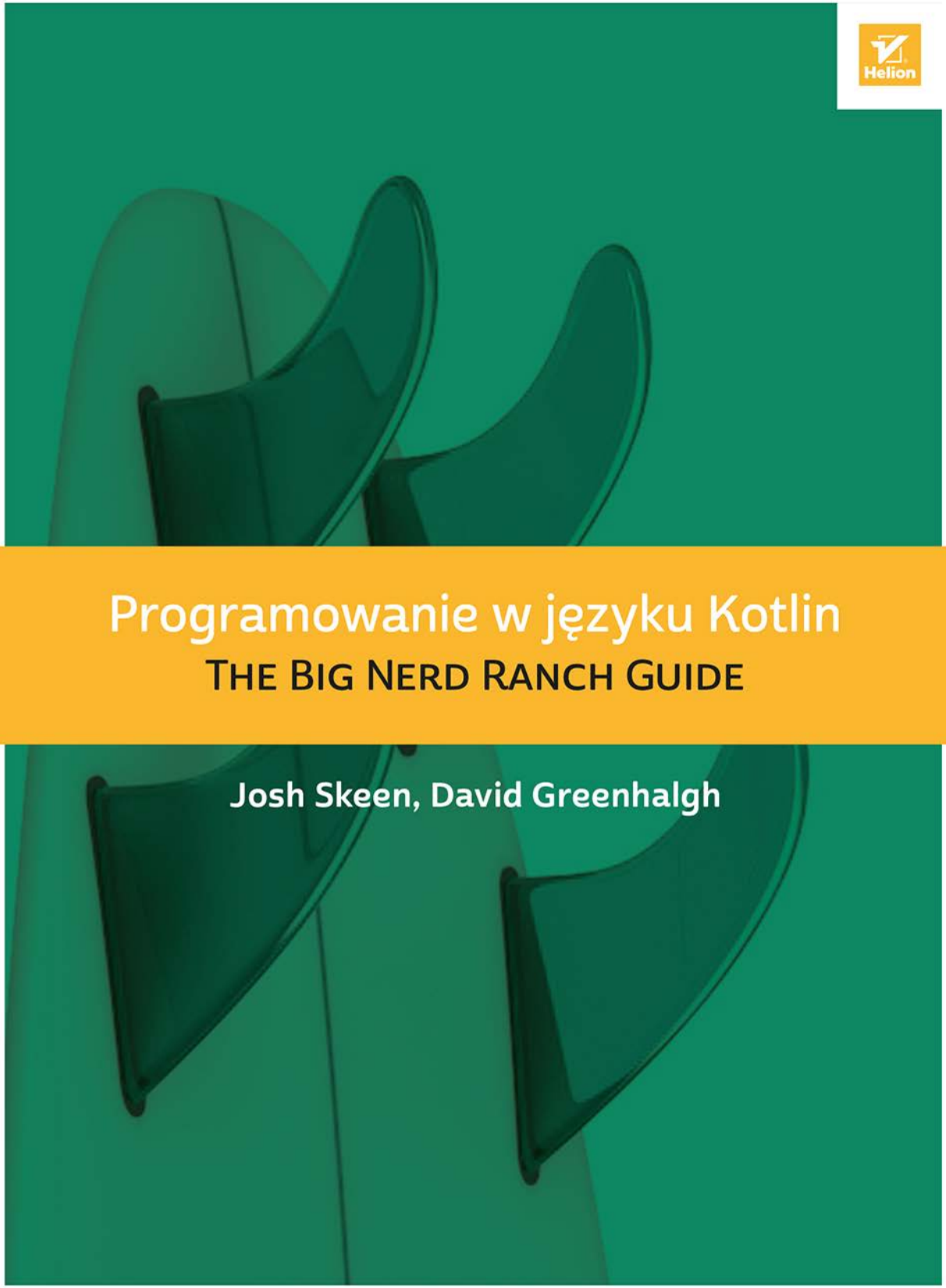


Programowanie w języku Kotlin

THE BIG NERD RANCH GUIDE

Josh Skeen, David Greenhalgh

Tytuł oryginału: Kotlin Programming: The Big Nerd Ranch Guide

Tłumaczenie: Piotr Rajca

ISBN: 978-83-283-5536-1

Authorized translation from the English language edition, entitled KOTLIN PROGRAMMING: THE BIG NERD RANCH GUIDE, 1st Edition by SKEEN, JOSH; GREENHALGH, DAVID; published by Pearson Education, Inc, publishing as The Big Nerd Ranch Guides.

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)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/prokot.zip

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

Printed in Poland.

Spis treści

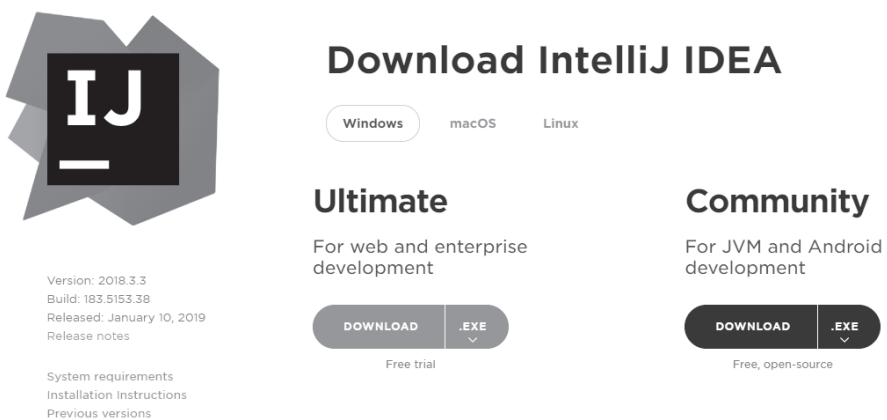
1

Pierwsza aplikacja w Kotlinie

W tym rozdziale Czytelnik napisze swoją pierwszą aplikację w języku Kotlin, używając przy tym IntelliJ IDEA. Przejście tej programistycznej inicjacji pozwoli poznać to środowisko programistyczne, utworzyć nowy projekt programu w języku Kotlin, napisać pierwszy fragment kodu w tym języku oraz przyjrzeć się generowanym przez niego wynikom. Projekt utworzony w tym rozdziale posłuży jako poligon doświadczalny do sprawdzania nowych pojęć prezentowanych w dalszej części książki.

Instalowanie IntelliJ IDEA

IntelliJ IDEA jest zintegrowanym środowiskiem programistycznym (tak zwanym *IDE* od ang.: *Integrated Development Environment*) przeznaczonym do pisania programów w języku Kotlin stworzonym przez firmę JetBrains (która jest także twórcą samego języka Kotlin). W pierwszej kolejności należy zacząć od pobrania IntelliJ IDEA Community Edition z witryny JetBrains: *https://www.jetbrains.com/idea/download* (patrz rysunek 1.1).



