Kotlin w akcji

Dmitry Jemerov, Svetlana Isakova

Helion

Tytuł oryginału: Kotlin in Action

Tłumaczenie: Andrzej Watrak

Projekt okładki: Studio Gravite / Olsztyn; Obarek, Pokoński, Pazdrijowski, Zaprucki
Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock Images LLC.

ISBN: 978-83-283-4720-5

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

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

Printed in Poland.

Spis treści

1 Kotlin: co to jest i po co?

Ten rozdział opisuje:

- podstawy języka Kotlin,
- najważniejsze cechy języka,
- tworzenie aplikacji serwerowych i aplikacji dla systemu Android,
- różnice w porównaniu z innymi językami,
- tworzenie i uruchamianie kodu.

Czym jest Kotlin? To nowy język programowania przeznaczony dla platformy JVM. Jest zwięzły, pragmatyczny i ukierunkowany na współpracę z kodem Java. Można go stosować niemal wszędzie tam, gdzie dzisiaj jest używana Java, tj. do tworzenia aplikacji serwerowych, aplikacji dla systemu Android i w wielu innych miejscach. Kod w języku Kotlin doskonale współpracuje ze wszystkimi bibliotekami i platformami języka Java i jest równie wydajny. W tym rozdziale szczegółowo opisujemy najważniejsze cechy Kotlina.

1.1. Przedsmak Kotlina

Zacznijmy od prostego kodu pokazującego, czym jest Kotlin. Kod ten definiuje klasę `Person` (osoba), kolekcję klas, wyszukuje najstarszą osobę i wyświetla wynik. Nawet w tym niewielkim przykładzie odkryjesz wiele ciekawych cech języka. Opiszemy dokładniej kilka z nich, aby były zrozumiałe w dalszej części książki. Kod jest opisany bardzo ogólnie, nie przejmuj się więc, jeżeli nie wszystko będzie od razu zrozumiałe. W dalszej części rozdziału szczegółowo opiszemy wszystkie zawiłości.

Aby wypróbować poniższy przykład, najłatwiej uruchomić go w internetowym „placu zabaw" na stronie *http://try.kotl.in*. Wpisz na niej kod z listingu 1.1 i uruchom, klikając przycisk *Run*.

Listing 1.1. Przedsmak Kotlina

```
data class Person(val name: String,                    ← Klasa „danych".
                  val age: Int? = null)                ← Typ zerowalny (Int?) i domyślna wartość argumentu.
fun main(args: Array<String>) {                        ← Główna funkcja.
  val persons = listOf(Person("Alicja"),
                       Person("Bartek", age = 29))     ← Nazwany argument.
  val oldest = persons.maxBy { it.age ?: 0 }           ← Wyrażenie lambda i operator Elvisa.
  println("Najstarsza osoba: $oldest")                 ← Znakowy szablon.
}
// Najstarsza osoba: Person(name=Bartek, age=29)       ← Automatyczne użycie metody toString().
```

Zadeklarowałeś prostą klasę z dwiema właściwościami: `name` (imię) i `age` (wiek). Jeżeli wartość właściwości `age` nie zostanie określona, wtedy przyjmie ona domyślną wartość `null`. Tworząc listę klas, nie określiłeś wieku Alicji, więc została użyta domyślna wartość. Później użyłeś metody `maxBy()` do wyszukania najstarszej osoby. Wyrażenie lambda umieszczone w argumencie tej metody ma jeden parametr o domyślnej nazwie `it` (to). Operator Elvisa (`?:`) zwraca wartość 0, jeżeli właściwość `age` ma wartość `null`. Ponieważ nie określiłeś wieku Alicji, operator użył wartości 0, zatem za najstarszą osobę został uznany Bartek.

Podoba Ci się to? Czytaj więc dalej i zostań ekspertem od Kotlina. Mamy nadzieję, że wkrótce taki kod będziesz stosował w swoich projektach, nie tylko podczas lektury tej książki.

1.2. Najważniejsze cechy języka Kotlin

Prawdopodobnie masz już wyobrażenie, czym jest język Kotlin. Przyjrzyjmy się teraz dokładniej jego najważniejszym cechom. Najpierw sprawdźmy, jakiego rodzaju aplikacje można pisać w tym języku.

1.2.1. Docelowe platformy: serwery, Android i wszystko, gdzie jest Java

Kotlin ma być przede wszystkim bardziej zwięzłą, skuteczniejszą i bezpieczniejszą alternatywą dla Javy. Powinien dać się stosować wszędzie tam, gdzie jest Java, która jest niezwykle popularnym językiem, wykorzystywanym w szerokim spektrum środowisk, od inteligentnych kart (Java Card) do największych centrów danych działających w serwisach Google, Twitter, LinkedIn oraz w dużych firmach internetowych. W większości powyższych przypadków programiści mogą za pomocą języka Kotlin osiągać swoje cele, pisząc krótszy kod i unikając dzięki temu kłopotów.

Największe obszary zastosowań języka Kotlin to:

- aplikacje serwerowe (zazwyczaj WWW),
- aplikacje przenośne dla systemu Android.

Język Kotlin można stosować również w innych sytuacjach. Na przykład można uruchamiać kod w systemie macOS, wykorzystując silnik Multi-OS Engine (*https://software.intel.com/en-us/multi-os-engine*), a za pomocą platform TornadoFX (*https://github.com/edvin/tornadofx*) i JavaFX[1] można tworzyć aplikacje dla komputerów stacjonarnych.

Ponadto, w odróżnieniu od Javy, kod Kotlin można przekształcać w kod JavaScript i uruchamiać w przeglądarce. Jednak w chwili powstawania tej książki wciąż trwało w JetBrains rozpoznawanie obsługi języka JavaScript oraz tworzenie prototypu konwertera, dlatego temat ten nie jest opisany w tej książce. Rozważamy też przystosowanie przyszłych wersji języka do nowych platform.

