W PROSTOCIE TKWI SIŁA

wydanie VII

Java®

dla bystrzaków

Używaj popularnych narzędzi Javy

Twórz proste obiekty Javy i ponownie używaj kodu

Obsługuj zdarzenia i wyjątki

dr Barry Burd
autor książki *Java Programming for Android Developers for Dummies*

septem
septem.pl
Helion

Tytuł oryginału: Java For Dummies, 7th Edition

Tłumaczenie: Wojciech Moch

ISBN: 978-83-283-5989-5

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://dlabystrzakow.pl/user/opinie/javby7
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *dlabystrzakow@dlabystrzakow.pl*
WWW: *http://dlabystrzakow.pl*

Printed in Poland.

Spis treści

W TYM ROZDZIALE:

- Czym jest język Java
- Skąd się wywodzi język Java
- Dlaczego język Java jest taki niesamowity
- Jak zorientować się w programowaniu obiektowym

Rozdział 1

Klasy w C++

Mów, co chcesz, o komputerach. Jeśli o mnie chodzi, to uważam, że komputery są dobre z dwóch prostych powodów:

- **Gdy komputery pracują, nie odczuwają oporów, stresu, nudy ani zmęczenia.** Komputery są naszymi elektronicznymi niewolnikami. Mój komputer pracuje 24 godziny na dobę, 7 dni w tygodniu, wykonując obliczenia dla *Cosmology@Home* — rozproszonego projektu obliczeniowego, którego celem jest zbadanie modeli opisujących wszechświat. Czy jest mi przykro z powodu ciężkiej pracy mojego komputera? Czy komputer narzeka? Czy komputer zgłosi mnie do Państwowej Inspekcji Pracy? Nie.

  Mogę żądać, wydawać rozkazy komputerowi i strzelać z bicza. Czy czuję się (lub powinienem się czuć) choć odrobinę winny? Ani trochę.

- **Komputery, a nie papier poruszają idee.** Jeszcze nie tak dawno temu, chcąc wysłać komuś wiadomość, trzeba było wynająć posłańca. Posłaniec wsiadł na swojego konia i osobiście dostarczył wiadomość. Wiadomość była zapisana na papierze, pergaminie, glinianej tabliczce lub jakimkolwiek innym materialnym nośniku, który był wtedy dostępny.

  Cały ten proces wydaje się teraz marnotrawstwem czasu, ale to tylko dlatego, że Ty i ja siedzimy wygodnie w erze elektronicznej. Wiadomości są ideami, a fizyczne rzeczy, takie jak atrament, papier i konie, mają niewiele lub nic wspólnego

z prawdziwymi ideami; są tylko ich tymczasowymi nosicielami (pomimo że ludzie przez kilka stuleci używali ich w ten sposób). Niemniej jednak same idee istnieją bez papieru, bez koni i bez posłańców.

Dobrą rzeczą w komputerach jest to, że skutecznie przenoszą idee. Przetwarzają wyłącznie idee, kilka fotonów i trochę energii elektrycznej. Robią to bez żadnych kłopotów i bez żadnego dodatkowego fizycznego bagażu.

Kiedy zaczynasz skutecznie radzić sobie z ideami, dzieje się coś bardzo ciekawego. Nagle znikają wszystkie dodatkowe obciążenia. Zamiast używać papieru i atramentu, tworzysz liczby i koncepcje. Bez dodatkowych obciążeń możesz znacznie szybciej wykonywać swoje prace, które dodatkowo są o wiele bardziej złożone niż kiedykolwiek wcześniej.

Co możesz zrobić za pomocą języka Java

Byłoby miło, gdyby cała ta złożoność była dostępna za darmo, ale niestety tak nie jest. Ktoś musi się mocno zastanowić i zdecydować, o co dokładnie poprosić komputer. Po tych przemyśleniach ktoś musi napisać zestaw instrukcji, tak aby komputer postępował zgodnie z nimi.