Rysunek 1.1. Pobieranie programu instalacyjnego IntelliJ IDEA Community Edition

Po pobraniu programu należy postępować zgodnie z instrukcjami dotyczącymi używanej platformy systemowej podanymi na stronie poświęconej instalacji i konfiguracji IDE — *https://www.jetbrains.com/help/idea/install-and-set-up-product.html*.

Środowisko IntelliJ IDEA, które dalej będziemy nazywali IntelliJ, ułatwia pisania odpowiednio sformatowanego kodu w języku Kotlin. Upraszcza także cały proces programistyczny, gdyż udostępnia wbudowane narzędzia do uruchamiania pisanych aplikacji, debugowania ich, sprawdzania oraz refaktoryzacji kodu. Więcej informacji na temat tego, dlaczego do pisania kodu w Kotlinie polecamy właśnie to zintegrowane środowisko programistyczne, można znaleźć pod koniec tego rozdziału w podrozdziale pt. „Dla ciekawskich: Dlaczego warto używać właśnie IntelliJ?”.

Pierwszy projekt programu w Kotlinie

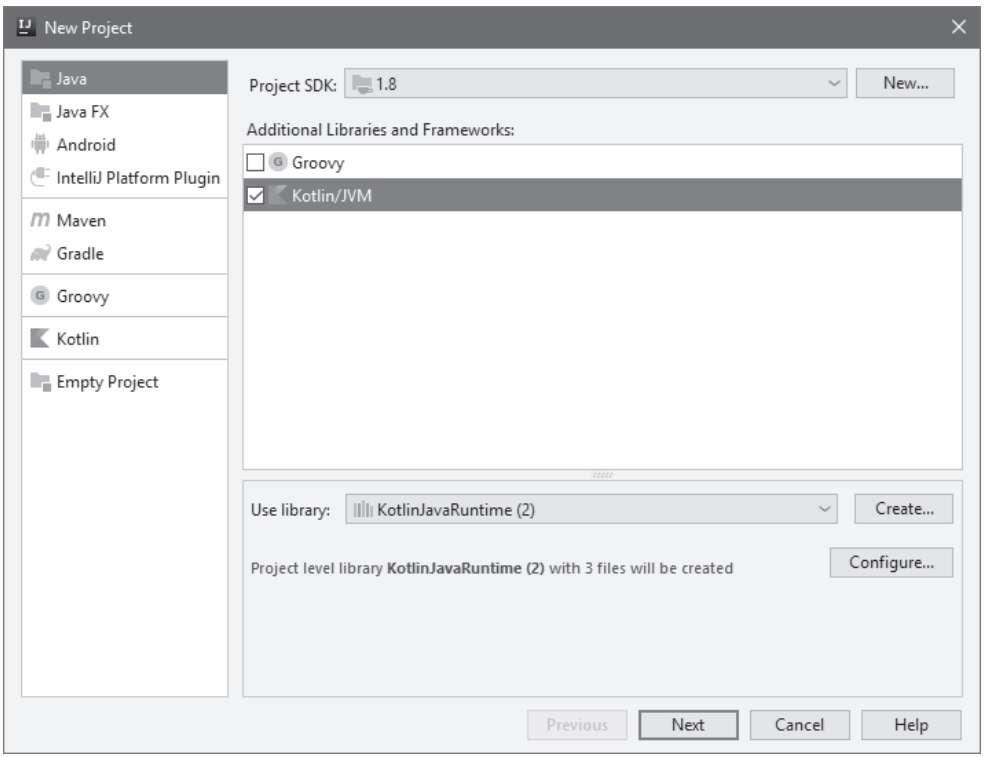
Gratulujemy! A zatem Czytelnik dysponuje już językiem Kotlin oraz potężnym zintegrowanym środowiskiem programistycznym do pisania programów w tym języku. Teraz pozostaje już tylko jedno: nauczyć się płynnie posługiwać tym językiem. Pierwszym zadaniem na tej drodze będzie utworzenie nowego projektu programu pisanego w Kotlinie.

Przede wszystkim należy uruchomić IntelliJ; w efekcie na ekranie zostanie wyświetlone okno dialogowe przedstawione na rysunku 1.2.

Rysunek 1.2. Powitalne okno dialogowe IDE IntelliJ

(Jeśli IDE zostanie uruchomione po raz kolejny, a nie pierwszy raz, to najprawdopodobniej zostanie w nim od razu wyświetlony projekt, nad którym pracowaliśmy wcześniej. Aby ponownie wyświetlić to powitalne okno dialogowe, wystarczy wybrać z menu opcję *File/Close Project*).

Kliknięcie przycisku *Create New Project* spowoduje wyświetlenie okna dialogowego *New Project* przedstawionego na rysunku 1.3.

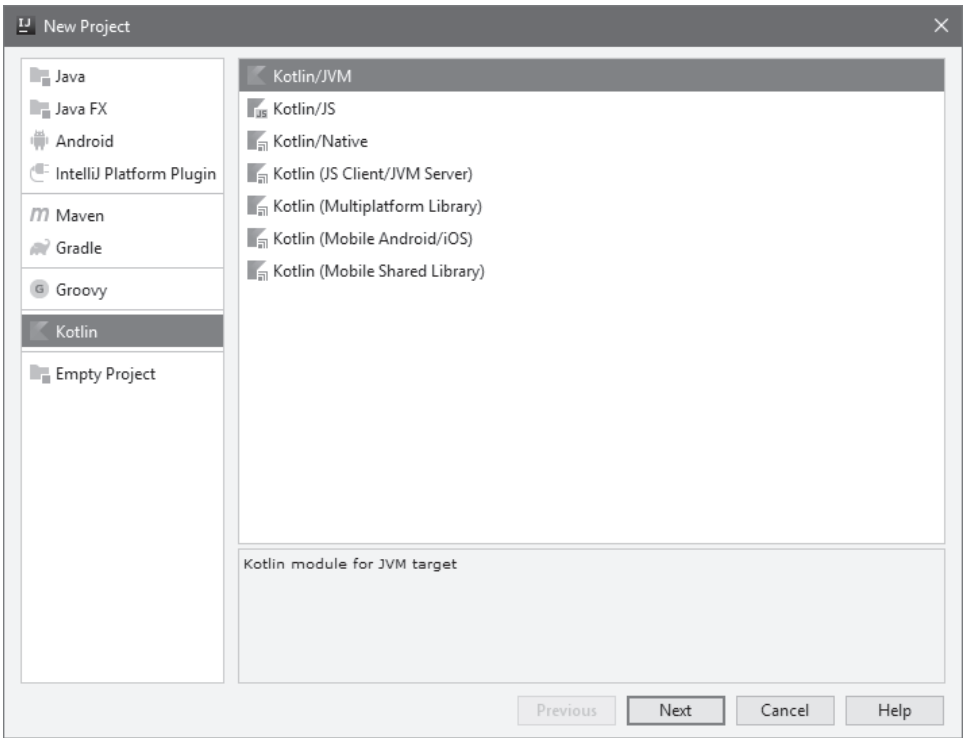


Rysunek 1.3. Okno dialogowe New Project

W tym oknie w jego lewej kolumnie należy zaznaczyć opcję *Kotlin*, a w obszarze po prawej — opcję *Kotlin/JVM* (jak pokazano na rysunku 1.4).

