JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 13 (2019) 354-358
Wysłane: 2019-09-14
Przyjęte: 2019-09-22

Analiza porównawcza języków Kotlin i Java używanych do tworzenia aplikacji na system Android

Daniel Sulowski*, Grzegorz Kozieł
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Niniejsza publikacja przedstawia rezultaty analizy porównawczej języków programowania Java i Kotlin, wykorzystywanych do tworzenia aplikacji mobilnych przeznaczonych do pracy pod kontrolą systemu Android. Analiza obejmuje aspekty związane z wydajnością, takie jak obciążenie procesora, obciążenie pamięci RAM, a także czasy kompilacji i wykonania programu. Pod uwagę wzięto również strukturę kodu, dostępność bibliotek, obsługiwane bazy danych, popularność oraz wsparcie społeczności.

Słowa kluczowe: Android; Java; Kotlin; wydajność

*Autor do korespondencji.

Adres e-mail: daniel.sulowski95@gmail.com

Comparative analysis of Kotlin and Java languages used to create applications for the Android system

Daniel Sulowski*, Grzegorz Kozieł
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This publication presents the results of a comparative analysis of Java and Kotlin programming languages used to create mobile applications for system Android. The analysis covers performance aspects such as CPU load, RAM load, as well as the compilation and execution times. Aspects such as code structure, availability of libraries, supported databases, popularity and community support were taken under consolidation.

Keywords: Android; Java; Kotlin; performance

*Corresponding author.

E-mail address: daniel.sulowski95@gmail.com

1. Wstęp

W październiku 2008 r. wydano pierwszą wersję systemu Android. Od tego momentu aż do 2017 r. Java była jedynym językiem programowania wykorzystywanym do tworzenia aplikacji mobilnych przeznaczonych do pracy pod kontrolą tego systemu [1]. Na konferencji Google I/O w 2017 roku ogłoszono bowiem oficjalne wsparcie dla języka Kotlin przy programowaniu aplikacji mobilnych. Głównym powodem stworzenia tego języka była chęć eliminacji błędów powstałych już w czasie projektowania języka Java. Kotlin miał za zadanie wyeliminowanie sytuacji mogących wywołać nieoczekiwane zamknięcie aplikacji (*null-pointer safety*), a także ułatwienie procesu implementacji aplikacji mobilnych. Istotnym aspektem była również łatwa możliwość migracji kodu z Javy do Kotlina. Ponadto, języki te pozostają do siebie bardzo podobne i generują zbliżony kod bajtowy [2].

Pomimo podobnej struktury kodu, ta sama aplikacja napisana w obydwu językach może w inny sposób wpływać na podzespoły urządzenia na którym pracuje. Z samej dokumentacji Kotlina wynika, że mogą wystąpić różnice w wykorzystaniu pamięci RAM. Autorzy tej dokumentacji podają również, że aplikacje pisane w Kotlinie cechują się krótszym czasem wykonania [2]. Ważne są również różnice związane z procesem projektowania i implementacji. Zmieniona struktura kodu oraz wprowadzenie nowej funkcjonalności może wpływać na czas i koszt tworzenia oprogramowania. Nie bez znaczenia pozostaje trudność danej technologii. Mimo podobnych zastosowań jeden język może być znacznie prostszy w użyciu aniżeli drugi.

2. Cel badań

Niniejszy artykuł ma na celu przeprowadzenie analizy porównawczej języków programowania Java i Kotlin. Skupiono się przede wszystkim na wydajności tworzonych aplikacji. Zbadano takie parametry jak obciążenie procesora, obciążenie pamięci RAM oraz czasy kompilacji i wykonania programu. Pod uwagę wzięto także budowę kodu, dostępność bibliotek, obsługiwane bazy danych oraz popularność i wsparcie społeczności.