Biorąc pod uwagę obecny stan rzeczy, nie możesz napisać tych instrukcji w języku polskim ani w żadnym innym języku naturalnym. Fantastyka naukowa jest pełna historii o ludziach, którzy mówią proste rzeczy robotom i uzyskują katastrofalne, nieoczekiwane rezultaty. Język polski i inne mu podobne języki z kilku powodów nie nadają się do komunikacji z komputerami:

- **Polskie zdanie może zostać źle zinterpretowane.** „Żuj jedną tabletkę trzy razy dziennie, aż do zakończenia".
- **Trudno jest wytworzyć bardzo skomplikowane polecenie w języku polskim.** „Połącz kołnierz A z wypukłością B, upewniając się, że tylko zewnętrzna krawędź kołnierza A jest połączona z większym końcem wypukłości B, jednocześnie łącząc środkowe i wewnętrzne krawędzie kołnierza A z przelotką C".
- **Zdanie polskie ma dużo dodatkowego bagażu.** „Zdanie zawiera niepotrzebne słowa".
- **Język polski jest trudny do zinterpretowania.** „W ramach niniejszej umowy wydawniczej między Johnem Wiley & Sons, Inc. (Wiley) a autorem (Barry Burd) Wiley zapłaci kwotę tysiąca dwustu pięćdziesięciu siedmiu złotych i sześćdziesięciu trzech groszy (1257,63 zł) na rzecz autora za częściowe przesłanie książki *Java dla bystrzaków* (dzieło)".

Aby powiedzieć komputerowi, co ma robić, musisz użyć specjalnego języka składającego się ze zwięzłych, jednoznacznych instrukcji. Specjalny język tego rodzaju nazywany jest *językiem programowania komputera*. Zestaw instrukcji napisanych w takim języku nazywany jest *programem*. Gdy potraktujemy je jak wielki zbiór instrukcji, to będziemy mogli nazywać je *oprogramowaniem* lub *kodem*. Oto jak wygląda taki kod, gdy jest napisany w języku Java:

```
public class PayBarry {
    public static void main(String args[]) {
        double checkAmount = 1257.63;
        System.out.print("Zapłać za rachunek od ");
        System.out.print("dr. Barry'ego Burda ");
        System.out.print("PLN");
        System.out.println(checkAmount);
    }
}
```

Dlaczego warto używać języka Java

Czas świętować! Masz już w ręce kopię książki *Java dla bystrzaków* i czytasz 1. rozdział. W tym tempie w krótkim czasie staniesz się ekspertem w dziedzinie języka Java[1], więc ciesz się z ewentualnego sukcesu, rzucając się w wir wielkiej zabawy.

Aby przygotować się do imprezy, trzeba będzie upiec ciasto. Jestem leniwy, więc użyję gotowej mieszanki do pieczenia ciasta. Zobaczmy... dodaj wodę do mieszanki, a następnie dodaj masło i jajka — hej, zaczekaj! Właśnie spojrzałem na listę składników. Co to jest MSG? A co to jest glikol propylenowy? Te składniki są chyba używane w płynach przeciw zamarzaniu, prawda?

Jednak zmienię plany i zrobię ciasto samodzielnie. Oczywiście, jest trochę trudniej, ale w ten sposób otrzymuję dokładnie to, czego chcę.

Programy komputerowe działają w ten sam sposób. Możesz użyć cudzego programu lub napisać własny. Jeśli korzystasz z cudzego programu, używasz wszystkiego, co jest w nim zawarte. Kiedy piszesz własny program, możesz dostosować program specjalnie do swoich potrzeb.

Pisanie kodu komputerowego to duży światowy przemysł. Robią to firmy, niezależni specjaliści, robią to hobbyści — robi to wiele różnych ludzi. Typowa duża firma posiada zespoły, działy i departamenty, które piszą dla niej programy. Ale możesz pisać programy dla siebie lub kogoś innego, dla zarobku lub dla zabawy.

[1] W kręgach zawodowych obowiązki twórcy oprogramowania są zazwyczaj szersze niż obowiązki samego programisty. Ale w tej książce używam terminów programista i twórca oprogramowania niemalże zamiennie.

Według ostatnich szacunków liczba wierszy kodu pisanego każdego dnia przez samych programistów w Stanach Zjednoczonych przekracza liczbę cząsteczek metanu na planecie Jowisz.[2] Zastanów się, co można zrobić przy użyciu komputera. Mając wystarczająco dużo czasu, możesz napisać własny program realizujący wybrane zadanie. (Oczywiście „wystarczająco dużo czasu" może okazać się bardzo długim okresem, ale nie o to chodzi. Wiele ciekawych i przydatnych programów można napisać w ciągu kilku godzin lub nawet minut).

Informacje historyczne: skąd pochodzi Java

Oto krótka historia nowoczesnego programowania komputerowego:

- **1954 – 1957 — powstaje język FORTRAN.**

  FORTRAN był pierwszym nowoczesnym językiem programowania komputerowego. Jeżeli chodzi o programowanie naukowe, to FORTRAN jest naprawdę niedościgniony. Z roku na rok FORTRAN stawał się coraz bardziej wiodącym językiem wśród programistów komputerowych na całym świecie.