Jak widzisz, obszar zastosowania Kotlina jest bardzo szeroki. Język ten nie skupia się na jednej domenie ani nie próbuje rozwiązać określonego problemu, z którym mierzą się dziś programiści. Wręcz przeciwnie, posiada wiele udoskonaleń usprawniających realizację zadań, które pojawiają się w procesie tworzenia oprogramowania. Jest doskonale zintegrowany z istniejącymi bibliotekami do różnych zastosowań i zagadnień programistycznych. Przyjrzyjmy się teraz kluczowym cechom wyróżniającym ten język programowania.

1.2.2. Statyczne typowanie danych

Kotlin, podobnie jak Java, jest językiem wykorzystującym **statyczne typowanie** danych. Oznacza to, że typ każdego wyrażenia jest znany na etapie kompilacji kodu, a kompilator sprawdza, czy istnieją metody i pola w obiektach, do których odwołuje się kod. Tym Kotlin różni się od języków z **dynamicznym typowaniem**, takich jak Groovy i JRuby w środowisku JVM. W tych językach można definiować zmienne oraz funkcje przechowujące i zwracające dane dowolnych typów, a także identyfikować metody i pola w trakcie działania kodu. Dzięki temu kod jest krótszy i bardziej elastyczny pod względem tworzenia struktur danych. Problem polega jednak na tym, że podczas kompilacji kodu nie są wykrywane pomyłki w nazwach identyfikatorów, co skutkuje błędnym niekiedy działaniem kodu.

W Kotlinie, inaczej niż w Javie, nie trzeba jawnie określać typu każdej zmiennej. W wielu przypadkach typ jest określany automatycznie na podstawie kontekstu, dzięki czemu można pomijać deklarację typu. Ilustruje to poniższy najprostszy przykład:

```
valx = 1
```

Deklarowana jest tu zmienna. Ponieważ jest ona inicjowana liczbą całkowitą, Kotlin automatycznie przyjmuje typ `Int`. Określanie przez kompilator typu danych na podstawie kontekstu nosi nazwę **domniemania typów**.

Statycznie typowany kod ma m.in. następujące zalety:

- *Wydajność* — metody wywoływane są szybciej, ponieważ w trakcie działania kodu nie trzeba wyszukiwać wywoływanej metody.
- *Niezawodność* — kompilator sprawdza poprawność kodu, dzięki czemu zmniejsza się prawdopodobieństwo jego awarii po uruchomieniu.

[1] Patrz: *JavaFX: Getting Started with JavaFX*, Oracle, *http://mng.bz/500y*.

- *Łatwość utrzymania* — praca z nieznanym kodem jest łatwiejsza, ponieważ wiadomo, jakiego rodzaju obiekty są w nim stosowane.
- *Dostępność narzędzi* — można korzystać z niezawodnej refaktoryzacji, precyzyjnego uzupełniania kodu oraz innych funkcjonalności środowiska IDE.

Dzięki domniemaniu typów zmniejsza się rozwlekłość kodu, charakterystyczna dla statycznego typowania, ponieważ nie trzeba jawnie deklarować wszystkich typów.

Jeżeli przyjrzysz się specyfikacji typów w języku Kotlin, znajdziesz w niej znane pojęcia. Klasy, interfejsy i typy generyczne są bardzo podobne do stosowanych w Javie. Zatem swoją wiedzę o tym języku możesz w dużej mierze przenieść do Kotlina.

Są też nowe idee. Najważniejsze z nich to **typy zerowalne**, dzięki którym można tworzyć stabilniejsze programy, w których już na etapie kompilacji można wykrywać wskaźniki o wartościach `null`. Do tych typów powrócimy w dalszej części rozdziału, a dokładniej opiszemy je w rozdziale 6.

Kolejną nowością w systemie typów są **typy funkcyjne**. Aby dowiedzieć się, czym one są i jak są obsługiwane w języku Kotlin, przypomnijmy sobie najważniejsze zasady programowania funkcyjnego.

1.2.3. Programowanie funkcyjne i obiektowe

Jako programista Java z pewnością znasz kluczowe pojęcia z dziedziny programowania obiektowego, natomiast programowanie funkcyjne może być dla Ciebie nowością. Poniżej wymienione są najważniejsze terminy stosowane w programowaniu funkcyjnym:

- *Funkcje pierwszej klasy* — funkcje (ciągi operacji) traktuje się jak wartości. Można je zapisywać w zmiennych, umieszczać w argumentach i zwracać w wynikach innych funkcji.
- *Niezmienność* — obiekty są niezmienne, tzn. po utworzeniu nie można zmieniać ich stanu.
- *Brak efektów ubocznych* — wykorzystywane są czyste funkcje zwracające te same wyniki dla tych samych wartości argumentów, nie można również zmieniać stanu innych obiektów ani komunikować się z kodem zewnętrznym.

Jakie korzyści płyną z programowania funkcyjnego? Po pierwsze, kod jest zwięzły. Jest krótszy i bardziej elegancki od kodu imperatywnego, ponieważ traktując funkcje jak wartości, lepiej wykorzystuje się siłę abstrakcji i unika duplikacji kodu.

Wyobraź sobie, że masz dwa podobne fragmenty kodu realizujące podobne zadania (na przykład wyszukiwanie określonych elementów kolekcji), jednak różniące się szczegółami (np. sposobem identyfikowania elementów). W takim wypadku wspólną logikę obu części można łatwo zapisać w postaci funkcji i umieścić ją w argumentach dwóch innych funkcji. Argumenty są funkcjami, ale można je opisać zwięzłą składnią w postaci funkcji anonimowych, tzw. **wyrażeń lambda**, jak niżej:

```
fun findAlicja() = findPerson { it.name == "Alicja" }
fun findBartek() = findPerson { it.name == "Bartek" }
```

Funkcja findPerson() zawiera wspólny algorytm wyszukiwania osoby.