IntelliJ można używać do pisania kodu także w innych językach programowania niż Kotlin, takich jak: Java, Python, Scala oraz Groovy. Wybór opcji *Kotlin/JVM* informuje IDE, że chcemy pisać w języku Kotlin. Konkretnie rzecz biorąc, informuje ona, że chcemy pisać kod *przeznaczony* do uruchamiania na wirtualnej maszynie Javy. Jedną z zalet języka Kotlin jest to, że udostępnia on zestaw narzędzi pozwalających na uruchamianie pisanego w nim kodu w różnych systemach operacyjnych i na różnych platformach.

(Od tego miejsca określeń „wirtualna maszyna Javy" oraz „JVM" będziemy używali zamiennie, gdyż oba stanowią standardowe określenia stosowane w środowisku programistów języka Java. Więcej informacji dotyczących pisania kodu przeznaczonego do uruchamiania na JVM można znaleźć pod koniec tego rozdziału, w podrozdziale pt. „Dla ciekawskich: Pisanie kodu przeznaczonego na JVM").

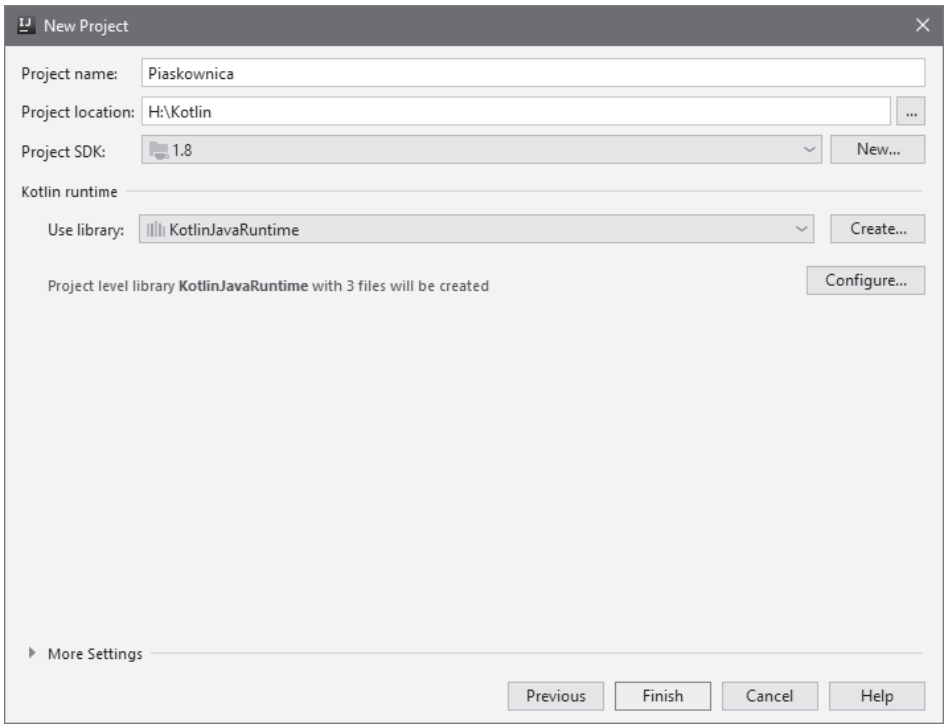


Rysunek 1.4. Tworzenie projektu Kotlin/JVM

Następnie w oknie dialogowym *New Project*, należy kliknąć przycisk *Next*. W efekcie IntelliJ wyświetli kolejne okno dialogowe (patrz rysunek 1.5), w którym można podać ustawienia tworzonego projektu. W polu *Project Name* wpiszemy nazwę projektu: `Piaskownica`. Kolejne pole *Project location* zostanie automatycznie wypełnione przez IDE. Tę domyślnie wybraną lokalizację projektu można pozostawić bez zmian bądź też można wskazać nowe miejsce — w tym celu należy kliknąć przycisk … umieszczony z prawej strony pola. Z listy w polu *Project SDK* należy wybrać opcję *1.8*, aby zaznaczyć, że w projekcie ma być używany Java Development Kit (JDK) w wersji 8.

Dlaczego do pisania programów Kotlinie potrzebny jest JDK? Otóż JDK daje IntelliJ dostęp do wirtualnej maszyny Javy oraz narzędzi tego języka, które są konieczne do konwersji kodu w Kotlinie do postaci kodów bajtowych (więcej na ich temat napiszemy w dalszej części rozdziału). Z technicznego punktu widzenia można używać każdej wersji JDK, o ile będzie to wersja 6 lub nowsza. Z naszych doświadczeń wynika jednak, że najlepiej do tego celu używać JDK 8; tak przynajmniej było w czasie pisania niniejszej książki.

Jeśli na liście w polu *Project SKD* nie jest widoczna opcja *1.8*, oznacza to, że ta wersja Javy nie jest zainstalowana na komputerze. W takim przypadku, przed przystąpieniem do dalszej lektury należy zainstalować tę wersję JDK. JDK 8 w wersji dla odpowiedniego systemu operacyjnego można pobrać ze strony: *https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html*. Po zainstalowaniu JDK należy ponownie uruchomić IntelliJ i jeszcze raz wykonać wszystkie opisane wcześnie czynności, aby utworzyć nowy projekt.



Rysunek 1.5. Określanie nazwy projektu

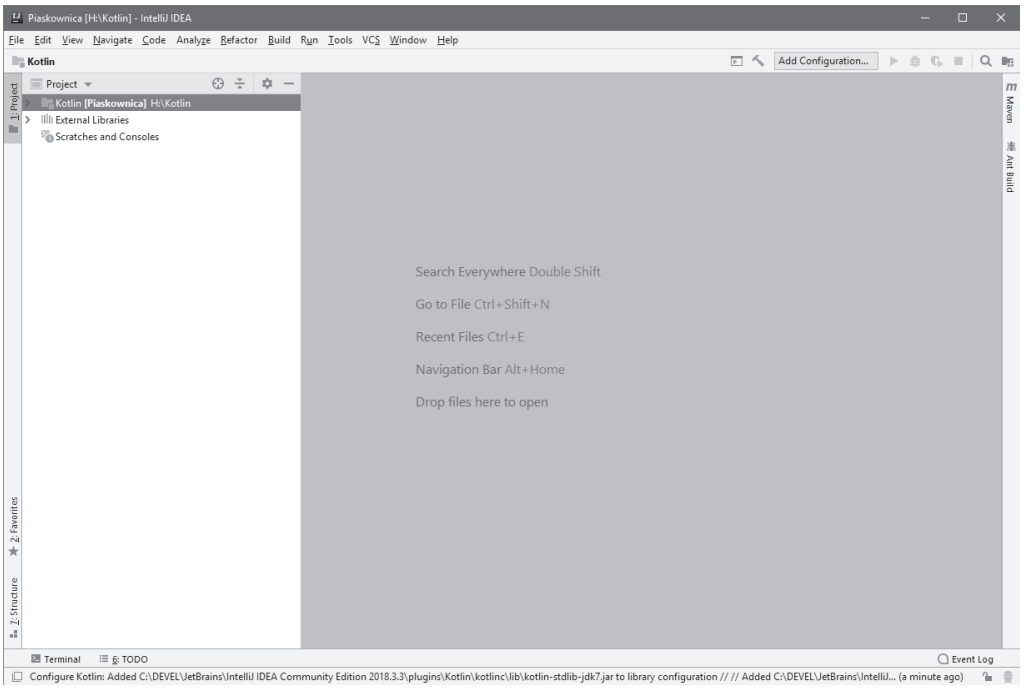
Kiedy opcje będą wyglądały tak jak na rysunku 1.5, można kliknąć przycisk *Finish*.