- **1959 — Grace Hopper w Remington Rand tworzy język programowania COBOL.**

  Litera B w nazwie języka COBOL oznacza *biznes*, i to na nim koncentruje się ten język. Podstawową cechą tego języka jest przetwarzanie jednego rekordu za drugim, jednego klienta po drugim lub jednego pracownika po drugim.

  W ciągu kilku lat od swojego powstania COBOL stał się najczęściej używanym językiem do przetwarzania danych biznesowych.

- **1972 — Dennis Ritchie z AT&T Bell Labs opracowuje język programowania C.**

  Charakter kodu widocznego w przykładach z tej książki pochodzi z języka programowania C. W kodzie napisanym w języku C używa się nawiasów klamrowych, instrukcji `if`, `for` i tak dalej.

  Jeśli chodzi o zakres zastosowań, to możesz użyć języka C, aby rozwiązać te same problemy, które możesz także rozwiązać przy użyciu języka FORTRAN, Java lub dowolnego innego nowoczesnego języka programowania. (Możesz na przykład napisać program kalkulatora naukowego w języku COBOL, ale to byłoby już naprawdę dziwne). Różnica między jednym językiem programowania a innym nie polega zatem na ich możliwościach. Różnica polega na łatwości używania danego języka i poziomie dopasowania go do danego zadania. Tutaj właśnie wyróżnia się język Java.

[2] Sam to wymyśliłem.

- **1986 — Bjarne Stroustrup (ponownie w AT&T Bell Labs) opracowuje C++.**

  W przeciwieństwie do swojego przodka (języka C) język C++ pozwala na programowanie obiektowe. Ta nowa możliwość stanowi ogromny krok naprzód (zobacz następny podrozdział).

- **23 maja 1995 — Sun Microsystems wydaje pierwszą oficjalną wersję języka programowania Java.**

  Java ulepsza jeszcze koncepcje zawarte w języku C++. Filozofia języka Java „napisz raz, uruchom gdziekolwiek" sprawia, że język jest idealny do rozpowszechniania kodu w internecie.

  Ponadto Java jest doskonałym językiem programowania ogólnego przeznaczenia. Dzięki Javie możesz pisać aplikacje okienkowe, tworzyć i przeszukiwać bazy danych, kontrolować urządzenia przenośne i jeszcze o wiele, wiele więcej. W ciągu pięciu krótkich lat język programowania Java miał 2,5 miliona programistów na całym świecie. (Wiem. Mam na dowód pamiątkową koszulkę).

- **Listopad 2000 — College Board ogłasza, że począwszy od roku 2003, egzaminy Advanced Science z Computer Science będą oparte na Javie.**

  Chcesz wiedzieć, czego ten nudziarz mieszkający na końcu ulicy uczy się w szkole średniej? Zgadłeś — języka Java.

- **2002 — Microsoft wprowadza nowy język o nazwie C#.**

  Wiele funkcji języka C# pochodzi bezpośrednio z języka Java.

- **Czerwiec 2004 — Sys-Con Media donosi, że popyt na programistów Javy przewyższa popyt na programistów C++ o 50% (*http://java.sys-con.com/node/48507*).**

  I to nie wszystko! Zapotrzebowanie na programistów języka Java o 8% przekracza łączne zapotrzebowanie na programistów języka C++ i C#. Programiści Javy są o 190% bardziej pożądani niż programiści języka Visual Basic (VB).

- **2007 — Google przyjmuje język Java jako podstawowy język do tworzenia aplikacji na urządzeniach mobilnych z Androidem.**

- **Styczeń 2010 — Oracle Corporation kupuje Sun Microsystems, wprowadzając technologię Java do rodziny produktów Oracle.**

- **Czerwiec 2010 — *eWeek* plasuje Javę na pierwszym miejscu listy „10 najpopularniejszych języków programowania pozwalających utrzymać zatrudnienie" (*www.eweek.com/c/a/Application-Development/Top-10-Programming-Languages-to-Keep-You-Employed-719257*).**

- **2016 — Java działa na 15 miliardach urządzeń (*http://java.com/en/about*), a Android Java działa na 87,6% wszystkich telefonów komórkowych na całym świecie (*www.idc.com/prodserv/smartphone-os-market- share.jsp*).**

Ponadto technologia Java zapewnia interaktywne możliwości wszystkim urządzeniom Blu-ray i jest najpopularniejszym językiem programowania w Indeksie Społeczności Programistycznej (*Programming Community Index*) TIOBE (*www.tiobe.com/index.php/content/paperinfo/tpci*), na PYPL: indeks popularności języków programowania (*PopularitY of Programming Language Index*) (*http://sites.google.com/site/pydatalog/pypl/PyPL-PopularitY-of-Programming-Language*) oraz na innych indeksach.