Fragment w nawiasach klamrowych opisuje osobę, którą chcemy znaleźć.

Inną zaletą kodu funkcyjnego jest **bezpieczeństwo międzywątkowe**. Jednym z głównych źródeł błędów w programach wielowątkowych jest modyfikowanie tych samych danych w różnych wątkach bez odpowiedniej synchronizacji. Jeżeli natomiast stosuje się niezmienne struktury danych i czyste funkcje, można mieć pewność, że tego typu niebezpieczne modyfikacje nie będą się zdarzać, więc nie trzeba stosować skomplikowanych mechanizmów synchronizacyjnych.

Ponadto programowanie funkcyjne oznacza **prostsze testowanie** kodu. Funkcje niewywołujące efektów ubocznych można testować w izolacji bez konieczności tworzenia mnóstwa kodu konfiguracyjnego implementującego otoczenie, od którego testowany kod jest uzależniony.

Podsumowując, programowanie funkcyjne można stosować w każdym języku programowania, również w Javie, i w dużej mierze jest to dobry styl. Jednak nie wszystkie języki posiadają składnię i biblioteki umożliwiające łatwe stosowanie tego stylu. Na przykład nie było to możliwe w wersjach starszych niż Java 8. Kotlin oferuje bogaty zestaw funkcjonalności umożliwiających programowanie funkcyjne już na starcie. Są to między innymi:

- *Typy funkcyjne* umożliwiające umieszczanie funkcji w argumentach i wynikach innych funkcji.
- *Wyrażenia lambda* umożliwiające umieszczanie bloków kodu z minimalnym narzutem.
- *Klasy danych* umożliwiające tworzenie zwięzłych, niezmiennych obiektów.
- Standardowa biblioteka z bogatym zestawem *interfejsów API* do przetwarzania obiektów i kolekcji.

Kotlin umożliwia programowanie funkcyjne, ale go nie narzuca. Można w nim bez dodatkowego wysiłku tworzyć zmienne obiekty i funkcje z efektami ubocznymi, jeżeli jest taka potrzeba. Oczywiście korzystanie z platform opartych na hierarchii interfejsów i klas jest równie proste jak w Javie. Pisząc kod w języku Kotlin, można łączyć style programowania obiektowego i funkcyjnego oraz używać narzędzi najbardziej odpowiednich do rozwiązania danego problemu.

1.2.4. Bezpłatne i otwarte oprogramowanie

Środowisko Kotlin, obejmujące kompilator, biblioteki i narzędzia, jest w całości otwartym oprogramowaniem do bezpłatnego korzystania w dowolnych celach. Udostępniane jest w ramach licencji Apache 2, jego kod jest dostępny w serwisie GitHub (*http://github.com/jetbrains/kotlin*), a nowi członkowie społeczności są zawsze mile widziani. Do wyboru są trzy otwarte, w pełni obsługujące język Kotlin środowiska programistyczne: IntelliJ IDEA Community Edition, Android Studio i Eclipse (oczywiście IntelliJ IDEA Ultimate też).

Teraz, gdy wiesz już, jakiego rodzaju językiem jest Kotlin, poznaj jego zalety ujawniające się w wybranych zastosowaniach.

1.3. Zastosowania języka Kotlin

Jak wspomniałem wcześniej, dwa główne obszary zastosowań języka Kotlin to aplikacje serwerowe i aplikacje dla systemu Android. Przyjrzyjmy się teraz obu tym obszarom i sprawdźmy, dlaczego Kotlin tak dobrze się w nich sprawdza.

1.3.1. Kotlin na serwerach

Programowanie serwerów to bardzo szeroki temat, obejmujący między innymi następujące zagadnienia:

- aplikacje WWW generujące kody HTML stron,
- silniki aplikacji przenośnych udostępniające interfejs JSON API za pomocą protokołu HTTP,
- mikrousługi komunikujące się ze sobą za pomocą protokołu RPC.

Od wielu lat programiści tworzą tego rodzaju aplikacje w języku Java. W efekcie powstało mnóstwo platform i technologii ułatwiających programowanie. Zazwyczaj takie aplikacje nie są tworzone w odosobnieniu, od podstaw. Niemal zawsze istnieje jakiś system, który jest rozbudowywany, ulepszany lub wymieniany, a nowy kod integrowany jest z już istniejącymi elementami systemu, napisanego wiele lat wcześniej.

Pod tym względem ogromną zaletą Kotlina jest jego kompatybilność z istniejącym kodem Java. Niezależnie od tego, czy tworzy się nowy komponent, czy migruje istniejącą usługę, Kotlin nadaje się do tego celu znakomicie. Nie ma problemów z rozszerzaniem istniejących klas Java ani dodawaniem adnotacji metod i właściwości klas. Dodatkową korzyścią jest bardziej zwarty, stabilny i łatwiejszy w utrzymaniu kod.

Ponadto Kotlin oferuje kilka nowych technik ułatwiających programowanie. Na przykład za pomocą szablonów Builder można w zwięzły sposób tworzyć dowolne obiekty graficzne przy zachowaniu pełnej abstrakcyjności kodu i wykorzystaniu narzędzi do tworzenia jego powtarzalnych części.

Jednym z najprostszych przykładów wykorzystania tej cechy jest biblioteka generująca kod HTML, dzięki której kod zewnętrzny napisany w języku szablonowym można zastąpić zwięzłym kodem zachowującym zgodność typów, jak niżej:

```
fun renderPersonList(persons: Collection<Person>) =
  createHTML().table {
    for (person in persons) {
      tr {
        td { +person.name }
        td { +person.age }
        }
      }
    }
  }
```

Metody powiązane ze znacznikami HTML.

