JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 33 (2024) 285–291
Received: 17 June 2024
Accepted: 10 July 2024

Performance comparison of the Java and Kotlin programming languages based on an auto-scroller mobile game

Porównanie wydajności języków programowania Java i Kotlin na przykładzie gry mobilnej typu auto-scroller

Piotr Gajek*, Małgorzata Plechawska-Wójcik

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
The article contains a performance comparison of two mobile applications, developed in the Java and Kotlin programming languages respectively. The applications took the form of an auto-scroller game and shared their graphics assets. The aim of the work was to confirm or reject the hypothesis: Java is slightly more efficient than Kotlin in the presented context. The subject of comparison was the consumption of CPU, RAM and the device's battery resources. The mentioned metrics were obtained using the Android Profiler tool. The study was conducted using two mobile devices. The results confirm the research hypothesis.

Keywords: mobile application; performance comparison; Java; Kotlin

Streszczenie
Artykuł zawiera porównanie wydajności dwóch aplikacji mobilnych, napisanych odpowiednio w językach programowania Java i Kotlin. Aplikacje miały postać gry typu auto-scroller i posiadały wspólne zasoby graficzne. Celem pracy było potwierdzenie lub odrzucenie hipotezy: język Java jest w niniejszym kontekście nieznacznie lepiej wydajny niż język Kotlin. Przedmiotem porównania było zużycie pamięci RAM, CPU oraz baterii urządzenia. Metryki te uzyskano z użyciem narzędzia Android Profiler. Badanie przeprowadzono z użyciem dwóch urządzeń mobilnych. Wyniki wskazują na potwierdzenie hipotezy badawczej.

Słowa kluczowe: aplikacja mobilna; porównanie wydajności; Java; Kotlin

*Corresponding author
Email address: piotr.gajek@pollub.edu.pl (P. Gajek)

1. Wstęp

Przez długi czas jedynym oficjalnym językiem programowania umożliwiającym wytwarzanie aplikacji mobilnych na urządzenia z systemem operacyjnym Android był język Java. Sytuacja zmieniła się w 2017 roku, kiedy do oficjalnego środowiska programistycznego tego systemu Android Studio firma Google dodała nowy język programowania o nazwie Kotlin. Został on napisany przez firmę JetBrains jako alternatywa dla Javy, zgodna z jej kodem bajtowym i równie szybka w kompilacji, ale bardziej nowoczesna, umożliwiająca zmniejszenie długości kodu programów oraz łatwiejszą obsługę błędów. Do momentu wyboru, przez firmę Google, języka Kotlin jako preferowanego do programowania aplikacji mobilnych na system Android w 2019 roku, wśród deweloperów takich aplikacji trwała debata mająca na celu ustalenie, który z tych dwóch języków lepiej nadaje się do ich pracy. Nawet po jej zakończeniu nie znaleziono jednoznacznej odpowiedzi na pytanie, który z nich jest bardziej wydajny w działaniu.

Powstało wiele artykułów naukowych, których autorzy chcieli zabrać głos w tej dyskusji. Jednym z nich jest „A Comparative Study: Java vs Kotlin Programming in Android Application Development", opublikowany w 2018 roku [1]. Jest to artykuł przeglądowy, zawierający zbiór informacji z materiałów wideo, blogów, dokumentacji języków oraz artykułów napisanych przez innych naukowców. Podaje on wady i zalety każdego z języków w oparciu o zebrane informacje. Autorzy sugerują, iż język Java jest bardziej przyjazny dla początkujących programistów, natomiast Kotlin dla bardziej doświadczonych. Wraz z wprowadzaniem do niego kolejnych aktualizacji i poprawek zaczęły na ten temat powstawać dedykowane artykuły naukowe, takie jak praca polskich naukowców K. Wasilewskiego i W. Zabierowskiego napisana dla czasopisma Sensors [2], w której oprócz języków Java i Kotlin porównano również język programowania Flutter. Badania zostały przeprowadzone z użyciem technik bazujących na pracach naukowców porównujących wydajność aplikacji mobilnych i internetowych [3]. Polegały one na wykonaniu dużej liczby operacji na kolekcjach, zapytań bazodanowych oraz operacji odczytu/zapisu. Wyniki wykazały mniejsze zużycie pamięci RAM oraz bardziej spójny czas wykonywania instrukcji dla języka Java oraz mniejsze zużycie CPU dla języka Kotlin, przy czym autorzy ostrzegli, że wyniki mogą stracić aktualność w niedługim czasie, ponieważ język Kotlin był dalej rozwijany i optymalizowany. Potwierdziły one także istnienie anomalii w kodzie języka Kotlin [4]. Innym artykułem o tematyce porównania wydajności języków Java i Kotlin jest „Java and Kotlin, a Performance Compari-