3. Metoda badań

W celu zbadania wydajności języków zaimplementowano specjalną aplikację mobilną, działającą w identyczny sposób zarówno w Javie, jak i Kotlinie. Jej działanie polega na generowaniu liczb i sortowaniu dużych zbiorów danych z wykorzystaniem kilku algorytmów oraz wyświetlaniu prostej animacji.

Badania dotyczące wydajności przeprowadzono na 3 różnych urządzeniach:

- Xiaomi Redmi 4A,
- Xiaomi Redmi 7,
- LG K350N.

Odizolowano aplikację od czynników zewnętrznych mogących wpłynąć na wynik pomiaru. W tym celu wyłączono połączenie z internetem, GPS, moduł Bluetooth, a także zamknięto aplikacje działające w tle. Uruchomiono również tryb samolotowy i ustawiono maksymalną jasność ekranu. Wyniki badań odczytano za pomocą środowiska programistycznego Android Studio. Do odczytu obciążenia procesora i pamięci RAM wykorzystano specjalny moduł Android Profiler. Łącznie dokonano 54 pomiarów każdego parametru na wszystkich urządzeniach.

Na podstawie literatury oraz doświadczeń związanych z programowaniem dedykowanej aplikacji porównano strukturę kodu badanych języków. Zbadano popularność oraz wsparcie społeczności w oparciu o dane zebrane z takich serwisów jak Stackoverflow czy Github. Sprawdzono również dostępność bibliotek oraz obsługiwane bazy danych.

4. Analiza literatury

Pierwszym istotnym źródłem wiedzy są oficjalne dokumentacje. Już z samej dokumentacji Kotlina można się dowiedzieć o różnicach w składni oraz bezpieczeństwie tworzonych aplikacji. Wprowadzono wiele udogodnień dla zaawansowanych programistów oraz znacznie zredukowano rozmiar kodu. Przekłada się to na krótszy czas potrzebny do implementacji programu. Skrócony kod jest jednak mniej intuicyjny, zwłaszcza dla niedoświadczonych programistów. Ograniczono również możliwość wystąpienia nieoczekiwanego zamknięcia aplikacji. Ponadto, dokumentacja Kotlina sugeruje, że programy napisane w tym języku działają szybciej niż programy napisane w Javie. Występują również różnice w czasie kompilacji. Kotlin osiąga lepszy czas w przypadku kompilacji przyrostowej, z kolei Java szybciej kompiluje program od zera [2].

Pracę o podobnej tematyce opublikował w maju 2018 Patrick Schwermer z Królewskiego Instytutu Technicznego w Sztokholmie. Autor dowodzi, że programy napisane w Javie działają nieznacznie szybciej niż programy napisane w Kotlinie, co jest sprzeczne z oficjalną dokumentacją Kotlina. Ponadto, programy napisane w Kotlinie mają zużywać więcej pamięci, ale lepiej funkcjonuje mechanizm odpowiedzialny za jej oczyszczanie [3]. Autorzy pracy „A comparative Study: Java vs Kotlin Programming in Android Application Development" wydanej w czerwcu 2018 podają, że Java jest lepszym wyborem dla początkujących programistów, ponieważ cechuje się prostszą składnią i większym wsparciem społeczności. Kotlin z kolei zdaje się być lepszym narzędziem w rękach doświadczonego programisty, ponieważ pisany kod jest krótszy oraz trudniej jest w nim popełnić błędy [4]. Do podobnych wniosków doszli autorzy pracy „Are you still smelling it?: A comparative study between Java and Kotlin language". Zbadano zapach kodu na przykładzie 50 projektów Java oraz 50 projektów Kotlin dostępnych na repozytorium Github. Autorzy udowadniają, że to Kotlin cechuje się mniejszym zapachem kodu [5].

Pozostałe prace są w zasadzie podsumowaniem informacji zawartych w dokumentacji Kotlina i nie wnoszą niczego nowego. Kotlin jest oficjalnie wspierany przez Androida od niespełna 2 lat, zatem temat ten nie jest jeszcze dostatecznie przebadany.

