Spis treści (podsumowanie)

Spis treści (z prawdziwego zdarzenia)

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?

1 Przełamując zalew początkowych trudności

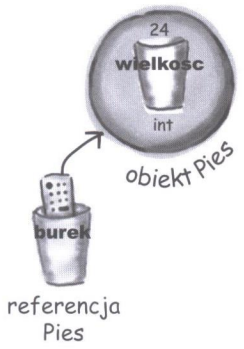
Java zabiera nas w nowe miejsca. Od momentu pojawienia się pierwszej, skromnej wersji o numerze 1.02 Java pociągała programistów ze względu na przyjazną składnię, cechy obiektowe, zarządzanie pamięcią, a przede wszystkim obietnicę przenośności. Wskoczymy od razu na głęboką wodę: napiszemy prosty program 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!

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.

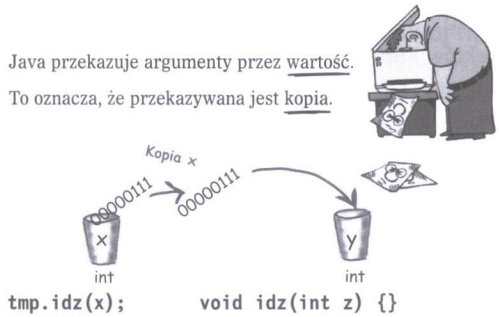
3 Poznaj swoje zmienne

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.



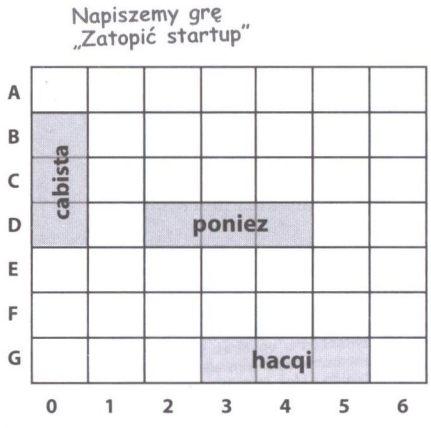
4 Jak działają obiekty?

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 pozmieniajmy stan.***



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.

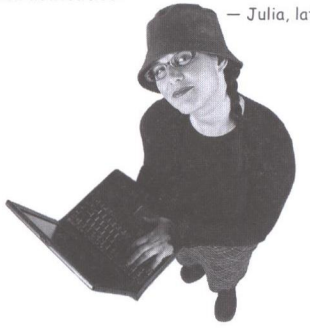


6 Korzystanie z biblioteki Javy

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.

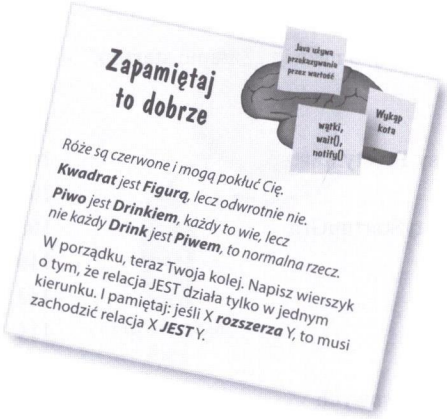
„Dobrze wiedzieć, że w pakiecie java.util istnieje klasa ArrayList. Ale jak mogłabym się o tym ***sama*** *dowiedzieć?"*

— Julia, lat 31, modelka



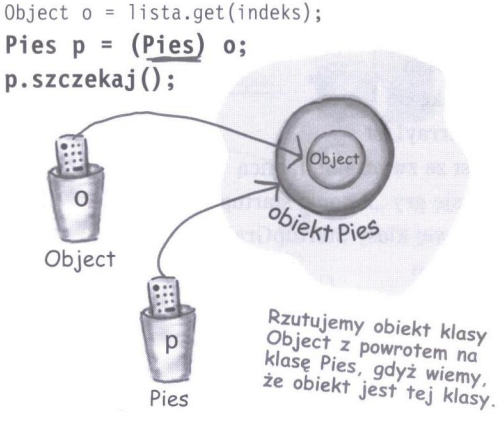
7 Wygodniejsze życie w Obiektowie

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.



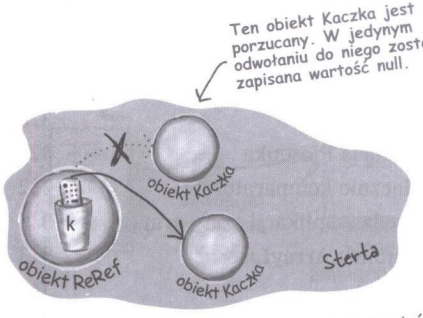
8 Poważny polimorfizm

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.



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.