son" [5], w którym metryką pomiaru był czas wykonywania iteracji wybranych algorytmów zaproponowanych w [6]. Charakterystyczną cechą użytych algorytmów była możliwość jednoznacznego przetłumaczenia ich kodu z języka Java na Kotlin. Badanie przeprowadzono na komputerze stacjonarnym, na którym wyniki wykazały podobny czas wykonania algorytmów w obu językach programowania, oraz na urządzeniu mobilnym, na którym większość implementacji algorytmów napisanych w języku Java okazała się szybsza. Dzieje się tak, ponieważ obciążenie CPU ma duży wpływ na szybkość działania programów napisanych w językach Java i Kotlin, co pokazano podczas konferencji SPLASH w 2017 roku [7]. Dlatego w roku 2020 powstał także artykuł opisujący stworzenie algorytmu umożliwiającego porównanie wydajności języków programowania Java, Kotlin i Dart na dowolnym urządzeniu [8]. Aplikacja stworzona na bazie tego algorytmu uruchamia na urządzeniu trzy programy kontrolowane przez skrypt napisany w języku komend Bash, który jednocześnie mierzy czas ich wykonania. Programy zawierają implementacje siedmiu algorytmów o różnych ilościach iteracji, przy czym każdy z nich jest napisany w innym z porównywanych języków programowania. Poprawność pomiarów została potwierdzona poprzez porównanie wyników z wynikami poprzedniej pracy jednego z autorów o podobnej tematyce [9]. Wyniki wykazują bardzo zbliżoną wydajność dla języków Java i Kotlin, a także mniejsze zużycie CPU dla języka Dart kosztem znacznego wydłużenia czasu wykonywania algorytmów.

2. Cel i zakres badań

Celem badań jest porównanie wydajności języków programowania Java i Kotlin w kontekście gry mobilnej. Praca ma za zadanie potwierdzić lub obalić hipotezę badawczą: język Java jest w niniejszym kontekście nieznacznie lepiej wydajny niż język Kotlin. W tym celu zostały utworzone dwie aplikacje mobilne, jedna w języku Kotlin a druga w języku Java. Obie aplikacje używają tych samych zasobów graficznych, takiej samej logiki kodu oraz posiadają te same funkcjonalności. Aplikacje mają postać gry typu auto-scroller działającej na urządzeniach z systemem operacyjnym Android. Ponieważ analiza literatury wykazała znaczne różnice w wydajności aplikacji pisanych w tych językach w zależności od urządzenia, na którym są uruchomione, badanie przeprowadzono z użyciem dwóch różnych urządzeń. Metryki użyte do porównania wydajności aplikacji to zużycie pamięci RAM, CPU oraz baterii urządzenia.

3. Materiał i metody

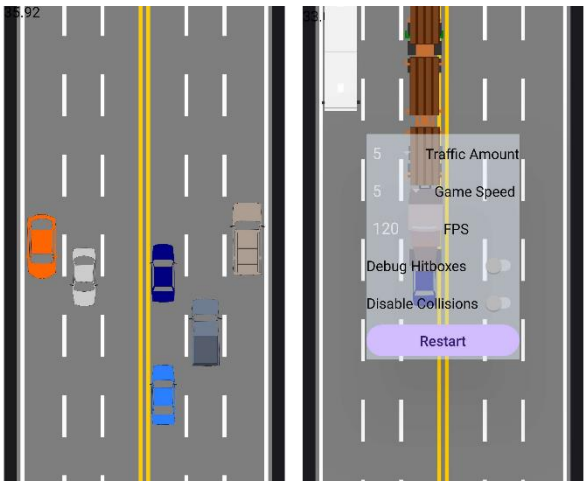
3.1. Narzędzia

Do wytworzenia aplikacji mobilnych oraz pomiaru ich wydajności wykorzystane zostały następujące narzędzia:

- Edytor grafiki wektorowej Inkscape – darmowy, otwartoźródłowy program [10], użyty w celu utworzenia zasobów graficznych zoptymalizowanych dla systemu Android używanych przez obie aplikacje.
- Zintegrowane środowisko programistyczne Android Studio – oficjalne środowisko oferowane przez firmę Google [11] dla rozwoju aplikacji mobilnych dedykowanych na urządzenia z systemem operacyjnym Android, użyte do napisania kodu aplikacji w językach programowania Java i Kotlin oraz ustalenia układu elementów na ekranie.
- Zestaw narzędzi pomiaru wydajności aplikacji Android Profiler [12] – grupa narzędzi wbudowana w środowisko Android Studio, zawierająca między innymi narzędzie Memory Profiler do pomiaru zużycia pamięci RAM urządzenia przez aplikację oraz CPU Profiler mierzące procentowe zużycie przez aplikację dostępnego czasu procesora i liczbę wykorzystywanych wątków.

3.2. Utworzone aplikacje

Tematem gry utworzonej na potrzeby badań jest jazda samochodem po autostradzie. Zadaniem gracza jest unikanie innych pojazdów poprzez zmianę pasa ruchu. Kontrola pojazdu odbywa się przez dotykanie odpowiednio prawej i lewej części ekranu. Wynik gracza zależy od długości przejechanego dystansu. Gra kończy się przy zderzeniu z innym pojazdem. Obie wersje aplikacji używają tych samych zasobów graficznych oraz identycznych plików XML definiujących ich położenie na ekranie. Rysunek 1 zawiera zrzuty ekranu z działającej wersji aplikacji w wersji napisanej w języku programowania Java.



Rysunek 1: Zrzuty ekranu z gotowej aplikacji.

Jako pierwszy napisany został kod aplikacji w języku Java, a następnie powstał kod w języku Kotlin o tej samej logice. Aplikacja składa się z trzech klas. Klasa główna odpowiada za ekran główny, kontrolę pojazdu gracza oraz menu, a dwie klasy pomocnicze odpowiednio za generowanie pojazdów w ruchu drogowym i ich zachowanie. Warto zauważyć znaczne różnice w długości kodu klas pomocniczych w językach programowania Java i Kotlin, czego przykład przedstawiono na Listingu 1.

Listing 1: Różnice długości kodu napisanego w językach programowania Java (na górze) i Kotlin (na dole) w jednej z klas aplikacji

```java
public class Car extends CarModel {
    4 usages
    private ImageView image;
    4 usages
    private int speed, lane;
    no usages
    public Car(String imageName, float imageSize, float topHitBoxPos, float bottomHitBoxPos,
               ImageView image, int speed, int lane) {
        super(imageName, imageSize, topHitBoxPos, bottomHitBoxPos);
        this.image = image;
        this.speed = speed;
        this.lane = lane;
    }
    1 usage
    public Car(CarModel carModel, ImageView image, int speed, int lane) {
        super(carModel.getImageName(), carModel.getImageSize(), carModel.getTopHitBoxPos(),
                carModel.getBottomHitBoxPos());
        this.image = image;
        this.speed = speed;
        this.lane = lane;
    }
    1 usage
    public void setNewModel(CarModel model){
        this.setImageName(model.getImageName());
        this.setImageSize(model.getImageSize());
        this.setTopHitBoxPos(model.getTopHitBoxPos());
        this.setBottomHitBoxPos(model.getBottomHitBoxPos());
    }
    9 usages
    public ImageView getImage() { return image; }
    no usages
    public void setImage(ImageView image) { this.image = image; }
    2 usages
    public int getSpeed() { return speed; }
    1 usage
    public void setSpeed(int speed) { this.speed = speed; }
    2 usages
    public int getLane() {return lane;}
    1 usage
    public void setLane(int lane) {this.lane = lane;}
}
```

```kotlin
class Car(carModel: CarModel, var image: ImageView, var speed: Int, var lane: Int) : CarModel(
    carModel.imageName, carModel.imageSize, carModel.topHitBoxPos, carModel.bottomHitBoxPos
) {
    fun setNewModel(model: CarModel) {
        this.imageName = model.imageName
        this.imageSize = model.imageSize
        this.topHitBoxPos = model.topHitBoxPos
        this.bottomHitBoxPos = model.bottomHitBoxPos
    }
}
```