W efekcie IntelliJ wygeneruje nowy projekt o nazwie Piaskownica i wyświetli go w głównym oknie IDE składającym się z dwóch paneli (patrz rysunek 1.6). Generując nowy projekt, IntelliJ utworzy na dysku jego katalog, a w nim zestaw podkatalogów i plików, przy czym umieści je w katalogu wskazanym w polu *Project location*.

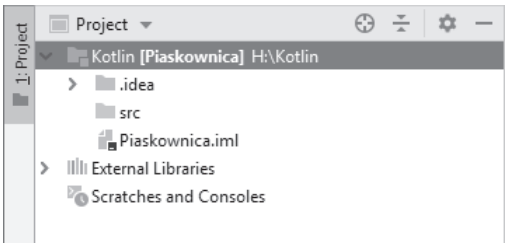
Lewy panel prezentuje *okno narzędzia Project*. Panel z prawej strony początkowo będzie pusty. To właśnie w nim, korzystając z *edytora*, będziemy przeglądać i edytować pliki źródłowe pisane w języku Kotlin. Zwróćmy teraz uwagę na okno narzędzia *Project* widoczne z lewej strony i kliknijmy strzałkę umieszczoną z lewej strony nazwy projektu *Piaskownica*. Spowoduje to wyświetlenie listy katalogów i plików wchodzących w skład projektu, jak pokazaliśmy na rysunku 1.7.

Projekt zawiera wszystkie pliki źródłowe tworzonego programu, jak również informacje o jego konfiguracji i zależnościach. Projekt można także podzielić na jeden lub większą liczbę *modułów*, które można sobie wyobrazić jako mniejsze podprojekty. Domyślnie każdy nowy projekt zawiera tylko jeden moduł i to nam w zupełności wystarczy na potrzeby naszego prostego pierwszego projektu.

Plik *Piaskownica.iml* zawiera informacje konfiguracyjne dotyczące naszego jedynego modułu. Katalog *.idea* zawiera pliki z ustawieniami dotyczącymi całego projektu, jak również z informacjami o naszych iteracjach z samym IDE (na przykład: z informacjami o plikach otworzonych w edytorze). Na razie można pozostawić pliki wygenerowane podczas tworzenia projektu.



Rysunek 1.6. Domyślne okno IntelliJ składające się z dwóch paneli



Rysunek 1.7. Widok projektu

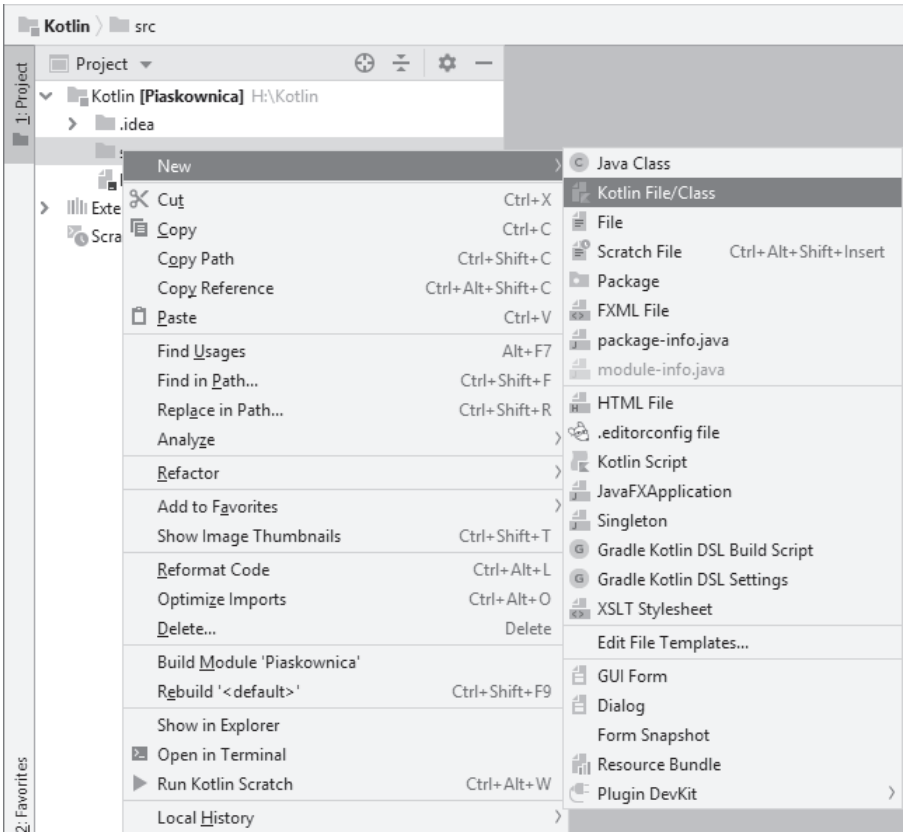
Opcja *External Libraries* zawiera informacje o bibliotekach, od których zależy działanie projektu. Jeśli ją rozwiniemy, zauważymy że IntelliJ automatycznie dodało do niej dwie zależności: *1.8* (reprezentującą Java 1.8) oraz *KotlinJavaRuntime*.

(Więcej informacji na temat struktury projektów IntelliJ można znaleźć w dokumentacji na witrynie firmy JetBrains pod linkiem: *https://www.jetbrains.org/intellij/sdk/docs/basics/project_structure.html*).

Pliki źródłowe tworzone w ramach projektu będą umieszczane w katalogu *src*. A skoro o plikach źródłowych mowa, to możemy utworzyć taki plik i wyświetlić go w edytorze.

