

KOTLIN vs JAVA

Kotlin



Java



Paulina M. Powiła

JAVA

Jest to język programowania obiektowego (OOP), zapoczątkowany w 1995 roku. Zaprojektowany przez J. Goslinga, a jej główną implementacją był OpenJDK. Opracowany w mikrosystemach Sun, a następnie przejęty przez Oracle. Jest to język najczęściej używany do samodzielnych aplikacji lub programowania back-endowego. Java jest podstawowym wyborem w przypadku tworzenia aplikacji na Androida, ponieważ sam Android opiera się na Javie.

- > Łatwy w zrozumieniu
- > Może działać na maszynie wirtualnej lub w oknie przeglądarki
- > Dobre rozwiązanie dla aplikacji wieloplatformowych
- > Możliwe uzupełnienie w nowe biblioteki dzięki Android SDK



- > Jego ograniczenia mogą powodować błędy w projekcie interfejsu Androida
- > Większa ilość linijek kodu zwiększa prawdopodobieństwo wystąpienia błędów
- > Wolniejsza i wymagająca dużo pamięci w porównaniu z innymi językami

Wybierz gdy:

- > Wydajność ma znaczenie (mniej kodu, wysoka jakość)
- > Dążysz do rozwoju Androida
- > Bezpieczeństwo jako jeden z filarów twoich projektów
- > Projekty o mało skomplikowanej architekturze

KOTLIN

Kotlin jest dosyć nowym językiem (po raz pierwszy zaprezentowany w 2011 roku, a oficjalnie wydany w 2016 roku) opracowanym przez IDE Jet Brains. Jest to język open source'owym. Oparty na JVM (Java Virtual Machine), z możliwością kompilacji do JavaScript, Android i Native. Jest to język w pełni kompatybilny z istniejącymi pakietami Java, dlatego też przejście z jednego języka jest dosyć proste (starczy odpowiednia wtyczka).

- > Wiele funkcji do tworzenia niezawodnych interfejsów API
- > Bardziej zwężły w porównaniu z Javą
- > Łatwe przeniesienie się z języka Java na Kotlin
- > Efektywne kodowanie z pomocą IDE

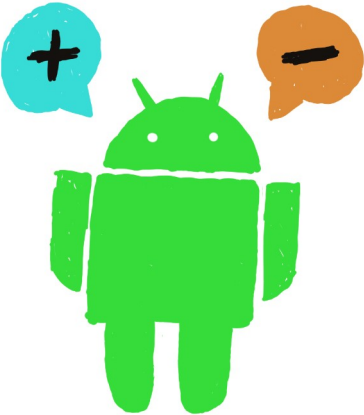




- > Kompilacja nieco wolniejsza niż w Javie
- > Aplikacje nie są kompaktowe
- > Niewielka ilość profesjonalnych programistów

Wybierz gdy:

- > Masz profesjonalnych programistów Java
- > Głównym celem jest tworzenie większych / złożonych projektów
- > Ograniczony budżet