W celu przeprowadzenia różnorodnych scenariuszy badawczych, generujących różne poziomy obciążenia urządzeń pomiarowych, do gry dodano menu ustawień. Pojawia się ono po zakończeniu rozgrywki i wpływa na jej kolejną iterację. Zawiera ono następujące ustawienia:

- liczba pojazdów w ruchu drogowym – wpływa na trudność rozgrywki, a do celów badań pozwala na zmianę ilości elementów wyświetlanych na ekranie aplikacji;
- szybkość gry – wpływa na trudność rozgrywki oraz na szybkość zdobywania punktów przez gracza;
- szybkość odświeżania ekranu w klatkach na sekundę – wskazuje liczbę klatek wyświetlanych na ekranie w ciągu jednej sekundy;
- tryb pokazywania punktów wystąpienia kolizji – używany do celów naprawy błędów związanych z pozycjami kolizji pojazdów, wpływa także na liczbę elementów wyświetlanych na ekranie ustawianą do celów badań;
- tryb wyłączenia kolizji – ułatwi przeprowadzenie badań w przypadku, kiedy pozostałe ustawienia znacznie zwiększają trudność rozgrywki.

Na Listingu 2 przedstawiono fragment kodu w języku programowania Java odpowiedzialnego za obsługę menu aplikacji.

Listing 2: Fragment kodu w języku programowania Java odpowiedzialnego za menu aplikacji

```java
gameSpeedSetting.setAdapter(gameSpeedSettingAdapter);
gameSpeedSetting.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    1 usage
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        gameSpeed[0] = Integer.parseInt(parent.getItemAtPosition(position).toString());
    }
    1 usage
    @Override
    public void onNothingSelected(AdapterView<?> parent) {}
});

Spinner fpsSetting = findViewById(R.id.fpsSetting);
ArrayAdapter<CharSequence> fpsSettingAdapter = ArrayAdapter.createFromResource(
        context: this, R.array.fps_options_array,
        android.R.layout.simple_spinner_item
);
fpsSettingAdapter.setDropDownViewResource(
        android.R.layout.simple_spinner_dropdown_item
);
fpsSetting.setAdapter(fpsSettingAdapter);
fpsSetting.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    1 usage
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        fps[0] = Integer.parseInt(parent.getItemAtPosition(position).toString());
    }
    1 usage
    @Override
    public void onNothingSelected(AdapterView<?> parent) {}
});

Switch debugHitBoxesSwitch = findViewById(R.id.debugHitboxesSwitch);
debugHitBoxesSwitch.setOnCheckedChangeListener(
        (buttonView, isChecked) -> debugHitBoxes[0] = isChecked
);

Switch disableCollisionsSwitch = findViewById(R.id.disableCollisionsSwitch);
disableCollisionsSwitch.setOnCheckedChangeListener(
        (buttonView, isChecked) -> disableCollisions[0] = isChecked
);
```

3.3. Środowiska i scenariusze testowe

Parametry urządzeń użytych do pomiaru wydajności aplikacji zostały przedstawione w Tabeli 1.

Tabela 1: Parametry urządzeń pomiarowych

Nazwa urządzenia	Samsung Galaxy J7	Samsung Galaxy A3
Liczba rdzeni procesora	8	4
Szybkość takt. procesora	1,6 GHz	1,5 GHz
Pojemność pamięci RAM	3,0 GB	1,5 GB
Wersja systemu Android	9.0	7.0

W Tabeli 2 przedstawiono scenariusze badawcze użyte podczas pomiaru wydajności kodu. Każdy ze scenariuszy wykonywany był na każdym z urządzeń dwukrotnie przez 5 minut. W ciągu tego czasu wykonano po 20 pomiarów aktualnego zużycia pamięci RAM oraz czasu procesora dla każdego ze scenariuszy, rozłożonych równomiernie w czasie co 15 sekund.