Zwykła pętla w języku Kotlin.

Można w ten sposób łatwo łączyć funkcje reprezentujące znaczniki HTML z typowymi konstrukcjami języka Kotlin. Nie trzeba stosować osobnego języka szablonowego z własną składnią. Wystarczy zastosować pętlę generującą kod HTML strony.

Innym obszarem, w którym stosuje się czyste, zwięzłe języki **DSL** (ang. *domain-specific language*, język dziedzinowy), są platformy obsługujące bazy danych. Na przykład platforma Exposed (*https://github.com/jetbrains/exposed*) oferuje czytelny język DSL umożliwiający tworzenie i wywoływanie zapytań SQL z pełnym zachowaniem kontroli typów danych. Poniżej przedstawiony jest przykład prostego kodu wykorzystującego tę funkcjonalność:

```
object CountryTable : IdTable() {
  val name = varchar("name", 250).uniqueIndex()
  val iso = varchar("iso", 2).uniqueIndex()
}
class Country(id: EntityID) : Entity(id) {
  var name: String by CountryTable.name
  var iso: String by CountryTable.iso
}
val poland = Country.find {
  CountryTable.iso.eq("pl")
}.first()
println(poland.name)
```

Opis tabeli w bazie danych.

Utworzenie klasy odpowiadającej obiektowi w bazie danych.

Bazę można odpytywać bezpośrednio za pomocą kodu Kotlin.

Powyższe techniki opiszemy bardziej szczegółowo w podrozdziale 7.5 oraz w rozdziale 11.

1.3.2. Kotlin w Androidzie

Typowa aplikacja dla urządzenia przenośnego bardzo różni się od aplikacji dla komputera. Jest znacznie mniejsza i w ograniczonym stopniu wykorzystuje istniejący kod. Ponadto zazwyczaj aplikacje przenośne trzeba tworzyć szybko i muszą one działać niezawodnie na urządzeniach bardzo różnych rodzajów. Do tworzenia tego typu projektów Kotlin również nadaje się bardzo dobrze.

Struktura języka Kotlin oraz dostępność specjalnej wtyczki dla kompilatora powodują, że programowanie w systemie Android jest efektywniejsze i przyjemniejsze. Najczęstsze zadania, na przykład kodowanie obsługi kontrolek lub wiązanie elementów interfejsu użytkownika z właściwościami klas, realizuje się za pomocą znacznie krótszego kodu, a czasami bez kodu w ogóle (generuje go kompilator). Biblioteka Anko (*https://github.com/kotlin/anko*), również opracowana przez zespół autorów Kotlina, jeszcze bardziej ułatwia to zadanie, oferując adaptery przystosowane do wielu standardowych interfejsów API systemu Android.

Poniżej znajduje się prosty kod wykorzystujący bibliotekę Anko i pokazujący, co powinien zrobić programista, aby w języku Kotlin utworzyć aplikację dla systemu Android. Musi ten kod umieścić w aktywności i prosta aplikacja gotowa!

```
verticalLayout {
  val name = editText()
  button("Powiedz: cześć!") {
    onClick { toast("Cześć, ${name.text}!") }
  }
}
```

Utworzenie prostego pola tekstowego.

Zwarta składnia wiążąca kod nasłuchujący, który wyświetla komunikat.

Po kliknięciu tego przycisku wyświetlane jest imię wpisane w polu tekstowym.

Inną ogromną zaletą języka Kotlin jest większa stabilność tworzonych w nim aplikacji. Jeżeli masz doświadczenie w pisaniu aplikacji dla systemu Android, na pewno dobrze znasz komunikat: „Niestety, aplikacja została zatrzymana". Komunikat ten pojawia się po zgłoszeniu nieobsługiwanego wyjątku, zazwyczaj `NullPointerException`. System typów w języku Kotlin, które umożliwiają precyzyjne śledzenie wartości `null`, łagodzi problem zerowych wskaźników. W większości przypadków kod, który w języku Java powoduje zgłoszenie wyjątku `NullPointerException`, w języku Kotlin w ogóle się nie kompiluje, dzięki czemu przed udostępnieniem aplikacji użytkownikom można błąd poprawić.

Ponieważ Kotlin jest w pełni kompatybilny z Java 6, nie występują problemy z integracją. Można korzystać ze wszystkich nowych, ciekawych funkcjonalności Kotlina, a użytkownicy będą mogli uruchamiać aplikacje na swoich urządzeniach wyposażonych nawet w starsze wersje systemu Android.

Pod względem wydajności Kotlin również wypada niezgorzej. Wygenerowany przez kompilator kod działa równie wydajnie jak kod Java. Biblioteka uruchomieniowa Kotlina jest bardzo mała, więc skompilowane pakiety aplikacyjne nie są duże. Ponadto wiele standardowych bibliotek Kotlina zamienia funkcje lambda na kod procesora. Dzięki temu nie są tworzone nowe obiekty, a działanie aplikacji nie jest wstrzymywane na czas porządkowania pamięci.

Teraz, gdy wiesz, jaka jest przewaga Kotlina nad Javą, poznaj jego filozofię, odróżniającą go od innych nowoczesnych języków dla środowiska JVM.

1.4. Filozofia języka Kotlin

O Kotlinie zwykliśmy mówić, że jest to pragmatyczny, zwięzły, bezpieczny, kompatybilny język programowania. Co konkretnie znaczą te słowa?

1.4.1. Pragmatyzm

Pragmatyzm oznacza prostą cechę: Kotlin jest językiem przeznaczonym do rozwiązywania praktycznych problemów. Jego kształt wynika z wieloletniego doświadczenia autorów w tworzeniu dużych systemów, a oferowane funkcjonalności zostały opracowane pod kątem typowych przypadków, z którymi mierzą się programiści. Co więcej, zarówno zespół JetBrains, jak i społeczność użytkowników od wielu lat korzystają ze wstępnych wersji Kotlina i ich opinie zostały uwzględnione w oficjalnej wersji. Dlatego twierdzimy, że Kotlin umożliwia rozwiązywanie problemów w rzeczywistych projektach.