Jestem pod wrażeniem.

Programowanie obiektowe (OOP)

Jest trzecia rano. Śnię o egzaminie z historii w szkole średniej, który udało mi się oblać. Nauczyciel krzyczy na mnie: „Masz dwa dni na naukę do sprawdzianu końcowego, ale nie pamiętasz o tym, żeby się uczyć. Zapomnisz o tym i poczujesz się winny, winny, winny".

Nagle dzwoni telefon. Obudziłem się wyrwany z głębokiego snu. (Jasne, nie podobał mi się sen o egzaminie z historii, ale jeszcze bardziej nie lubię się z nagła obudzić). Po pierwsze upuszczam telefon na podłogę. Po jego niezdarnym podniesieniu wydaję zrzędliwe: „Cześć, kto to?". Głos odpowiada: „Jestem reporterem z »New York Timesa«. Piszę artykuł o Javie i muszę wiedzieć wszystko o tym języku programowania w pięciu lub mniej słowach. Możesz nam przybliżyć ten temat?".

Mój umysł jest zbyt zamglony. Nie mogę myśleć. Mówię więc pierwszą rzecz, która przychodzi mi do głowy, a potem wracam spać.

Rano prawie nie pamiętam rozmowy z reporterem. Nie pamiętam nawet, w jaki sposób odpowiedziałem na to pytanie. Czy powiedziałem reporterowi, gdzie może sobie wsadzić ten artykuł o Javie?

Zakładam wdzianko i wybiegam na podjazd przed domem. Gdy zabieram poranną gazetę, spoglądam na pierwszą stronę i widzę nagłówek w rozmiarze dwóch cali:

Burd nazywa język Java „świetnym językiem obiektowym".

Języki obiektowe

Java jest zorientowana obiektowo. Co to znaczy? W przeciwieństwie do języków takich jak FORTRAN, które koncentrują się na wydawaniu komputerowi poleceń „Zrób to, zrób tamto", języki obiektowe skupiają się na danych. Oczywiście programy zorientowane obiektowo nadal informują komputer, co ma robić, ale zaczynają jednak od uporządkowania danych, a polecenia pojawiają się później.

Języki zorientowane obiektowo są lepsze od języków „Zrób to, zrób tamto", ponieważ organizują dane w sposób, który pomaga ludziom zrobić z nimi różne rzeczy. Aby zmodyfikować dane, możesz korzystać z tego, co już masz, zamiast wyrzucać wszystko, co zrobiłeś, i zaczynać od nowa za każdym razem, gdy musisz zrobić coś nowego. Chociaż programiści komputerowi są zazwyczaj inteligentnymi ludźmi, trochę czasu zajęło im rozgryzienie tego tematu. Pełna lekcja historii znajduje się w ramce „Kręta droga z FORTRAN-a do Javy" (ale nie będę mieć Ci za złe, jeśli jej nie przeczytasz).

KRĘTA DROGA Z FORTRAN-A DO JAVY

W połowie lat 50. XX wieku zespół ludzi stworzył język programowania o nazwie FORTRAN. Był to dobry język, ale opierał się na idei, że należy wydawać komputerowi bezpośrednie, imperatywne polecenia. „Komputer, zrób to. Komputer, zrób tamto". (Oczywiście komendy w prawdziwym programie FORTRAN były znacznie bardziej precyzyjne niż takie ogólne „zrób to" lub „zrób tamto").

W kolejnych latach zespoły programistów opracowały wiele nowych języków programowania komputerów, a wiele z tych języków skopiowało znany z FORTRAN-a model „zrób to, zrób tamto". Jeden z bardziej popularnych języków z tej grupy otrzymał jednoliterową nazwę — C. Oczywiście obóz „zrób to, zrób tamto" miał kilku odszczepieńców. W językach o nazwach SIMULA i Smalltalk programiści przenieśli niezbędne polecenia „zrób to" na drugi plan i skoncentrowali się na opisie danych. W tych językach nie można było bezpośrednio powiedzieć: „Wydrukuj mi listę zaległych rachunków", zamiast tego trzeba było zaczynać od stwierdzenia: „Tak wygląda konto. Konto ma swoją nazwę i saldo". Następnie można było powiedzieć: „A tak można zapytać konto, czy ma ono zaległości". Nagle dane stały się najważniejsze. Konto było czymś, co miało nazwę, saldo i odpowiedni sposób na przekazanie Ci informacji dotyczących zaległych płatności.

