod względem wielowątkowym	656
Ćwiczenia	660
Rozwiązania ćwiczeń	662