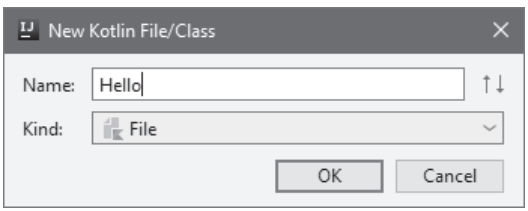
Tworzenie pierwszego pliku źródłowego w Kotlinie

Aby utworzyć plik źródłowy, należy kliknąć prawym przyciskiem myszy katalog *src* wyświetlony w oknie narzędzia *Project*, a następnie z wyświetlonego menu kontekstowego (patrz rysunek 1.8), wybrać opcję *New/Kotlin File/Class*.



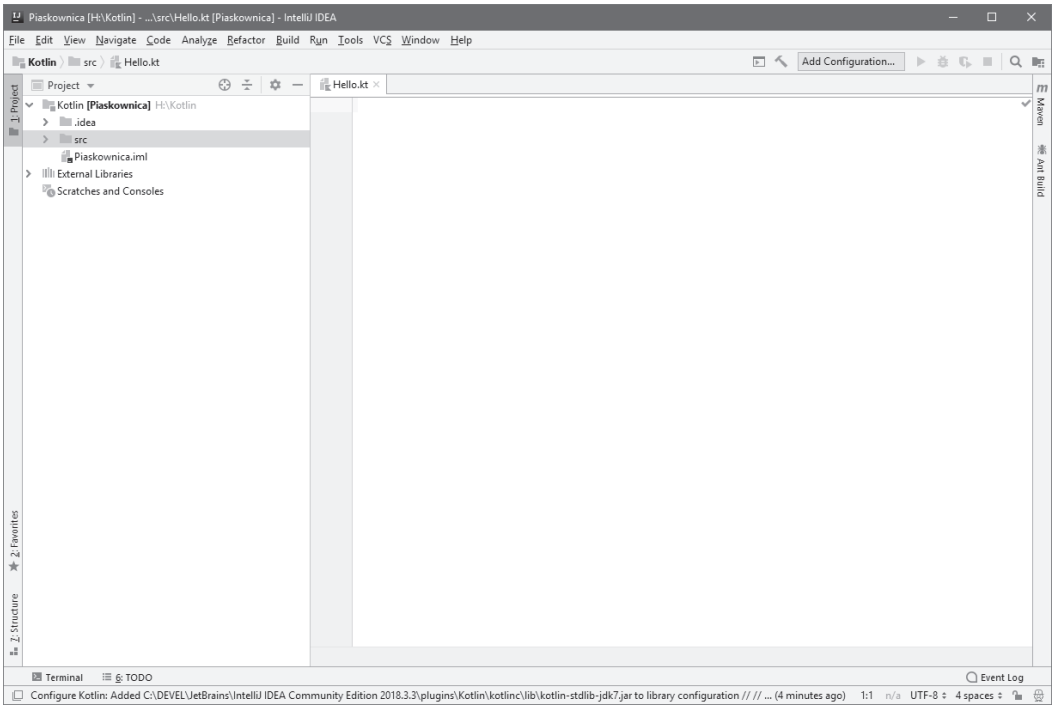
Rysunek 1.8. Tworzenie nowego pliku źródłowego w języku Kotlin

W wyświetlonym oknie dialogowym *New Kotlin File/Class* (patrz rysunek 1.9) w polu *Name* wpiszmy teraz nazwę pliku: `Hello`; opcję *Kotlin* wybraną na liście w polu *Kind* można zostawić bez zmian.



Rysunek 1.9. Określanie nazwy tworzonego pliku

Aby utworzyć plik, wystarczy teraz kliknąć przycisk *OK*. W efekcie IntelliJ utworzy plik *src\Hello.kt* i wyświetli jego zawartość w edytorze w prawym panelu okna IDE (patrz rysunek 1.10). Rozszerzenie *.kt* informuje, że jest to plik źródłowy z kodem napisanym w języku Kotlin; analogicznie: tak jak pliki źródłowe Javy mają rozszerzenie *.java*, a Pythona — *.py*.



Rysunek 1.10. Pusty plik Hello.kt wyświetlony w edytorze

Teraz można już przystąpić do napisania pierwszego fragmentu kodu w Kotlinie. Proszę zatem rozciągnąć palce i pisać w edytorze kod przedstawiony na listingu 1.1. (Warto przy tym zapamiętać, że w dalszej części książki kod który należy wpisać, będzie oznaczany pogrubioną czcionką).

Listing 1.1. „Witaj, świecie!" w Kotlinie (Hello.kt)

```
fun main(args: Array<String>) {
    println("Witaj, świecie!")
}
```

Ten kod będzie zapewne wyglądał obco. Nie ma się jednak, czym przejmować — pod koniec lektury tej książki czytanie i pisanie kodu w Kotlinie stanie się naszą drugą naturą. Jak na razie w zupełności wystarczy, byśmy zrozumieli ten kod na wysokim poziomie abstrakcji.

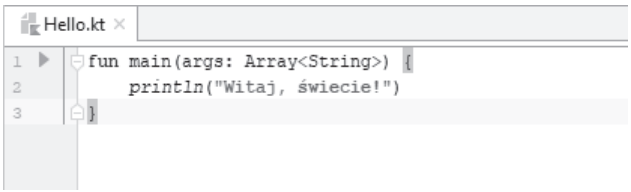
Kod z listingu 1.1 definiuje nową *funkcję*. Funkcja to grupa instrukcji, które kiedyś można wykonać. Wszelkie szczegółowe informacje dotyczące definiowania funkcji oraz sposobu ich działania można znaleźć w rozdziale 4.

Ta konkretna funkcja — **`main`** — ma w języku Kotlin specjalne znaczenie. Oznacza ona miejsce, od którego rozpocznie się wykonywanie programu. Jest ona określana jako *punkt wejścia aplikacji* (ang. *application entry point*), a w naszym projekcie (jak również w każdym innym) musi być zdefiniowany jeden taki punkt, by program można było uruchomić.

Powyższa funkcja **`main`** zawiera tylko jedną *instrukcję*: `println("Witaj, świecie!")`. **`println()`** to także funkcja, jednak tym razem wchodzi ona w skład *standardowej biblioteki Kotlina*. Kiedy program zostanie uruchomiony, a instrukcja `println("Witaj, świecie!")` wykonana, IntelliJ wyświetli zawartość podaną w nawiasach na ekranie (w tym przypadku zostanie wyświetlony łańcuch `Witaj, świecie!` bez cudzysłowów).

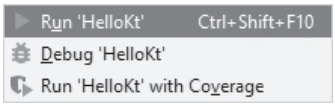
Wykonywanie pliku źródłowego

Krótko po zakończeniu wpisywania kodu z listingu 1.1 IntelliJ wyświetli z lewej strony jego pierwszego wiersza zieloną strzałkę ▶ nazywaną także „przyciskiem uruchamiania" (ang. *run button*; patrz rysunek 1.11). (Jeśli strzałka nie zostanie wyświetlona lub jeśli pod nazwą pliku na karcie albo pod którymkolwiek wierszem kodu pojawi się czerwona linia, będzie to oznaczało, że w kodzie jest błąd. W takim przypadku należy sprawdzić, czy kod został wpisany dokładnie tak jak na listingu 1.1. Z drugiej strony, jeśli zostanie wyświetlona czerwono-niebieska litera K stanowiąca symbol języka Kotlin, to będzie ona oznaczać to samo co przycisk uruchamiania).