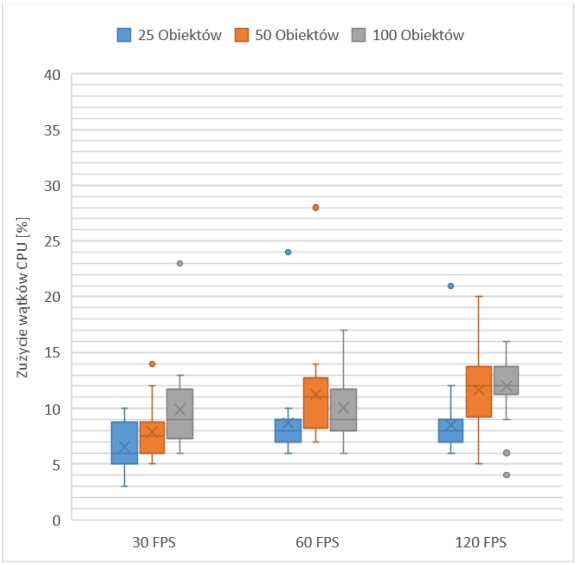
Tabela 2: Scenariusze badawcze

Numer scenariusza	Liczba klatek na sekundę	Liczba elementów graficznych na ekranie
1	30	25
2	60	25
3	120	25
4	30	50
5	60	50
6	120	50
7	30	100
8	60	100
9	120	100

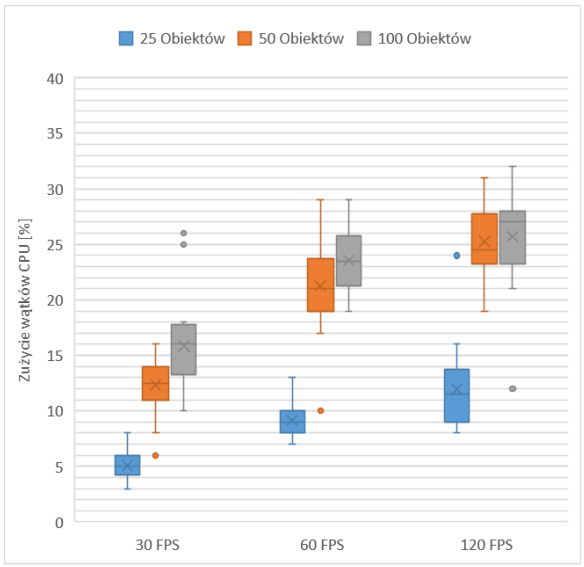
Osobno przeprowadzono badania zużycia energii przez aplikacje działające na najbardziej wymagającym zestawie ustawień. Polegały one na określeniu czasu zużycia 50% baterii przez każdą z aplikacji na każdym z urządzeń. Testy zostały przeprowadzone dwukrotnie, a wyniki uśredniono.

4. Wyniki

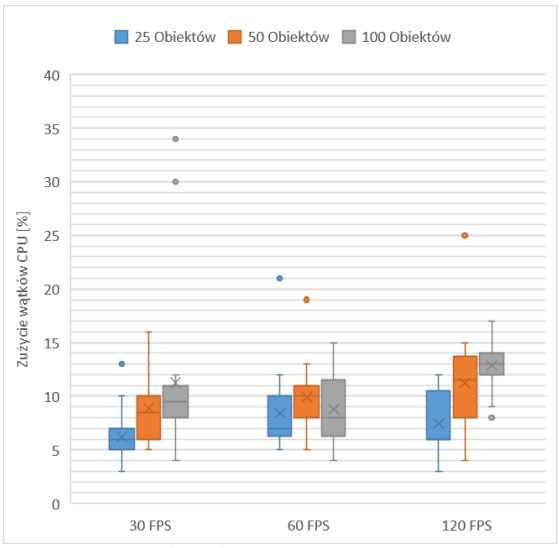
Wyniki badań w postaci wykresów typu „skrzynka i wąsy" przedstawiono na rysunkach 2-10. Rysunki 2-5 zawierają wykresy poziomów procentowych zużycia wątków CPU z podziałem na szybkość działania aplikacji w klatkach na sekundę oraz na liczbę obiektów wyświetlanych na ekranie. Osobno zaprezentowano wyniki dla każdego z urządzeń i każdej wersji aplikacji. Rysunki 6-9 to analogiczne przedstawienie zużycia pamięci RAM przez aplikacje. Na rysunku 10 przedstawiono wykres porównania zużycia baterii przez obie aplikacje na obu urządzeniach.