Kotlin nie jest językiem akademickim. Nie staramy się wzbogacać sztuki programowania i implementować innowacji informatycznych. Zamiast tego tam, gdzie jest to możliwe, wykorzystujemy funkcjonalności i rozwiązania spotykane w innych językach, które sprawdziły się w praktyce. Dzięki temu nasz język jest mniej skomplikowany, prostszy w opanowaniu i wykorzystujący sprawdzone koncepcje.

Co więcej, Kotlin nie narzuca żadnego stylu ani zasad programowania. Rozpoczynając naukę tego języka, możesz stosować style i techniki, które znasz z Javy. W miarę postępów będziesz stopniowo odkrywał przydatne cechy Kotlina i uczył się wykorzystywać je w swoim kodzie, dzięki czemu będzie on bardziej zwięzły i funkcjonalny.

Innym pragmatycznym aspektem Kotlina jest koncentracja na narzędziach. Dobre środowisko programistyczne jest równie ważne dla skutecznego programowania jak dobry język. Dlatego marginalne traktowanie środowiska IDE jest nie do przyjęcia. W przypadku języka Kotlin wtyczka dla środowiska IntelliJ IDEA była rozwijana równolegle z kompilatorem, a funkcjonalności języka są projektowane również pod kątem narzędzi.

Środowisko programistyczne odgrywa wielką rolę w poznawaniu funkcjonalności Kotlina. Często narzędzia automatycznie wykrywają najczęściej stosowane schematy kodowania, które można zastępować bardziej zwartymi konstrukcjami, i oferują stosowne zmiany. Analizując składnię sugerowaną przez zautomatyzowane narzędzia, możesz uczyć się wykorzystywać nowe funkcjonalności w swoim kodzie.

1.4.2. Zwięzłość

Powszechnie wiadomo, że poznanie istniejącego kodu zajmuje programiście więcej czasu niż napisanie nowego. Wyobraź sobie, że jesteś członkiem zespołu pracującego nad ogromnym projektem i musisz zakodować nową funkcjonalność lub poprawić błąd. Jaki jest Twój pierwszy krok? Musisz najpierw znaleźć sekcję kodu, którą masz zmienić. Dopiero wtedy możesz poprawić błąd. Aby określić, co powinieneś zrobić, musisz przeanalizować mnóstwo kodu, który mógł zostać napisany stosunkowo niedawno przez Twoich kolegów, ale równie dobrze dużo wcześniej przez Ciebie samego lub kogoś, kto już nie pracuje nad projektem. Dopiero po zrozumieniu sąsiedniego kodu jesteś w stanie wprowadzić niezbędne zmiany.

Im prostszy i krótszy jest kod, tym szybciej można się dowiedzieć, co się w nim dzieje. Oczywiście przemyślana struktura i opisowe nazwy mają tu istotne znaczenie. Ale ważny jest też sam język i jego *zwięzłość*. Język jest zwięzły, jeżeli jego składnia jasno oddaje przeznaczenie kodu i nie zaciemniają go dodatkowe instrukcje niezbędne do osiągnięcia zamierzonego celu.

Dokładamy wszelkich starań, aby cały kod Kotlin niósł treść, a nie tylko spełniał wymogi struktur programistycznych. Niektóre standardowe konstrukcje z języka Java, takie jak gettery, settery lub parametry konstruktorów powiązane z właściwościami, są w języku Kotlin stosowane niejawnie i nie zaciemniają kodu.

Niekiedy przyczyną niepotrzebnego wydłużenia kodu jest konieczność jawnego implementowania często wykonywanych zadań, na przykład wyszukiwania elementu w kolekcji. Podobnie jak wiele innych nowoczesnych języków, Kotlin również zawiera bogatą standardową bibliotekę umożliwiającą zastępowanie długich, powtarzalnych sekcji kodu gotowymi metodami z biblioteki. Dzięki funkcjom lambda można w funkcjach biblioteki łatwo umieszczać małe bloki kodu. W ten sposób wszystkie często wykonywane zadania można zlecać bibliotece i pisać tylko unikatowe, charakterystyczne dla danego zadania sekcje kodu.

Jednakże przeznaczeniem języka Kotlin nie jest ściśnięcie kodu do minimalnej liczby znaków. Na przykład mimo tego, że możliwe jest przeciążanie operatorów, nie można definiować własnych. Dlatego twórcy bibliotek nie mogą zastępować nazw metod tajemniczymi symbolami. Słowa są zazwyczaj bardziej zrozumiałe niż symbole i łatwiej jest znaleźć ich opis w dokumentacji.

Napisanie zwartego kodu zajmuje mniej czasu i, co ważne, mniej czasu potrzeba na jego przeczytanie. Dzięki temu programista może pracować wydajniej i kodować szybciej.

1.4.3. Bezpieczeństwo

Mówiąc, że język programowania jest *bezpieczny*, mamy zazwyczaj na myśli jego strukturę chroniącą programistę przed popełnianiem określonego rodzaju błędów. Oczywiście nie jest to bezwzględnym wyznacznikiem jakości języka. Żaden język nie chroni całkowicie przed błędami, a ponadto zazwyczaj taka ochrona kosztuje, ponieważ kompilatorowi trzeba dostarczyć dodatkowych informacji o operacjach, które ma wykonywać program. Kompilator porównuje te informacje z wykonywanymi operacjami. Z tego powodu zawsze istnieje pewien kompromis pomiędzy zapewnianym poziomem bezpieczeństwa a nakładem pracy, jakiego wymaga umieszczenie w kodzie szczegółowych adnotacji.