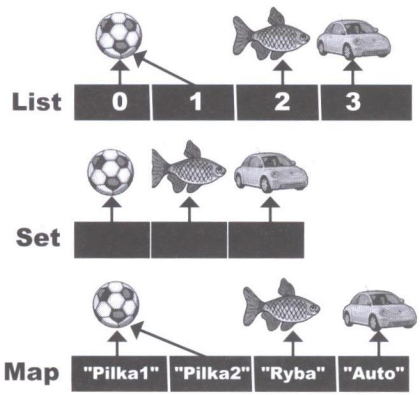
10 Liczby mają znaczenie

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 dziesiętnymi 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.

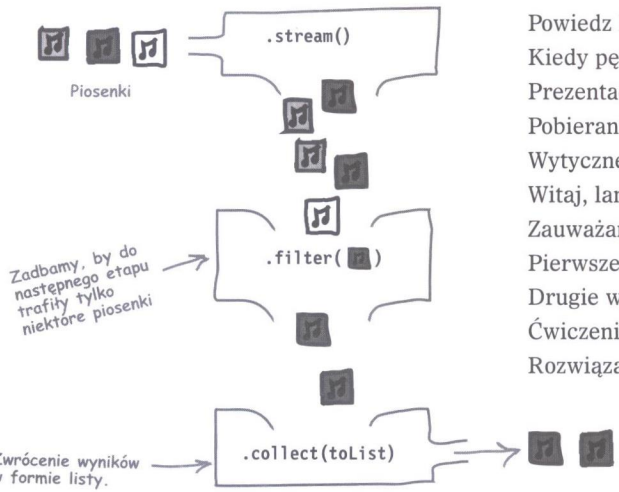
11 Struktury danych

Przechowywanie w Javie nie nastręcza 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 znienacka uderzyli Cię w plecy?



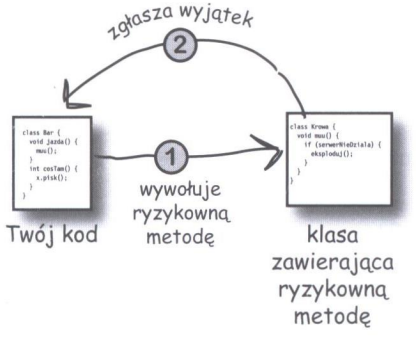
12 Wyrażenia lambda i strumienie: co, a nie jak

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.



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.

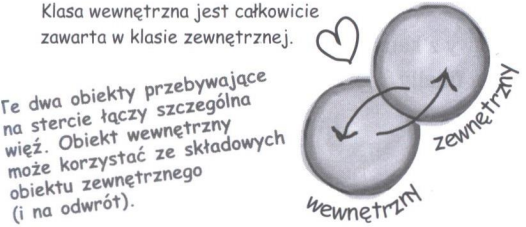


14 Historia bardzo graficzna

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 wyposażyć 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.

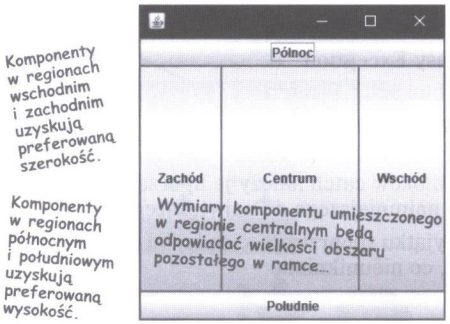
```
class MojaKlasaZewnetrzna {

    class MojaKlasaWewnetrzna {
        void doDziela() {
        }
    }

}
```

Klasa wewnętrzna jest całkowicie zawarta w klasie zewnętrznej.



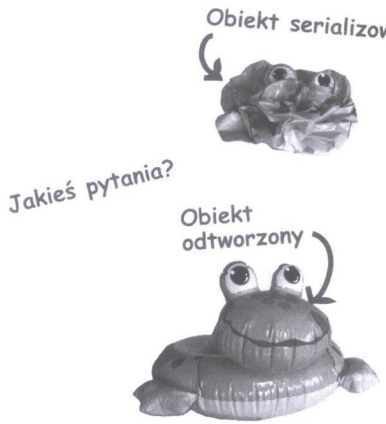
15 Popracuj nad Swingiem

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.



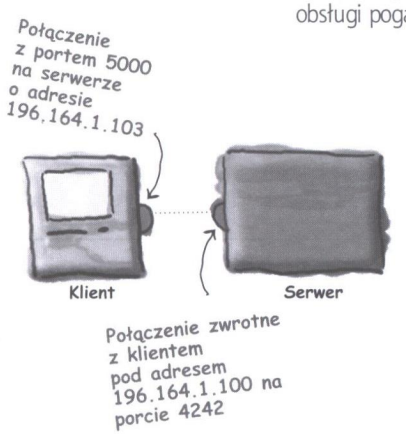
16 Zapisywanie obiektów (i tekstu)

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".



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. Napiszemy 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*?



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.