Języki, które skupiają się przede wszystkim na danych, nazywane są *językami programowania obiektowego*. Stanowią one doskonałe narzędzia programistyczne. Oto lista powodów, dlaczego tak jest:

- Jeżeli najpierw skupiasz się na danych, to znaczy, że jesteś dobrym programistą komputerowym.
- Możesz w kółko rozszerzać i ponownie wykorzystywać opisy danych. Kiedy próbujesz nauczyć stare programy FORTRAN nowych sztuczek, pokazują one, jak bardzo są kruche. Po prostu psują się.

W latach 70. XX wieku języki zorientowane obiektowo, takie jak na przykład SIMULA i Smalltalk, wspominane były jedynie w artykułach hobbystycznego magazynu komputerowego. Natomiast w międzyczasie języki oparte na starym modelu FORTRAN rozmnażały się jak króliki.

Jednak w 1986 roku człowiek o nazwisku Bjarne Stroustrup stworzył język o nazwie C++. Stał się on bardzo popularny, ponieważ zmieszał starą terminologię języka C z ulepszoną strukturą obiektową. Wiele firm odwróciło się od starego stylu programowania FORTRAN/C i przyjęło język C++ jako swój standard.

Ale język C++ miał pewną wadę. Używając go, można pominąć wszystkie funkcje obiektowe i napisać program przy użyciu starego stylu programowania FORTRAN/C. Podczas pisania programu księgowego w języku C++ możesz wybrać jedną z dróg:

- Zacząć od wydawania bezpośrednich poleceń komendami „zrób to", tworząc matematyczny odpowiednik polecenia: „Wydrukuj listę kont z zaległościami i zrób to szybko".
- Wybrać podejście zorientowane obiektowo i opisać najpierw, jak ma wyglądać konto.

Niektórzy mówili, że programowanie w języku C++ oferuje najlepsze z obu światów, ale inni twierdzili, że pierwszy świat (świat FORTRAN i C) nie powinien być częścią nowoczesnego programowania. Gdybyś dał programistom możliwość pisania kodu w dowolny sposób, zbyt często zdecydowaliby się na napisanie go w sposób niewłaściwy.

W 1995 roku James Gosling z firmy Sun Microsystems stworzył język o nazwie *Java*. Tworząc Javę, Gosling zapożyczył wygląd języka C++, ale większość jego starych funkcji „zrób to, zrób tamto" wrzucił do kosza. Następnie dodał funkcje, dzięki którym tworzenie obiektów było znacznie łatwiejsze. W sumie Gosling stworzył język, którego filozofia obiektowa jest przejrzysta i czysta. Kiedy programujesz w Javie, nie masz innego wyboru; musisz pracować z obiektami. I tak właśnie powinno być.

Obiekty i ich klasy

W języku zorientowanym obiektowo do porządkowania danych używa się obiektów *oraz* klas.

Wyobraź sobie, że piszesz program komputerowy przechowujący informacje o domach w nowej inwestycji domków własnościowych (wciąż w budowie). Domy różnią się tylko nieznacznie od siebie, każdy z nich ma charakterystyczny kolor elewacji, kolor ścian wewnętrznych oraz styl szafek kuchennych i tak dalej. W Twoim zorientowanym obiektowo programie komputerowym każdy dom jest obiektem.

Ale same obiekty to nie wszystko. Chociaż domy różnią się nieco od siebie, wszystkie mają tę samą listę wspólnych cech. Na przykład każdy dom ma cechę zwaną *kolorem elewacji*, każdy z nich ma też cechę zwaną *stylem szafek kuchennych*. W Twoim programie obiektowym potrzebna jest główna lista zawierająca wszystkie te cechy, które może mieć obiekt domu. Ta główna lista cech nazywa się *klasą*.

A to dopiero! Programowanie obiektowe otrzymało błędną nazwę. Powinno być nazywane „programowaniem za pomocą klas i obiektów".

Teraz zauważ, że najpierw użyłem słowa *klasa*. Dlaczego to zrobiłem? Wbrew pozorom jeszcze nie oszalałem. Pomyśl o tym, jak budowany jest nowo powstający dom. Gdzieś w okolicy w rozklekotanej przyczepie zaparkowanej przy drodze znajduje się główna lista cech znanych jako projekt. Projekt architekta przypomina klasę programisty stosującego rozwiązania obiektowe Projekt to lista cech, które będzie miał każdy dom. W projekcie zapisana jest „elewacja", a rzeczywisty obiekt domu ma szarą elewację. W projekcie znajdziesz element „szafka kuchenna", a rzeczywisty obiekt domu ma szafki kuchenne w stylu Ludwika XIV.