Naszym zamiarem było zapewnienie w języku Kotlin większego bezpieczeństwa niż w Javie, ale mniejszym kosztem. Sama maszyna JVM daje gwarancję bezpieczeństwa, na przykład zapewnia ochronę pamięci, zabezpiecza przed przepełnieniem buforów, jak również chroni przed różnymi problemami wynikającymi z niewłaściwego korzystania z dynamicznie przydzielanej pamięci. Kotlin jako język ze statycznym typowaniem przeznaczonym dla maszyny JVM zapewnia dodatkowo bezpieczeństwo typów danych. Odbywa się to jednak mniejszym kosztem niż w Javie, ponieważ kompilator automatycznie domniemywa typy i nie trzeba ich wszędzie specyfikować.

Kotlin oferuje więcej funkcjonalności umożliwiających wykrywanie błędów jeszcze na etapie kompilacji kodu, a nie dopiero po jego uruchomieniu. Przede wszystkim minimalizuje ryzyko zgłoszenia wyjątku `NullPointerException`. System typów umożliwia śledzenie zmiennych, które mogą przyjmować wartość `null`, oraz tych, które nie mogą jej przyjmować, i uniemożliwia wykonywanie na nich operacji prowadzących do zgłoszenia powyższego wyjątku. Związany z tym dodatkowy koszt jest minimalny, ponieważ w celu oznaczenia zerowalnej zmiennej wystarczy wpisać znak zapytania, jak niżej:

```
val s: String? = null  <-- Zmienna może przyjmować wartość null.
val s2: String = ""    <-- Zmienna nie może przyjmować wartości null.
```

Ponadto Kotlin oferuje wiele wygodnych sposobów przetwarzania zerowalnych danych, co znacznie zmniejsza ryzyko awarii aplikacji.

Innym wyjątkiem, przed którym chroni Kotlin, jest `ClassCastException`. Wyjątek ten jest zgłaszany podczas rzutowania typów bez uprzedniego sprawdzenia, czy jest to poprawna operacja. W Javie często programiści rezygnują ze sprawdzania typu, ponieważ jego nazwy trzeba użyć dwukrotnie: najpierw w instrukcji sprawdzającej, a następnie w instrukcji rzutującej typ. Natomiast w Kotlinie sprawdzanie i rzutowanie typu realizuje się za pomocą jednej instrukcji. Po sprawdzeniu typu można odwoływać się do elementów członkowskich bez dodatkowego rzutowania. Zatem nie ma powodu, aby rezygnować ze sprawdzania typu, nie ma również możliwości popełnienia błędu. Poniżej przedstawiony jest przykład:

```
if (value is String)        ← Sprawdzenie typu.
  println(value.toUpperCase()) ← Użycie metody wybranego typu.
```

1.4.4. Kompatybilność

Twoje pierwsze pytanie dotyczące *kompatybilności* zapewne brzmi: „Czy mogę używać istniejących bibliotek?". W języku Kotlin odpowiedź brzmi: „Tak, oczywiście". Niezależnie od interfejsu API, który udostępnia biblioteka, można z niej korzystać. Można wywoływać metody Java, rozszerzać klasy, implementować interfejsy, stosować adnotacje itp.

Kotlin, w odróżnieniu od innych języków dla maszyny JVM, oferuje jeszcze większą kompatybilność umożliwiającą łatwe wywoływanie jego kodu w kodzie Java. Nie są do tego potrzebne żadne sztuczki: klasy i metody napisane w języku Kotlin wywołuje się tak samo jak klasy i metody napisane w Javie. Daje to ogromną elastyczność w mieszaniu kodów Kotlin i Java. Jeżeli trzeba użyć języka Kotlin w projekcie napisanym w języku Java, można za pomocą konwertera Java-Kotlin przekształcić dowolną wybraną klasę, a pozostałą część kodu skompilować i uruchomić bez dodatkowych modyfikacji. Konwersja dotyczy dowolnego typu klasy.

Innym obszarem kompatybilności, na którym skupia się Kotlin, jest jak najpełniejsze wykorzystanie istniejących bibliotek Java. Kotlin nie ma własnej kolekcji bibliotek, ponieważ w całości opiera się na standardowych bibliotekach Java i rozszerza je o dodatkowe metody, bardziej wygodne w użyciu (dokładniej ten mechanizm opisany jest w podrozdziale 3.3). Oznacza to, że nie trzeba opakowywać ani przekształcać klas z języka Java na Kotlin lub odwrotnie. Z bogactwa interfejsu API języka Kotlin można korzystać bez żadnych dodatkowych kosztów.

Narzędzia języka Kotlin również można w pełni wykorzystywać w projektach obejmujących dwa języki. Można kompilować dowolne kombinacje plików źródłowych Java i Kotlin, niezależnie od ich wzajemnych zależności. Środowisko programistyczne również obsługuje oba języki i umożliwia m.in.:

- swobodne przemieszczanie się pomiędzy plikami źródłowymi Java i Kotlin,
- diagnozowanie mieszanych projektów i przełączanie się pomiędzy kodami napisanymi w obu językach,
- refaktorowanie metod Java oraz odpowiednią modyfikację kodu Kotlin i odwrotnie.

Mamy nadzieję, że przekonaliśmy Cię, abyś wypróbował język Kotlin. Jak więc zacząć go używać? W następnej części rozdziału opisujemy proces kompilowania i uruchamiania kodu Kotlin za pomocą wiersza poleceń oraz innych narzędzi.