5.1. Analiza porównawcza cech badanych języków

W tabeli 1 zestawiono największe różnice pomiędzy strukturą kodu Javy i Kotlina.

Tabela 1. Różnice w budowie kodu pomiędzy Javą i Kotlinem

Java	Kotlin
Wyrażenia lambda wprowadzono w wersji 8, która nie jest jeszcze w pełni wspierana przez Androida [6]	Wprowadzono wyrażenia lambda używane do przekazywania bloku kodu jako argumentu funkcji
Brak funkcji jednoliniowych (*inline functions*)	Wprowadzenie funkcji jednoliniowych
Konieczność deklaracji każdego elementu interfejsu	Możliwość bezpośredniego odwołania się do każdego elementu interfejsu
Brak mechanizmów chroniących przed wyjątkiem *NullPointerException*	Wprowadzenie operatorów eliminujących wyjątek *NullPointerException* [7]
Obowiązek sprawdzenia wystąpienia wyjątków [6]	Wystąpienie wyjątku nie musi być sprawdzane
Konieczność deklaracji typów danych	Zwolnienie programisty w obowiązku deklaracji typów danych [8]
Istnienie prymitywnych typów danych (np. *int*, *char*)	Wszystkie zmienne są obiektami
Istnienie modyfikatora *static*	Usunięto modyfikator *static*. Wprowadzenie tzw. obiektów towarzyszących (*companion objects*) [9]
Dobrej jakości kod powinien zawierać metody *get* i *set*	Metody *get* i *set* generowane są automatycznie i pozostają niewidoczne w kodzie źródłowym
Średnik występuje na końcu każdej instrukcji	Brak średnika na końcu instrukcji [9]

Odświeżona składnia języka Kotlin pozwala pisać zdecydowanie krótszy kod. Przekłada się to na czas pisania programu. W przypadku niektórych aplikacji, Kotlin może zredukować rozmiar kodu nawet o kilkaset linijek, jeżeli weźmiemy pod uwagę brak konieczności sprawdzania wystąpienia wyjątków czy pisania metod *get* i *set*. Poniższe listingi przedstawiają tę samą klasę napisaną w Javie oraz w Kotlinie. Kilkanaście linijek kodu Javy odpowiada jednej linijce kodu w Kotlinie.

Przykład 1. Klasa Student w Javie

```
class Student {
   private String name;
   private int age;

   public String getName() {
      return name;
   }
```

```
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

Przykład 2. Klasa Student w Kotlinie