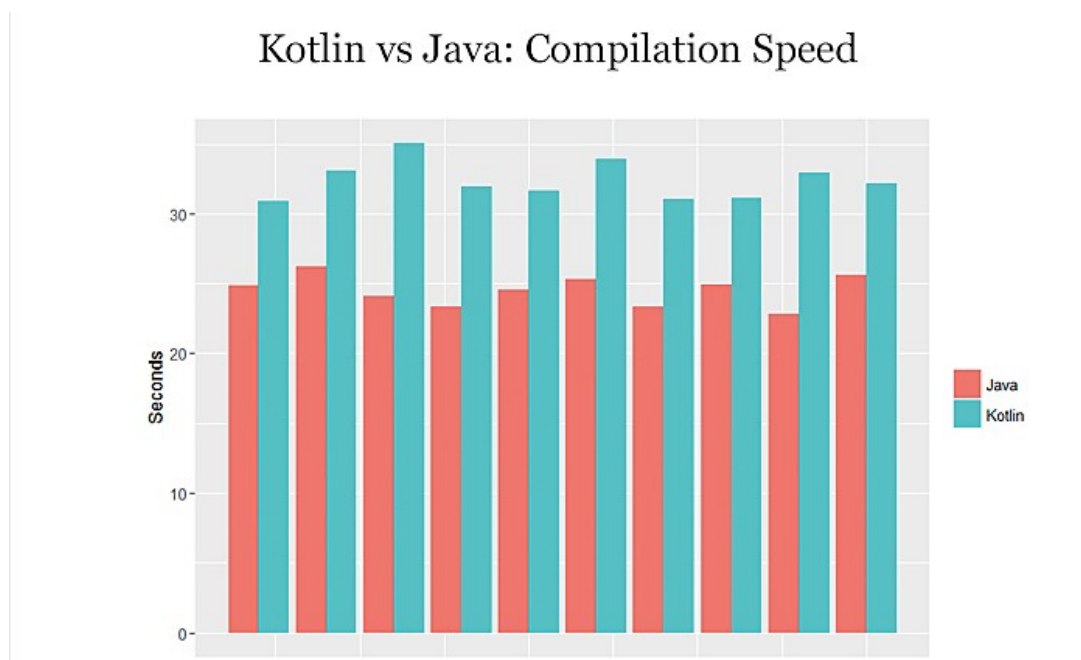
PORÓWNANIE

		
<p>Null Safe</p>	<p>Użytkownik może przypisywać wartości null (zerowe) do dowolnych zmiennych, lecz podczas uzyskiwania dostępu do obiektu o tej wartości, wyrzucany jest wyjątek wskaźnika zerowego, który należy obsłużyć.</p>	<p>Nie mam możliwości przypisania wartości zerowej do zmiennych/ obiektów. Jeśli spróbujemy przypisać i zwrócić taką wartość, kompilacja nie dojdzie do skutku. Deklaracja wartości zerowej jest możliwa tylko poprzez bezpośrednie jej zadeklarowanie.</p>
<p>Funkcje rozszerzeń</p>	<p>Rozszerzenie funkcjonalności klasy poprzez utworzenie nowej klasy i dziedziczenie z klasy nadrzędnej. Funkcje rozszerzeń niedostępne.</p>	<p>Możliwe rozszerzanie istniejących klas o nową funkcjonalność. Tworzenie funkcji rozszerzeń poprzez umieszczenie przed nazwą klasy nazwy nowej funkcji.</p>
<p>Wsparcie współprogramów</p>	<p>Inicjacja długotrwałych operacji sieciowych lub intensywnych operacji CPU, powoduje blokowanie odpowiednich wątków. Java umożliwia tworzenie i uruchamianie wielu wątków lecz panowanie nad nimi to skomplikowany proces.</p>	<p>W Kotlin możemy utworzyć wiele wątków, aby uruchomić te długotrwałe intensywne operacje, ale mamy obsługę coroutines, która zawiesi ich wykonywanie w pewnym momencie bez blokady wątków.</p>
<p>Obsługa wyjątków</p>	<p>Konieczność deklaracji Try catch, nawet gdy nie potrzebujemy wychwytywać danego wyjątku.</p>	<p>Jest możliwość wychwytywania wyjątków, ale nie musimy tego robić, bo przy braku deklaracji mamy krótszy kod, a odpowiedni błąd i tak zostanie zwrócony.</p>
<p>Deklaracja typu zmiennej</p>	<p>Konieczność wyraźnego określenia typu zmiennej.</p>	<p>Nie trzeba określać, ale możemy to zrobić.</p>