1.5. Narzędzia języka Kotlin

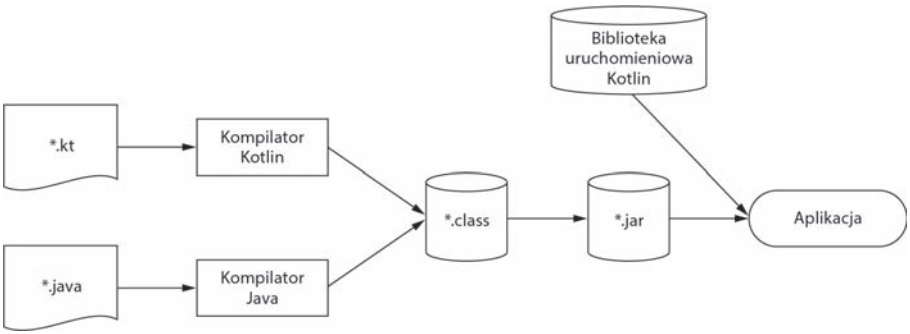
Kotlin, tak jak Java, jest językiem kompilowanym. Oznacza to, że aby uruchomić kod, trzeba go najpierw skompilować. Przeanalizujmy teraz proces kompilacji i przyjrzyjmy się różnym narzędziom, które zajmują się tym procesem za Ciebie. Jeżeli potrzebujesz dodatkowych informacji o przygotowaniu środowiska, zapoznaj się z sekcją *Tutorials* (podręczniki) na stronie Kotlina (*https://kotlinlang.org/docs/tutorials*).

1.5.1. Kompilator kodu

Kod źródłowy Kotlin zazwyczaj zapisuje się w plikach z rozszerzeniem *.kt*. Kompilator analizuje kod źródłowy i generuje pliki *.class* tak samo jak kompilator Java. Następnie wygenerowane pliki są pakowane i uruchamiane w sposób właściwy rodzajowi aplikacji. W najprostszym przypadku kod kompiluje się za pomocą polecenia `kotlinc`, a uruchamia poleceniem `java`, jak niżej:

```
kotlinc <plik źródłowy lub katalog> -include-runtime -d <nazwa pliku JAR>
java -jar <nazwa pliku JAR>
```

Rysunek 1.1 przedstawia uproszczony proces budowania aplikacji w języku Kotlin.



Rysunek 1.1. Proces budowania aplikacji w języku Kotlin

Kod wygenerowany za pomocą kompilatora Kotlin jest uzależniony od **biblioteki uruchomieniowej** tego języka zawierającej własne klasy i rozszerzenia, o które uzupełniane są standardowe interfejsy API języka Java. Biblioteka uruchomieniowa musi być dostarczana razem z aplikacją.

W większości przypadków będziesz kompilował kod za pomocą systemów Maven, Gradle i Ant. Kotlin jest kompatybilny ze wszystkimi wymienionymi systemami. Szczegółowe informacje na ten temat znajdziesz w dodatku A. Systemy te obsługują również projekty z mieszanym kodem. Ponadto Maven i Gradle dołączają bibliotekę uruchomieniową Kotlina jako zależność wymaganą przez aplikację.

1.5.2. Wtyczki dla IntelliJ IDAE i Android Studio

Wtyczka Kotlin dla środowiska IntelliJ IDEA była rozwijana równolegle z samym językiem. IntelliJ IDEA jest najlepiej wyposażonym środowiskiem programistycznym dla języka Kotlin. Jest ono dojrzałe, stabilne i oferuje pełny zestaw narzędzi programistycznych.

Wtyczka jest standardowo dostępna w wersjach IntelliJ IDEA 15 i nowszych, nie trzeba więc wykonywać dodatkowych czynności, aby z niej korzystać. Można jej używać w wersjach IntelliJ IDEA Community Edition i IntelliJ IDEA Ultimate. Wystarczy w oknie *New Project* (nowy projekt) wybrać opcję *Kotlin* i gotowe.

Jeżeli korzystasz z Android Studio, musisz zainstalować wtyczkę za pomocą menedżera wtyczek. W tym celu w oknie *Settings* (ustawienia) wybierz sekcję *Plugins* (wtyczki), kliknij przycisk *Install JetBrains plugin* (zainstaluj wtyczkę JetBrains) i wybierz z listy pozycję *Kotlin*.

1.5.3. Interaktywna powłoka

Aby szybko sprawdzić krótki kod, możesz użyć interaktywnej powłoki (tzw. REPL — ang. *read-eval-print loop*, pętla „wczytaj, wykonaj, wyświetl"). W tej powłoce można wpisywać kod Kotlin wiersz po wierszu i od razu sprawdzać wyniki jego wykonania. Powłokę uruchamia się, wpisując polecenie `kotlinc` bez argumentów lub klikając odpowiednią opcję menu w środowisku IntelliJ IDEA.

1.5.4. Wtyczka dla Eclipse

Jeżeli używasz środowiska Eclipse, również możesz korzystać z wtyczki dla języka Kotlin. Wtyczka ta oferuje najważniejsze funkcjonalności, takie jak poruszanie się po projekcie i uzupełnianie kodu. Jest dostępna w serwisie Eclipse Marketplace. Aby ją zainstalować, wybierz polecenie menu *Help/Eclipse Marketplace*, a następnie wyszukaj na liście pozycję *Kotlin*.

1.5.5. Internetowy „plac zabaw"

Najprostszy sposób wypróbowania języka Kotlin nie wymaga instalacji ani konfiguracji środowiska. Na stronie *http://try.kotl.in* dostępny jest „plac zabaw", na którym można pisać, kompilować i uruchamiać proste programy. Strona zawiera również przykładowe kody prezentujące możliwości tego języka, między innymi wszystkie przykłady opisane w tej książce. Dostępne są też serie interaktywnych ćwiczeń do nauki języka.

1.5.6. Konwerter Java-Kotlin

Opanowanie nowego języka programowania nigdy nie przychodzi bez wysiłku. Dlatego przygotowaliśmy ciekawe, małe narzędzie, dzięki któremu, wykorzystując swoje doświadczenie z językiem Java, szybciej nauczysz się Kotlina. Tym narzędziem jest zautomatyzowany konwerter Java-Kotlin.