```
class Student(var name: String, var age : Int)
```

Ponadto, Kotlin oferuje nową funkcjonalność taką jak np. wyrażenia lambda czy funkcje jednoliniowe. Powoduje to, że będzie on zdecydowanie lepszym narzędziem w rękach doświadczonego programisty. Korzystając z Kotlina, można uzyskać zamierzony cel, wykorzystując znacznie mniej kodu i czasu niż w przypadku Javy. Osoba początkująca powinna jednak rozpocząć swoją naukę od Javy. Język ten cechuje się bowiem bardziej intuicyjnym i klarownym kodem. Warto dodać, że w internecie znaleźć można zdecydowanie więcej materiałów do nauki Javy, niż do nauki Kotlina. Jest to oczywiście związane z tym, że Kotlin jest językiem stosunkowo młodym, natomiast Java istnieje na rynku od ponad 20 lat. W tym czasie wydano wiele książek do nauki Javy, w praktycznie wszystkich językach. Kotlin oferuje w zamian narzędzie online zwane Kotlin Koans. Jest to specjalny moduł oferujący interaktywną naukę tego języka bez potrzeby pobierania dodatkowego oprogramowania. Jego wadą jest stosunkowo wysoki poziom trudności, jeżeli dana osoba nie ma doświadczenia z programowaniem.

Java cieszy się również znacznie większą popularnością. We wrześniu 2019 r. w serwisie Github znajdowało się ponad milion repozytoriów pod tagiem „Java" i nieco ponad 67 tysięcy pod tagiem „Kotlin". Natomiast w serwisie Stackoverflow zadanych zostało do tej pory aż 1,5 miliona pytań dotyczących Javy i nieco ponad 70 tysięcy pytań dotyczących Kotlina. W przyszłości jednak trend ten może się odwrócić i to Kotlin może stać się liderem, jeżeli chodzi o programowanie aplikacji mobilnych. Aktualnie można zaobserwować dużą migrację projektów z Javy do Kotlina. Zdecydowały się na to takie serwisy jak Twitter, Pinterest czy Netflix. Nawet twórcy Kotlina przewidzieli taką możliwość i udostępnili mechanizm pozwalający na konwersje kodu z Javy do Kotlina. Obydwa języki są interoperacyjne i kompilują się do podobnego kodu bajtowego [10]. Sprawia to, że języki te mogą być mieszane w dowolnym projekcie bez obaw o komplikacje z kompatybilnością. Tyczy się to również bibliotek i baz danych. Wszystkie narzędzia obsługiwane przez jeden język będą również obsługiwane przez drugi.

5.2. Analiza porównawcza wydajności badanych języków

Na podstawie wyników badań obliczono średnią, medianę, wartość minimalną oraz wartość maksymalną każdego badanego parametru. Uzyskane rezultaty zestawiono w formie tabeli.

5.2.1. Obciążenie procesora

Poniższe tabele zawierają dane statystyczne dotyczące obciążenia procesora. Tabela 2 odnosi się do Javy, z kolei tabela 3 do Kotlina.

Tabela 2. Obciążenie procesora - Java

Urządzenie	LG K350N	Xiaomi Redmi 4A	Xiaomi Redmi 7
Wartość minimalna	44,5%	40,9%	14,9%
Wartość maksymalna	53,5%	45,9%	18%
Średnia	48,94%	43,41%	16,37%
Mediana	49,4%	43,8%	16%

Tabela 3. Obciążenie procesora – Kotlin

Urządzenie	LG K350N	Xiaomi Redmi 4A	Xiaomi Redmi 7
Wartość minimalna	44,5%	40,8%	15%
Wartość maksymalna	52%	46%	18,8%
Średnia	48,77%	43,49%	16,3%
Mediana	49,3%	43,75%	16%

Poszczególne wartości w powyższych tabelach są do siebie bardzo zbliżone, a różnice nie przekraczają 1%. Zatem wybór technologii pomiędzy Javą a Kotlinem nie wpływa znacząco na obciążenie procesora.

5.2.2. Obciążenie pamięci RAM

W tabelach poniżej zaprezentowano jak kształtowało się obciążenie pamięci RAM na badanych urządzeniach. Tabela 4 dotyczy Javy, natomiast tabela 5 Kotlina.

Tabela 4. Obciążenie pamięci RAM - Java

Urządzenie	LG K350N	Xiaomi Redmi 4A	Xiaomi Redmi 7