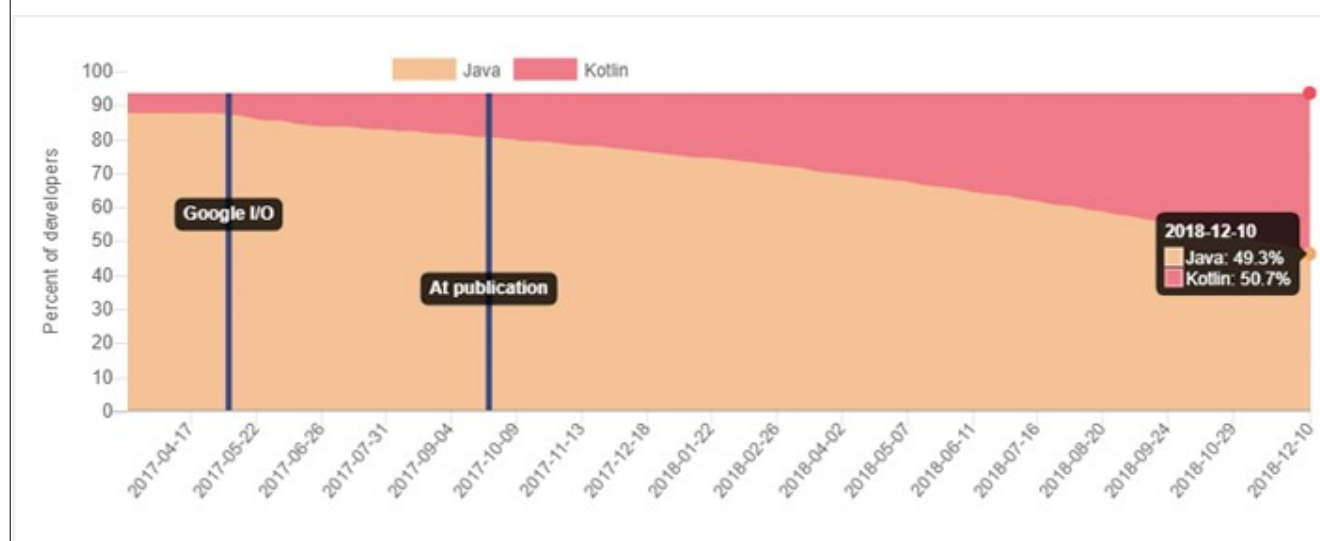
Klasy danych	Aby stworzyć klasę do przechowywania danych, musimy zdefiniować konstruktory, zmienne do przechowywania danych, metody pobierające i ustawiające oraz funkcje.	W przypadku tego typu klas można zadeklarować klasę z odpowiednim słowem kluczowym, a resztą zajmie się kompilator.
Inteligentne rzutowanie	Konieczność sprawdzenia zmiennych i rzutowania zgodnie z przeprowadzaną operacją.	Obsługa inteligentnego sprawdzania dzięki słowom kluczowym.
Programowanie Funkcjonalne	Nie obsługiwane[aż do Javy 8, w której wprowadzono lambdy.	Jest to język który składa się z wielu przydatnych metod takich jak lambda, przeciążenie operatora, funkcja wyższego rzędu itd.
Jakość kodu	Brak możliwości skrócenia kodu do długości kodu w Kotlinie.	Zwięzły i funkcjonalny kod, dzięki wielu dodatkowym funkcją

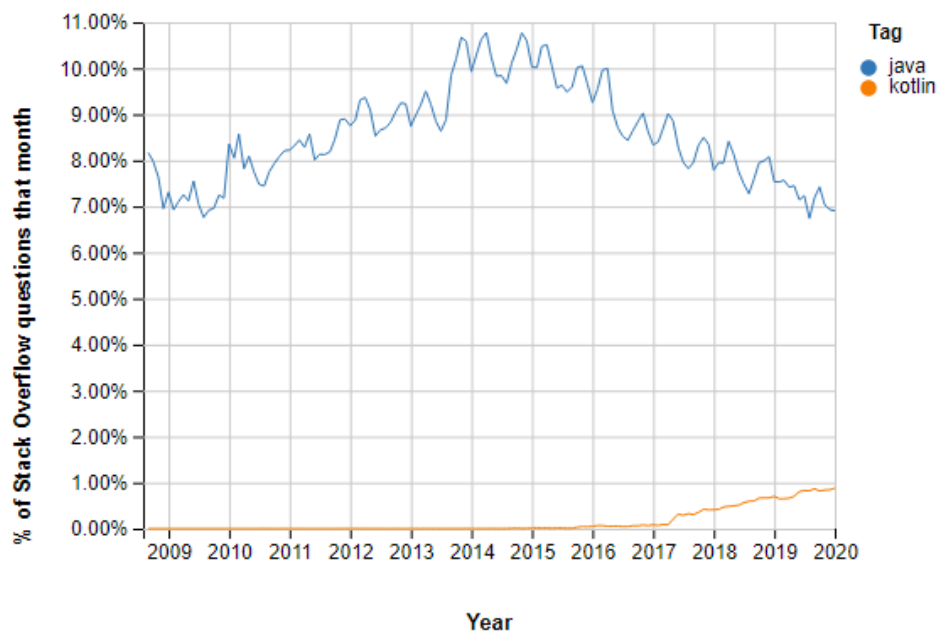
Java	Kotlin
<pre> class Person { private String name; public Person(String name) { this.name = name; } public String getName() { return name; } public void setName(String name) { this.name = name; } // toString... // hashCode... // equals... // copy... } </pre>	<pre> data class Person(val name: String) </pre>
Java	Kotlin
<pre> public void createAndPrintPerson() { String name = "Pieter"; Person person = new Person(name); printName(person.getName()); // Prints: Pieter Otten } </pre>	<pre> fun createAndPrintPerson() { val name = "Pieter" val person = Person(name) printName(person.name) // Prints: Pieter Otten } </pre>