Rysunek 1.11. Przycisk uruchamiania

W końcu nadszedł czas, by powołać program do życia i by powitał on cały świat. Wystarczy w tym celu kliknąć przycisk uruchamiania i z wyświetlonego menu wybrać opcję *Run 'HelloKt'* (patrz rysunek 1.12).

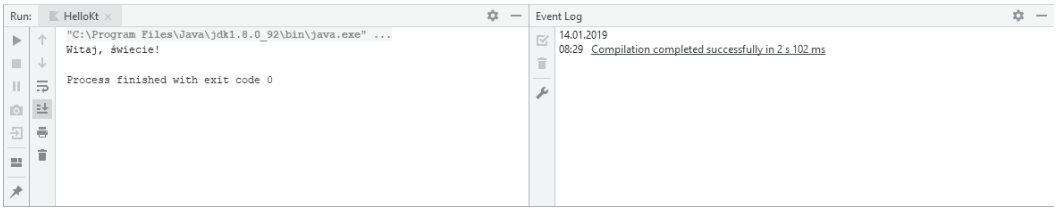


Rysunek 1.12. Wykonywanie pliku Hello.kt

Po uruchomieniu programu IntelliJ wykona wiersz po wierszu kod umieszczony pomiędzy nawiasami klamrowymi ({}), a następnie zakończy jego działanie. Jednocześnie u dołu głównego okna IntelliJ zostaną wyświetlone dwa nowe okna narzędzi (pokazane na rysunku 1.13).

Z lewej strony widoczne będzie *okno narzędzi Run* nazywane także *oknem konsoli* (i tak też od tej pory będziemy je nazywać). Wyświetlane są w nim informacje na temat tego, co się stało po uruchomieniu programu przez IntelliJ, jak również wszelkie wyniki generowane przez program. W naszym przypadku w oknie konsoli zostanie wyświetlony komunikat `Witaj, świecie!`. Poniżej niego widoczny będzie kolejny komunikat: `Process finished with exit code 0`[1] oznaczającym prawidłowe wykonanie programu.

[1] Proces zakończony z kodem wyjścia o wartości 0 — *przyp. tłum.*



Rysunek 1.13. Okna narzędzi Run oraz Event log

Ten wiersz pojawia się na samym końcu informacji prezentowanych w oknie konsoli, kiedy podczas wykonywania programu nie pojawią się żadne błędy; w dalszej części książki, prezentując wyniki generowane w oknie konsoli, będziemy pomijać ten komunikat końcowy.

(Użytkownicy systemu macOS mogą także zobaczyć w oknie konsoli czerwony komunikat błędu informujący o jakichś problemach z `JavaLauncherHelper`. Nie trzeba jednak zwracać na niego uwagi. Jest to niefortunny efekt uboczny sposobu w jaki Java Runtime Environment — środowisko uruchomieniowe Javy — jest instalowane w systemie macOS. Wyeliminowanie tego problemu wymagałoby sporo pracy, a w zasadzie w niczym o nie szkodzi — komunikat ten można zignorować).

Z prawej strony widoczne będzie natomiast *okno narzędzia Event Log* (nazywane także *dziennikiem zdarzeń*), a w nim informacje o tym, co IntelliJ zrobiło, by przygotować program do wykonania. O dzienniku zdarzeń nie będziemy już wspominać w dalszej części książki, gdyż znacznie więcej interesujących informacji można uzyskać w oknie konsoli. (Z tego względu nie trzeba się także przejmować, jeśli w przyszłości okno dziennika zdarzeń nie będzie już otwierane). Okno to można zamknąć, klikając przycisk — umieszczony w jego prawym górnym rogu.

Kompilacja i wykonywanie kodu w Kotlinie

W tym krótkim czasie pomiędzy naciśnięciem przycisku uruchamiania a wyświetleniem komunikatu `Witaj, świecie!` w oknie konsoli dzieje się bardzo dużo.

W pierwszej kolejności IntelliJ *kompiluje* kod źródłowy napisany w Kotlinie przy użyciu kompilatora `kotlin-jvm`. Oznacza to, że IntelliJ tłumaczy kod źródłowy na *kody bajtowe Javy* — język, którym porozumiewa się JVM. Jeśli kompilator napotka jakiekolwiek błędy podczas tłumaczenia kodu źródłowego na kody bajtowe, wyświetli stosowny komunikat błędu (lub większą liczbę takich komunikatów), który może nam pomóc w rozwiązaniu problemu. W przeciwnym razie, jeśli proces kompilacji nie napotka żadnych problemów, IntelliJ przejdzie do fazy wykonywania programu.

W fazie wykonywania kody bajtowe wygenerowane wcześniej przez kompilator `kotlin-jvm` zostają wykonane przez wirtualną maszynę Javy. W oknie konsoli wyświetlane są jedynie wyniki generowane przez program, takie jak teksty podane w wywołaniach funkcji **`println()`**.

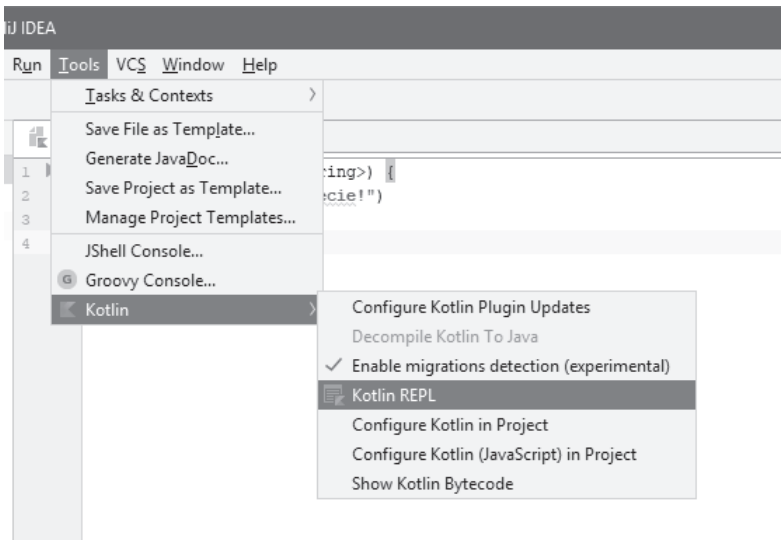
Kiedy nie będzie już żadnych kolejnych kodów bajtowych do wykonania, JVM zakończy działanie. Kod zakończenia zostanie wyświetlony w oknie konsoli, informując w ten sposób o tym, czy program został wykonany prawidłowo, czy też zwrócił jakiś kod błędu.