Wartość minimalna	266,4 MB	263,6 MB	266,5 MB
Wartość maksymalna	275,3 MB	293,5 MB	278,9 MB
Średnia	268,8 MB	269,9 MB	269 MB
Mediana	268,5 MB	265,2 MB	268,9 MB

Tabela 5. Obciążenie pamięci RAM – Kotlin

Urządzenie	LG K350N	Xiaomi Redmi 4A	Xiaomi Redmi 7
Wartość minimalna	268,1 MB	263,4 MB	267,1 MB
Wartość maksymalna	277,2 MB	292,1 MB	279,3 MB
Średnia	269,6 MB	270,6 MB	270,4 MB
Mediana	269,5 MB	266,4 MB	269,8 MB

Na podstawie mediany i średniej można stwierdzić, że to Kotlin zużywa więcej pamięci. Różnica ta jest jednak bardzo mała i wynosi tylko 1 MB, jednak może się ona pogłębiać w przypadku mocno rozbudowanych programów. Różnice pomiędzy wartościami minimalnymi i maksymalnymi wynoszące do 30 MB mogą być spowodowane tym, że aplikacja generuje losowe obiekty i zbiory danych, które z każdym uruchomieniem zajmują inny obszar pamięci. Ponadto, pamięć RAM jest wykorzystywana w procesach systemowych, co również mogło mieć wpływ na wyniki badań.

5.2.3. Czas kompilacji

Jak podaje oficjalna dokumentacja Kotlina, aplikacje napisane w Javie powinny kompilować się ok. 15% szybciej niż aplikacje napisane w Kotlinie [2]. Tabela 6 przedstawia czas kompilacji w Javie, a tabela 7 czas kompilacji w Kotlinie. Czas ten został odczytany za pomocą środowiska programistycznego Android Studio.

Tabela 6. Czas kompilacji – Java

Urządzenie	LG K350N	Xiaomi Redmi 4A	Xiaomi Redmi 7
Wartość minimalna	899ms	802ms	838ms
Wartość maksymalna	1383ms	1336ms	1321ms
Średnia	1043ms	966ms	1043ms
Mediana	1020ms	943ms	1036ms

Tabela 7. Czas kompilacji – Kotlin

Urządzenie	LG K350N	Xiaomi Redmi 4A	Xiaomi Redmi 7
Wartość minimalna	879ms	900ms	895ms
Wartość maksymalna	1172ms	1812ms	2179ms
Średnia	999ms	1102ms	1193ms
Mediana	993ms	1057ms	1178ms

Rezultaty badań przeprowadzonych na telefonach Xiaomi są zgodne z oficjalną dokumentacją Kotlina. Java kompiluje programy o ok. 15% szybciej niż Kotlin. Niestety, sprzeczne wnioski dają badania przeprowadzone na LG K350N. Wynika z nich, że to Kotlin szybciej kompiluje programy (o ok. 3%). Trudno jest zatem na tej podstawie wysunąć jednoznaczne wnioski, ponieważ czas kompilacji oprócz technologii, jest też mocno uzależniony od samego urządzenia.

5.2.4. Czas wykonania

Czas wykonania programu jest w dużej mierze zależny od podzespołów, w które wyposażone jest badane urządzenie. Lepszy procesor i pamięć RAM o większej pojemności pozwalają osiągnąć krótszy czas wykonania. W tabeli 8 zestawiono czasy wykonania aplikacji Java, z kolei w tabeli 9 czasy wykonania aplikacji Kotlin. Czasy te zostały zmierzone programistycznie.

Tabela 8. Czas wykonania – Java

Urządzenie	LG K350N	Xiaomi Redmi 4A	Xiaomi Redmi 7
Wartość minimalna	43549ms	4046ms	2319ms
Wartość maksymalna	47212ms	4153ms	2401ms
Średnia	44301ms	4077ms	2338ms
Mediana	44081ms	4073ms	2335ms

Tabela 9. Czas wykonania – Kotlin

Urządzenie	LG K350N	Xiaomi Redmi 4A	Xiaomi Redmi 7
Wartość minimalna	40339ms	4627ms	2498ms
Wartość maksymalna	41078ms	4856ms	2540ms
Średnia	40686ms	4656ms	2518ms
Mediana	40694ms	4653ms	2518ms

Badania przeprowadzone na telefonach marki Xiaomi sugerują, że aplikacje napisane w Javie są szybsze od aplikacji napisanych w Kotlinie. Natomiast w przypadku LG K350N to Kotlin osiąga lepszy czas. Podobnie jak w przypadku czasu kompilacji, trudno jest jednoznacznie określić który język pozwala pisać szybsze programy.

6. Wnioski