Wydajność Aplikacji	Kompilacja szybsza niż w przypadku Kotlina.	Lepsza kompilacja przyrostowa oraz czystszy kod.
----------------------------	---	--



Wsparcie Android Studio	Najnowsza wersja Java nie jest obsługiwana przez Android Studio.	Całkowicie kompatybilny, ze względu na współzależności z IntelliJ IDEA.
Popularność	Wysoce ulokowana na rynku światowym. W 20'tce najpopularniejszych języków programowania świata.	Zdobywa na popularności, lecz nie znalazł na ten moment jeszcze miejsca w TOP 20'tce.





Stack Overflow Questions Java vs. Kotlin

Wsparcie dla złożonej architektury	Idealne do tworzenia złożonych aplikacji.	Mniej przyjazne dla rozbudowanych aplikacji.
Typy pierwotne	Pierwotne typy danych nie są obiektami tworzonymi przez instancję z klasy / struktury.	Pierwotne typy są obiektami.
Wsparcie dla konstruktorów	Tylko konstruktor podstawowy.	Może mieć jeden lub więcej konstruktorów dodatkowych, oprócz konstruktora podstawowego.











Operator trójskładnikowy	Posiada. Działa on jak podstawowa instrukcja if. Składa się z warunku, który daje wartość prawda lub fałsz.	Nie obsługuje.
Nieprywatne pola	Występują.	Nie występują.

KTÓRY LEPSZY?

Biorąc pod uwagę wszystkie pułapki związane z Javą (m.in. NullPointerException), Kotlin wydaje się lepszym wyborem. Chociaż Java jest nadal niezbędnym językiem, a na świecie jest od groma specjalistów z nim związanych, to przyszłość Androida idzie w stronę Kotlin. Nie zapominajmy jednak że nie tylko na Androidzie świat się opiera,, więc sama Java nie zostanie „wyparta” w 100% przez Kotlin, lecz podzieli się z nim „zastosowaniami”.

KRÓTKIE PODSUMOWANIE

Features	Java	Kotlin
Fully OOP(Object-Oriented Programming)	Not pure OOP	Fully OOP
Null Safety	No	Yes
Checked Exception	Yes	No
Invariant Array	No	Yes
Smart Casts	No	Yes
Singletons Object	Yes	Easily create Singleton objects
Functional Reactive Programming	No	Yes

 Attributes	 Java	 Kotlin
 App Performance	High	Super High
 Android Studio 3.0 Support	Partial	Excellent
 Code Quality	Not-Optimized	Excellent
 Market Presence	Excellent	Good
 Adoption Cost	High	Low
 App Security	Good	Excellent
 Support for Complex Architecture	Excellent	Not Good

ŹRÓDŁA

- [1] <https://www.educba.com/java-vs-kotlin/>
- [2] <https://www.promptbytes.com/blog/java-vs-kotlin-the-no-nonsense-comparison-of-android-programming-languages>
- [3] <https://hackr.io/blog/kotlin-vs-java>
- [4] <https://www.guru99.com/kotlin-vs-java-difference.html>
- [5] <https://www.apptunix.com/blog/kotlin-vs-java/>
- [6] <https://www.mobileappdaily.com/kotlin-vs-java>