Do lektury tej książki nie będzie potrzebna dokładna znajomość procesu kompilacji kodu źródłowego pisanego w Kotlinie, niemniej jednak zagadnienia związane z kodami bajtowymi zostały opisane w rozdziale 2.

Kotlin REPL

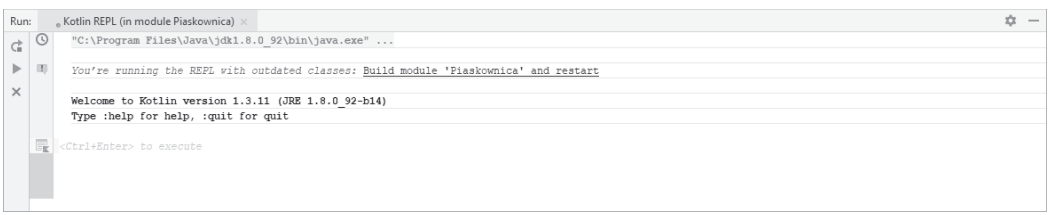
Czasami może się zdarzyć, że będziemy chcieli przetestować niewielkie fragmenty kodu napisanego w Kotlinie i sprawdzić, co się dzieje podczas ich wykonywania — podobnie jak moglibyśmy zapisywać kolejne etapy prostych obliczeń na kawałku papieru. Taka możliwość będzie szczególnie przydatna podczas nauki pisania w języku Kotlin. Na szczęście IntelliJ udostępnia narzędzie do szybkiego sprawdzania kodu bez konieczności zapisywania go w odrębnym pliku. Narzędzie to nosi nazwę *Kotlin REPL*. Już za moment wyjaśnimy znaczenie tej nazwy, na razie otwórzmy to narzędzie, by przekonać się, co za jego pomocą można zrobić.

Aby otworzyć narzędzie *Kotlin REPL*, należy wybrać z menu głównego IDE opcję *Tools/Kotlin/Kotlin REPL* (patrz rysunek 1.14).



Rysunek 1.14. Wyświetlanie okna narzędzia Kotlin REPL

Okno narzędzia *Kotlin REPL* zostanie wyświetlone u dołu głównego okna IDE (patrz rysunek 1.15).



Rysunek 1.15. Okno narzędzia Kotlin REPL

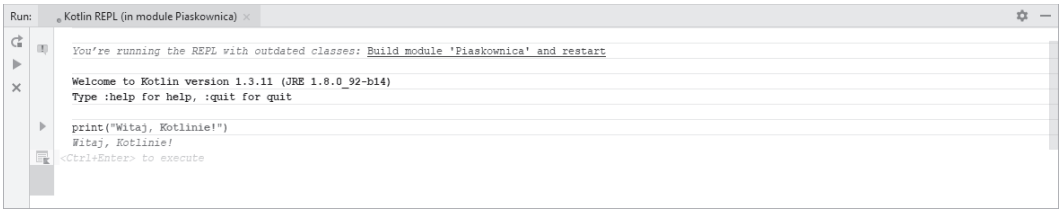
W oknie *Kotlin REPL* można wpisywać kod tak samo jak w edytorze. Różnica polega jednak na tym, że tu wpisany kod może zostać błyskawicznie przetworzony bez konieczności kompilowania całego projektu.

Spróbujmy wpisać w tym oknie poniższy fragment kodu (patrz listing 1.2).

Listing 1.2. „Witaj, Kotlinie!" (REPL)

```
println("Witaj, Kotlinie!")
```

Po wpisaniu tego kodu należy nacisnąć kombinację klawiszy *Ctrl+Enter*, a w efekcie kod zostanie przetworzony i wykonany. Po krótkiej chwili poniżej kodu zostaną także wyświetlone wyniki; w naszym przypadku będzie to tekst *Witaj, Kotlinie!* (patrz rysunek 1.16).



Rysunek 1.16. Przetwarzanie kodu źródłowego

REPL to skrót od angielskich słów: „read, evaluate, print, loop"[2]. W oknie *Kotlin REPL* wpisujemy kod, a następnie przesyłamy go, klikając zieloną strzałkę wyświetloną z prawej strony okna lub naciskając kombinację klawiszy *Ctrl+Enter*. Następnie narzędzie REPL *odczytuje* kod, *przetwarza* go (czyli wykonuje), *wyświetla* wyniki lub efekty uboczne. Po zakończeniu wykonywania kodu narzędzie zwraca sterowanie, a cały proces można *powtórzyć* od nowa.

I w ten sposób rozpoczęliśmy podróż po języku Kotlin! W tym rozdziale udało się nam zrobić całkiem dużo — zbudowaliśmy fundamenty do dalszego poznawania Kotlina. W następnym rozdziale zajmiemy się szczegółami języka, a konkretnie: zmiennymi, stałymi oraz typami reprezentującymi dane.

Dla ciekawskich: Dlaczego warto używać właśnie IntelliJ?

Kod źródłowy w języku Kotlin można pisać w dowolnym edytorze tekstów. Niemniej jednak zachęcamy do korzystania z IntelliJ, zwłaszcza na etapie uczenia się programowania w Kotlinie. Podobnie jak dobry edytor tekstów wyposażony w narzędzia do sprawdzania ortografii i gramatyki potrafi ułatwić pisanie poprawnej prozy, tak IntelliJ ułatwia pisanie dobrego i właściwie sformatowanego kodu w Kotlinie. Oto kilka rzeczy, w których pomaga IntelliJ:

- pisanie kodu poprawnego syntaktycznie i semantycznie, dzięki takim mechanizmom jak: kolorowanie składni, podpowiedzi kontekstowe oraz automatyczne uzupełnianie kodu;
- uruchamianie i debugowanie kodu dzięki takim mechanizmom jak: punkty wstrzymania oraz krokowe wykonywania kodu bezpośrednio podczas działania aplikacji;
- modyfikowanie struktury kodu dzięki opcjom refaktoryzacji (takim jak zmienianie nazw lub wyodrębnianie stałych) oraz formatowania kodu (pozwalającym uporządkować wcięcia i odstępy).