W niniejszej publikacji dokonano analizy porównawczej języków programowania Java i Kotlin wykorzystywanych do tworzenia aplikacji mobilnych przeznaczonych do pracy pod kontrolą systemu Android. Na podstawie wyników badań można stwierdzić, że wybór języka programowania pomiędzy Javą a Kotlinem nie wpływa na obciążenie procesora. Aplikacje napisane w Kotlinie mogą jednak zużywać więcej pamięci. W przypadku prostych aplikacji, różnica powinna być niezauważalna. Może się ona jednak znacznie pogłębiać przy rozbudowanych projektach. Nie udało się sformułować jednoznacznych wniosków dotyczących czasów kompilacji i wykonania. Telefony marki Xiaomi osiągają lepsze czasy dla aplikacji napisanych w Javie. Natomiast w przypadku LG K350N to Kotlin pozwala szybciej kompilować i wykonywać programy.

Ponadto, Java zdaje się być lepszym wyborem dla poczatkujących programistów ze względu na większy zasób materiałów pomocniczych oraz większe wsparcie społeczności. Cechuje się też ona łatwiejszym do zrozumienia kodem. Kotlin jest natomiast językiem stosunkowo młodym, skierowanym do osób które mają już doświadczenie w programowaniu aplikacji mobilnych. Twórcy postanowili znacząco zredukować rozmiar kodu. Jest on również mniej intuicyjny, a część jego funkcjonalności pozostaje niewidoczna nawet dla programisty. Dlatego też będzie on lepszym wyborem dla doświadczonych osób poszukujących nowych, lepszych rozwiązań. Niemniej jednak ostatecznie, wybór powinien zależeć od indywidualnych preferencji programisty.

Języki Java i Kotlin są interoperacyjne, dlatego też nic nie stoi na przeszkodzie, aby dowolnie je mieszać w jednym projekcie. Mogą bez przeszkód korzystać z tych samych bibliotek i łączyć się z tymi samymi bazami danych. Z tego powodu następuje duża migrację projektów z Javy do Kotlina. Jako nowa alternatywa, Kotlin dostarcza wiele ciekawych i wygodnych rozwiązań, a jednocześnie pozostaje bardzo podobny do Javy. Można zatem się spodziewać, że w przyszłości zastąpi jej miejsce i stanie się najważniejszym językiem używanym do tworzenia aplikacji mobilnych. Trudno jest jednak oszacować, kiedy to nastąpi.

Literatura

[1] T. McDonnell, B. Ray, M. Kim: *An Empirical Study of API Stability and Adoption in the Android Ecosystem*, Texas 2013.

[2] Oficjalna dokumentacja języka Kotlin, https://kotlinlang.org/docs/reference/, Sierpień 2019

[3] P. Schwermer, *Performance Evaluation of Kotlin and Java on Android Runtime*, Sztokholm, Maj 2018.

[4] S. Bose, M. Mukherjee, A. Kundu i M. Banerjee, *A comparative Study: Java vs Kotlin Programming in Android Application Development*, International Journal of Advanced Research in Computer Science, Tom 9, Numer 3, Czerwiec 2018.

[5] M. Flauzino i inni: *Are you still smelling it?: A comparative study between Java and Kotlin language*, XII Sympozjum Brazylijskie dotyczące komponentów oprogramowania, architektury i ponownego użycia, s. 23-32, São Carlos, Wrzesień 2018.

[6] Oficjalna dokumentacja języka Java, https://docs.oracle.com/javase/8/docs, Kwiecień 2019.

[7] I. Kucherenko, A. Khan, *Hands-On Object-Oriented Programming with Kotlin*, Packt Publishing, Październik 2018.

[8] M. Devcic, *Kotlin Quick Start Guide: Core Features to Get You Ready for Developing Applications*, Packt Publishing, Sierpień 2018.

[9] K. Raghavendra Rao, *Kotlin for Enterprise Applications using Java EE: Develop, test, and troubleshoot enterprise applications and microservices with Kotlin and Java EE*, Packt Publishing, Listopad 2018.

[10] Oficjalna dokumentacja języka Android, https://developer.android.com/docs/, Kwiecień 2019.