Na początku nauki, gdy nie będziesz jeszcze znał dokładnie składni języka, konwerter pomoże Ci formułować wyrażenia. Będziesz mógł pisać kod w Javie, zapisywać go w pliku i automatycznie konwertować na język Kotlin. Uzyskany kod nie zawsze będzie idealnie ścisły, ale będzie działał poprawnie i pozwoli Ci robić postępy w nauce.

Konwerter doskonale nadaje się również do wprowadzenia języka Kotlin do istniejącego projektu. Jeżeli będziesz chciał utworzyć nową klasę, najlepiej zrób to od razu w Kotlinie. W przypadku konieczności wprowadzenia większych zmian w istniejącej klasie również możesz użyć Kotlina. W tym miejscu pojawia się konwerter. Najpierw przekształć klasę na język Kotlin, a następnie wprowadź w niej zmiany, wykorzystując wszystkie zalety tego języka.

Używanie konwertera w środowisku IntelliJ IDEA jest wyjątkowo proste. Wystarczy skopiować kod Java i wkleić go do pliku Kotlin albo użyć polecenia menu *Convert Java File to Kotlin File* (przekształć plik Java w Kotlin), jeżeli trzeba przekształcić cały plik. Konwerter jest również dostępny w środowisku Eclipse i w internecie.

1.6. Podsumowanie

- Język Kotlin wykorzystuje statyczne typy danych i domniemanie typów, dzięki czemu kod jest poprawny, wydajny i zwięzły.
- Jest to zarówno język programowania obiektowego, jak i funkcyjnego o wysokim poziomie abstrakcji dzięki funkcjom pierwszej klasy. Obsługa niezmiennych wartości ułatwia tworzenie i testowanie kodu wielowątkowego.
- Język ten dobrze nadaje się do tworzenia aplikacji serwerowych, w pełni obsługuje wszystkie istniejące platformy Java i oferuje nowe narzędzia do wykonywania najczęstszych zadań, takich jak generowanie kodu HTML i obsługa baz danych.
- Można go używać do tworzenia aplikacji dla systemu Android, ponieważ jego biblioteka uruchomieniowa jest mała, kompilator obsługuje interfejs API systemu Android, a biblioteka funkcji realizujących najczęstsze operacje w tym systemie jest bardzo bogata.
- Kotlin jest bezpłatnym, otwartym oprogramowaniem, w pełni obsługiwanym przez najważniejsze środowiska programistyczne i systemy kompilujące.
- Jest to język pragmatyczny, bezpieczny, zwięzły i kompatybilny. Wykorzystuje sprawdzone sposoby realizacji typowych zadań i zapobiega powstawaniu najczęstszych problemów, na przykład zgłaszaniu wyjątku `NullPointerException`. Kod jest zwarty i czytelny, w pełni integrujący się z kodem Java.

Skorowidz

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

W

Z


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

Kotlin został zaprojektowany jako obiektowy język w pełni interoperacyjny z kodem napisanym w Javie. Został zaprezentowany w 2011 roku. Od tego czasu jest rozwijany, a jego popularność stale rośnie. Kotlin jest językiem o statycznie typowanych zmiennych. Charakteryzuje się czytelną składnią oraz ścisłą integracją z istniejącymi projektami, bibliotekami i platformami utworzonymi w Javie. Kompiluje się do postaci binarnego kodu JVM, więc można go uruchamiać wszędzie tam, gdzie jest Java, również w Androidzie. Kotlin zainteresuje programistów aplikacji mobilnych także dlatego, że kod napisany w tym języku obciąża system w minimalnym stopniu. Przy tym wszystkim środowisko Kotlina jest w całości otwartym oprogramowaniem do bezpłatnego korzystania w dowolnych celach!

Ta książka jest przeznaczona dla osób, które mają pewne doświadczenia z Javą i chcą szybko poznać Kotlina w stopniu pozwalającym na tworzenie aplikacji serwerowych, dla systemu Android i maszyn JVM. Znalazł się tu opis podstawowych cech języka i jego najważniejszych struktur, a następnie przedstawiono bardziej zaawansowane zagadnienia, takie jak tworzenie wysokopoziomowych abstrakcji i języków domenowych. Duży nacisk położono na integrację kodu Kotlin z istniejącymi projektami Java oraz na sposoby wprowadzania Kotlina do aktualnie użytkowanego środowiska. Zaprezentowano kilka przydatnych bibliotek i narzędzi, znakomicie ułatwiających pracę programiście. Nie zabrakło również licznych przykładów kodu, ilustrujących omawiane zagadnienia.

W tej książce między innymi:

- solidne wprowadzenie do Kotlina
- klasy, klasy danych i funkcje lambda
- typy danych, w tym kolekcje i puste zmienne
- własne interfejsy i abstrakcje w Kotlinie
- parametry reifikowanego typu, adnotacje i refleksje

Dmitry Jemerov pracuje w JetBrains (firma programistyczna z siedzibą w Pradze) od 2003 roku. Jest jednym ze współautorów Kotlina, dla którego stworzył pierwszą wersję generatora kodu na maszyny JVM. Prezentował ten język na konferencjach na całym świecie. Obecnie kieruje zespołem rozwijającym wtyczkę Kotlin dla środowiska IntelliJ IDEA.

Svetlana Isakova dołączyła do zespołu Kotlina w 2011 roku. Pracowała nad domniemaniem typów i rozpoznawaniem przeciążania w kompilatorze. Obecnie zajmuje się popularyzacją języka Kotlin, prezentując go na konferencjach i szkoleniach.

Kotlin — idealny dla Androida!

Helion

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶

ISBN 978-83-283-4720-5

9 788328 347205

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 67,00 zł