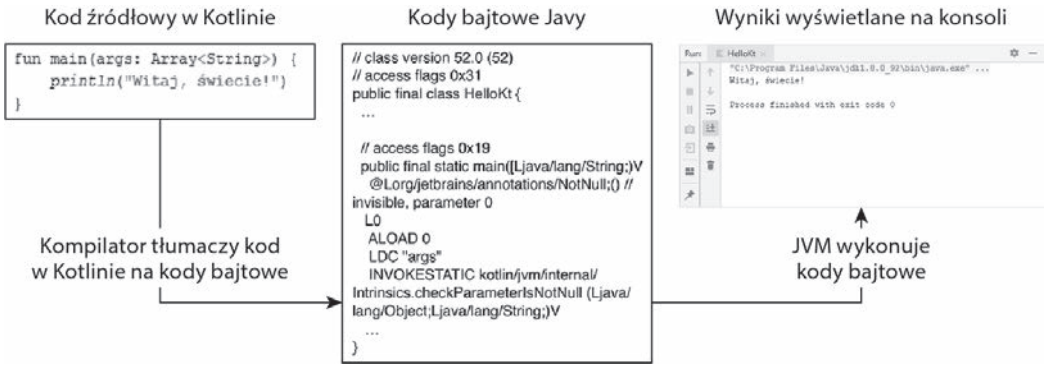
Co więcej, ponieważ Kotlin został opracowany przez firmę JetBrains, integracja tego języka z IntelliJ została uważnie zaprojektowana — co bardzo często można zauważyć i docenić podczas edycji kodu.

[2] Czytaj, przetwarzaj, wyświetlaj, powtarzaj w pętli — *przyp. tłum.*

Dodatkową zaletą IntelliJ jest fakt, że stanowi podstawę środowiska programistycznego Android Studio, a zatem skróty i narzędzia, których nauczymy się teraz, będzie można później wykorzystać podczas pisania w Android Studio.

Dla ciekawskich: Pisanie kodu przeznaczonego na JVM

JVM to oprogramowanie, które wie, jak wykonywać pewien zestaw instrukcji nazywanych kodami bajtowymi. „Pisanie kodu przeznaczonego na JVM" oznacza kompilowanie oraz tłumaczenie kodu źródłowego napisanego w języku Kotlin na kody bajtowe Javy w celu wykonywania ich na wirtualnej maszynie Javy (patrz rysunek 1.17).



Rysunek 1.17. Proces kompilacji i wykonywania

Każda platforma systemowa taka jak Windows czy macOS ma własny zestaw instrukcji. JVM działa jako swoisty pomost pomiędzy kodami bajtowymi a różnymi środowiskami sprzętowymi i systemowymi, na jakich ona działa, odczytując kody bajtowe i wykonując odpowiednie, odpowiadające im instrukcje używanej platformy. Właśnie z tego względu istnieje wiele wersji JVM przeznaczonych do użycia na różnych platformach systemowych i sprzętowych. To właśnie dzięki temu programiści używający Kotlina mogą pisać kod niezależny od platformy, który można napisać raz, a następnie skompilować do postaci kodów bajtowych i wykonywać na różnych urządzeniach niezależnie od używanego na nich systemu operacyjnego.

Ponieważ kod napisany w Kotlinie można przekształcić na kody bajtowe, które z kolei mogą być wykonywane przez JVM, dlatego uważa się go za język JVM. Prawdopodobnie najbardziej znanym językiem JVM jest oczywiście Java, gdyż to ona powstała jako pierwsza. Niemniej jednak istnieje więcej języków JVM, takich jak Kotlin lub Scala, które zostały opracowane, by uzupełnić pewne niedostatki Javy występujące w niej z programistycznego punktu widzenia.

Możliwości Kotlina nie ograniczają się jednak do wykonywania napisanych w nim programów tylko na JVM. W czasie pisania niniejszej książki można je było kompilować do kodu JavaScript, a nawet do rodzimych kodów konkretnej platformy systemowej, takiej jak Windows czy macOS, eliminując w ten sposób konieczność stosowania warstwy maszyny wirtualnej.

Wyzwanie: Arytmetyka REPL

Na końcu wielu rozdziałów tej książki umieszczonych zostało jedno wyzwanie lub kilka wyzwań. Są one przeznaczone do samodzielnego wykonania w celu pogłębienia wiedzy i nabrania dodatkowego doświadczenia.

Celem tego wyzwania jest poznanie operatorów arytmetycznych stosowanych w Kotlinie: `+`, `-`, `*` oraz `/`. Na przykład w oknie *Kotlin REPL* można wpisać wyrażenie: `(9+12)/2`. Czy wyświetlone wyniki będą zgodne z oczekiwaniami?

Aby zdobyć więcej informacji na temat tego zagadnienia, można przejrzeć informacje na temat funkcji matematycznych dostępnych w standardowej bibliotece języka Kotlin (są one dostępne na stronie: *https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.math/index.html*) i wypróbować jedną z nich, na przykład `min(94, -99)`, która zwraca mniejszą z dwóch liczb podanych w nawiasach.

Skorowidz

A

B

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

W

Z

Ż

Kotlin: najlepsze rozwiązania nie tylko dla Androida!

Gdy w 2017 roku na konferencji Google ogłoszono, że Kotlin jest jednym z oficjalnie wspieranych języków programowania aplikacji dla Androida, język ten błyskawicznie stał się popularny. Najpoważniejsze firmy technologiczne doceniają jego zalety, takie jak zwarta składnia i integracja z kodem Javy. Co ważne, twórcy Kotlina czerpali z doświadczeń projektantów Javy i zapewnili, że te dwa języki są ze sobą ściśle powiązane. W ten sposób świeżość rozwiązań i nowoczesne mechanizmy zostały znakomicie połączone z najlepszymi cechami Javy, a Kotlin stał się wszechstronnym wieloplatformowym językiem programowania.

Ta książka jest przeznaczona dla programistów, którzy chcą poznać język o możliwościach wykraczających poza Javę i pisać solidne aplikacje dla różnych platform. Pozwala w praktyczny sposób zapoznać się z unikalnymi możliwościami Kotlina i — na podstawie licznych przykładowych projektów — stopniowo zgłębiać złożone zagadnienia. Znakomitym uzupełnieniem prezentowanych treści są informacje o wewnętrznych mechanizmach działania języka wraz z ćwiczeniami do samodzielnego wykonania.

W tej książce między innymi:

- wprowadzenie do Kotlina
- funkcje w Kotlinie, w tym funkcje anonimowe
- praca na obiektach: dziedziczenie, klasy, klasy abstrakcyjne
- programowanie funkcyjne w Kotlinie
- najciekawsze biblioteki i współdziałanie z Javą

Big Nerd Ranch mieści się w Atlancie w USA. Firma została założona w 2001 roku przez Aarona Hillegassa, znakomitego programistę i trenera programowania. Jej misją jest krzewienie wiedzy o rozwijających się technologiach mobilnych. Słynie ze świetnych szkoleń i znakomitych trenerów.

Josh Skeen szkoli dla Big Nerd Ranch — uczy Javy, Kotlina i pisania aplikacji dla Androida. Programowanie fascynuje go jako forma sztuki interaktywnej. W wolnych chwilach biega, trenuje brazylijskie jiu-jitsu i bawi się syntezatorem modularnym.

David Greenhalgh prowadzi kursy w Big Nerd Ranch. Uwielbia odkrywać nowe możliwości programowania dla Androida. Dla relaksu gotuje, czyta i narzeka na Buffalo Bills.

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 978-83-283-5536-1

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 79,00 zł

Big Nerd Ranch