Analogia nie kończy się na listach cech. Istnieje jeszcze jedna ważna paralela między planami a klasami. Rok po utworzeniu projektu możesz użyć go do budowy dziesięciu domów. To samo dotyczy klas i obiektów. Najpierw programista pisze kod opisujący klasę. Po uruchomieniu programu komputer tworzy obiekty na podstawie klasy (projektu).

To jest prawdziwy związek między klasami i obiektami. Programista definiuje klasę, a na jej podstawie komputer tworzy poszczególne obiekty.

Co jest takiego dobrego w języku zorientowanym obiektowo?

Korzystając z opowieści o budowie domu z poprzedniego punktu, wyobraź sobie, że napisałeś już program komputerowy, który śledzi kolejne kroki budowy domów na nowym osiedlu. I wtedy ważny szef firmy decyduje o zmodyfikowaniu planu budowy, tak że połowa z domów będzie miała trzy sypialnie, a druga połowa będzie wyposażona w cztery.

Jeśli używasz starego stylu programowania komputerowego FORTRAN/C, Twoje instrukcje będą wyglądały tak:

```
Wykop ziemię pod fundamenty.
Wylej beton na ławy fundamentów.
Połóż dwa czworoboki wzdłuż boków ław fundamentów.
...
```

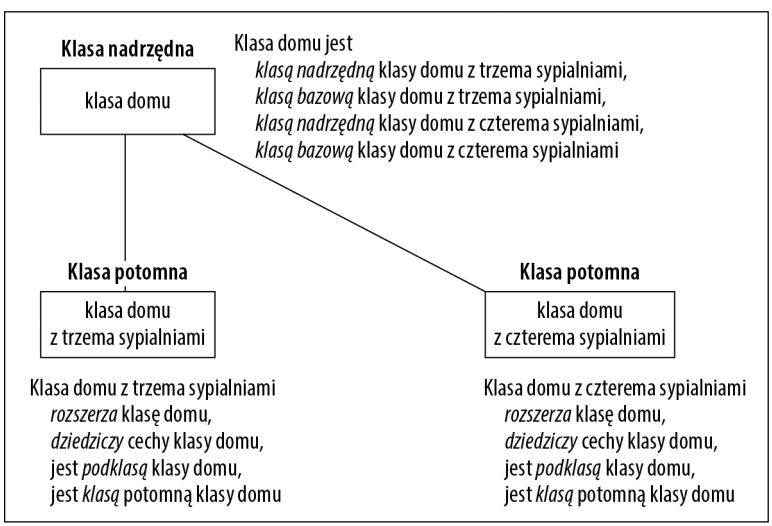
Wyglądałoby to tak, jakby architekt tworzył długą listę instrukcji zamiast narysować plan. Aby znaleźć instrukcje dotyczące budowania sypialni, musisz zmodyfikować plan budowy, sortując listę kolejnych kroków. Co gorsza, poszczególne instrukcje mogą się znajdować na stronach 234, 394 – 410, 739, 10 i 2. Jeśli budowlaniec musiałby rozszyfrować skomplikowane instrukcje innych ludzi, zadanie byłoby dziesięć razy trudniejsze.

Rozpoczęcie prac nad programem od przygotowania klasy jest podobne do rozpoczęcia budowy domu od przygotowania projektu. Jeśli zdecydujesz się budować domy z trzema i czterema sypialniami, możesz zacząć od ogólnego projektu zwanego planem domu, na którym znajdzie się rysunek parteru i piętra, ale nie będzie rozrysowanych na piętrze ścian wewnętrznych. Następnie tworzysz dwa projekty piętra — jeden dla domu z trzema sypialniami, a drugi dla domu z czterema sypialniami. (Nazywasz te nowe plany *domem z trzema sypialniami* i *domem z czterema sypialniami*).

Twoi koledzy budowlańcy są zdumieni Twoim poczuciem logiki i organizacji, ale mają zastrzeżenia. I zadają pytanie: „Nazwałeś jeden z projektów domem z trzema sypialniami. To się nie zgadza. Ale to jest przecież projekt samego piętra, a nie całego domu?".

Mądrze się uśmiechasz i mówisz: „Zauważcie, że w projekcie domu z trzema sypialniami znajduje się notatka: »informacje na temat dolnych pięter znajdują się w oryginalnym projekcie domu«. W ten sposób projekt domu z trzema sypialniami staje się kompletnym projektem całego domu. W projekcie domu z czterema sypialniami znajdziecie taką samą notatkę. Dzięki tej konfiguracji możemy wykorzystać całą wykonaną wcześniej pracę, aby opracować kompletny projekt domu i zaoszczędzić mnóstwo pieniędzy".

W języku programowania obiektowego klasy domów z trzema i czterema sypialniami *dziedziczą* cechy oryginalnej klasy domu. Można również powiedzieć, że klasy domów z trzema i czterema sypialniami *rozszerzają* pierwotną klasę domów (patrz rysunek 1.1).



RYSUNEK 1.1. Terminologia w programowaniu obiektowym

Oryginalna klasa domu nazywa się *klasą nadrzędną* klas domów trzy- i czteropokojowych. Podążając tym tropem, można powiedzieć, że klasy domów z trzema i czterema sypialniami są *podklasami* oryginalnej klasy domu. Innymi słowy, oryginalna

klasa domu nazywa się *klasą bazową* klas domów trzy- i czteroosobowych. Klasy domów z trzema i czterema sypialniami są *klasami potomnymi* pierwotnej klasy domu (patrz rysunek 1.1).

Nie trzeba dodawać, że Twoi koledzy budowlańcy są zazdrośni. Cały tłum ludzi kręci się wokół Ciebie, aby usłyszeć o Twoich wspaniałych pomysłach. W tym momencie dorzucasz jeszcze jedną rewelację: „Tworząc klasę z podklasami, możemy w przyszłości ponownie wykorzystać nasz projekt. Jeśli ktoś w przyszłości zażyczy sobie projektu domu z pięcioma sypialniami, możemy rozszerzyć nasz oryginalny plan domu, tworząc wersję z pięcioma sypialniami. Nigdy więcej nie będziemy musieli wydawać pieniędzy na oryginalny plan domu".

„Ale co się stanie, jeśli ktoś zechce inny projekt parteru? — pyta kolega z tylnego rzędu. — Będzie trzeba wyrzucić oryginalny plan domu, czy zaczniemy poprawiać gotowy, oryginalny projekt? To będzie sporo kosztować, prawda?"

Pewnym tonem odpowiadasz: „Nie musimy nawet dotykać oryginalnego projektu domu. Jeśli ktoś chce jacuzzi w swoim salonie, możemy stworzyć nowy, mały projekt opisujący tylko nowy salon i nazwać go *projektem domu z jacuzzi*. Ten nowy projekt może odwoływać się do oryginalnego projektu domu, pobierając z niego informacje o pozostałej części domu (części, która nie znajduje się w salonie)". W języku programowania obiektowego projekt domu z jacuzzi w salonie nadal *rozszerza* oryginalny plan domu. Projekt domu z jacuzzi jest nadal podklasą oryginalnego planu domu. Nadal obowiązuje tu cała terminologia dotycząca klasy nadrzędnej, klasy bazowej i klasy potomnej. Jedyną nowością jest to, że projekt domu z jacuzzi *zastępuje* funkcje salonu w oryginalnym projekcie domu.

W czasach poprzedzających języki obiektowe świat programowania doświadczył kryzysu w rozwoju oprogramowania. Programiści pisali kod, następnie pojawiały się nowe potrzeby, przez co byli zmuszeni zniszczyć swój kod i zacząć od zera. Ten problem wciąż się powtarzał, ponieważ kod, który pisali programiści, nie mógł być ponownie użyty. Programowanie zorientowane obiektowo zmieniło to wszystko na lepsze (a jak powiedział Burd, Java to „świetny język obiektowy").

Lepsze zrozumienie klas i obiektów

Podczas programowania w Javie stale pracujesz z klasami i obiektami. Te dwie idee są naprawdę ważne. To dlatego w tym rozdziale będę prezentował kolejne analogie opisujące klasy i obiekty.

Zamknij na chwilę oczy i pomyśl, co to znaczy, że coś jest krzesłem...

Krzesło ma siedzisko, podparcie pleców i nogi. Każde siedzisko ma kształt, kolor, stopień miękkości i tak dalej. To są właściwości każdego krzesła. To, co o czym teraz piszę, to *istota krzesła* — opis czegoś, co jest krzesłem. Stosując terminologię obiektową, opisuję tutaj klasę `Chair`.

Teraz odłóż na chwilę tę książkę i rozejrzyj się po swoim pokoju. (Jeśli nie siedzisz teraz w swoim pokoju, spróbuj sobie go wyobrazić).

W pokoju jest kilka krzeseł, a każde krzesło jest obiektem. Każdy z tych obiektów jest przykładem tej eterycznej rzeczy zwanej klasą `Chair`. Tak to działa — klasa jest opisem *istoty krzesła*, a każde krzesło jest obiektem.

Klasa nie jest jedynie zbiorem elementów. W pewnym sensie jest ona ideą pewnego rodzaju rzeczy. Kiedy mówię o klasie krzeseł z Twojego pokoju, mówię o tym, że każde z nich ma nogi, siedzisko, kolor i tak dalej. Każde z krzeseł znajdujących się w pokoju może mieć inny kolor, ale to nie ma znaczenia. Kiedy mówisz o klasie rzeczy, skupiasz się na właściwościach, które ma każda z tych rzeczy.

Sensowne w tym miejscu jest myślenie o obiekcie jako konkretnym egzemplarzu danej klasy. I rzeczywiście, oficjalna terminologia jest zgodna z tym tokiem myślenia. Jeśli napiszesz program w języku Java, w którym to zdefiniujesz klasę `Chair`, każde rzeczywiste krzesło (krzesło, na którym siedzisz, puste krzesło tuż obok itd.) nazywane będzie *instancją* klasy `Chair`.

A oto inny sposób myślenia o klasie. Wyobraź sobie tablicę wyświetlającą dane Twoich trzech kont bankowych (patrz tabela 1.1).

TABELA 1.1. Tabela kont

Numer konta	Typ konta	Saldo konta
16-13154-22864-7	Konto główne	174,87
1011 1234 2122 0000	Kredyt	–471,03
16-17238-13344-7	Oszczędności	247,38

Możesz myśleć o zbiorze nagłówków kolumn tabeli jak o klasie, a o każdym wierszu tabeli jak o obiekcie. Nagłówki kolumn tabeli opisują klasę `Account`.

Zgodnie z nagłówkami kolumn tabeli każde konto ma numer konta, typ konta i saldo. Definiując to w terminologii programowania zorientowanego obiektowo, każdy obiekt klasy `Account` (tj. każda instancja klasy `Account`) ma numer konta, typ konta i saldo konta. Tak więc dolny wiersz tabeli jest obiektem o numerze konta *16-17238-13344-7*, ma on typ *Oszczędności* i saldo *247,38*. Po otworzeniu nowego konta mielibyśmy kolejny obiekt, a tabela powiększyłaby się o dodatkowy wiersz. Taki nowy obiekt byłby instancją tej samej klasy `Account`.

Co dalej?

Ten rozdział wypełniony jest ogólnymi opisami różnych rzeczy. Ogólny opis jest dobry w momencie, gdy dopiero zaczynasz poznawać tę tematykę, ale tak naprawdę rozumiesz to wszystko po poznaniu dodatkowych szczegółów. Dlatego w następnych kilku rozdziałach będę omawiał te szczegóły.

Proszę więc, przewróć tę stronę. Następny rozdział już się nie może doczekać, aż go przeczytasz.


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

Szybko naucz się Javy!

Java jest wszędzie, a rosnący popyt na aplikacje WWW i dla Androida sprawia, że programiści Javy są poszukiwani bardziej niż kiedykolwiek! Ten świetny podręcznik opisuje najważniejsze elementy tego języka, takie jak powłoka JShell. W książce znajdziesz też praktyczne pytania i ćwiczenia, które pomogą Ci rozwinąć umiejętności programowania w Javie. Dzięki prostym instrukcjom obsługiwania klas i metod języka Java, stosowania zmiennych oraz sterowania przepływem programu szybko staniesz się ekspertem programowania w Javie!

W książce:

- Podstawowe elementy Javy
- Wszystko o powłoce JShell
- Wskazówki dotyczące pętli
- Używanie klas i obiektów
- Żonglowanie wartościami w tablicach
- Obsługa interfejsów w Javie

Dr Barry Burd

— profesor nauk komputerowych na Drew University. Autor książek *Java Programming for Android Developers for Dummies* oraz *Android Application Development All-in-One for Dummies*. Pisze artykuły dla witryn Server Side (*theserverside.com*), Android Authority (*androidauthority.com*), InfoQ (*www.infoq.com*) oraz wielu innych.

dla bystrzaków

Zamówienia telefoniczne:

0 801 339900 0 601 339900

septem
septem.pl

Sprawdź najnowsze promocje:
http://dlabystrzakow.pl/promocje
Książki najchętniej czytane:
http://dlabystrzakow.pl/bestsellery
Zamów informacje o nowościach:
http://dlabystrzakow.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: rady@dlabystrzakow.pl
http://dlabystrzakow.pl

Helion

Cena 69,00 zł

ISBN 978-83-283-5989-5

9 788328 359895