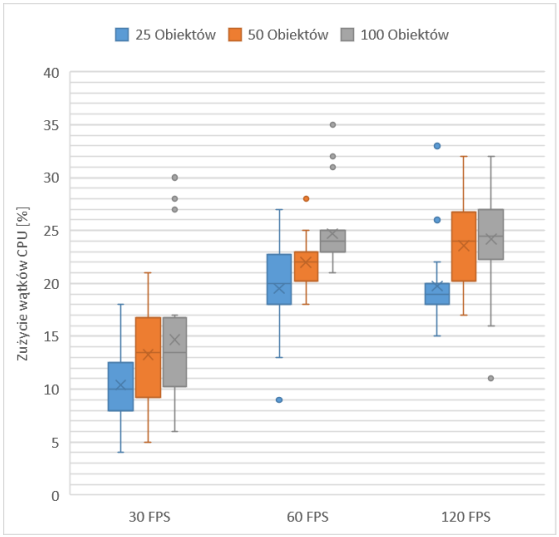
Rysunek 2: Zużycie wątków CPU w wersji aplikacji Java na urządzeniu Samsung Galaxy J7.



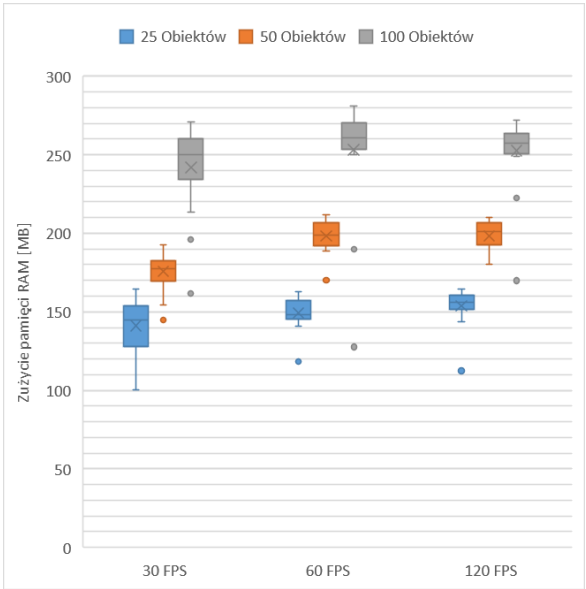
Rysunek 3: Zużycie wątków CPU w wersji aplikacji Java na urządzeniu Samsung Galaxy A3.



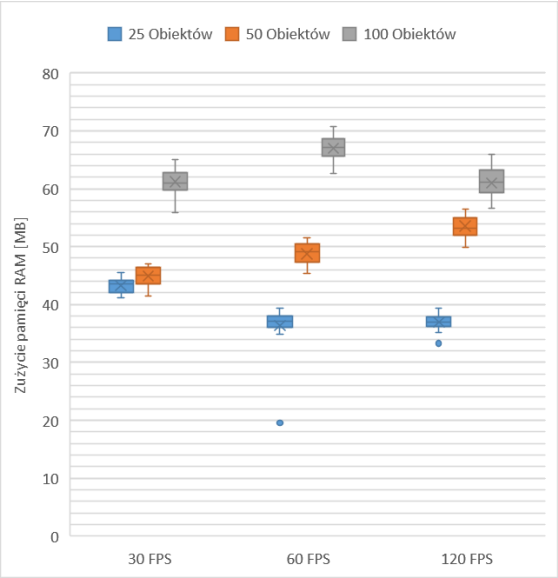
Rysunek 4: Zużycie wątków CPU w wersji aplikacji Kotlin na urządzeniu Samsung Galaxy J7.



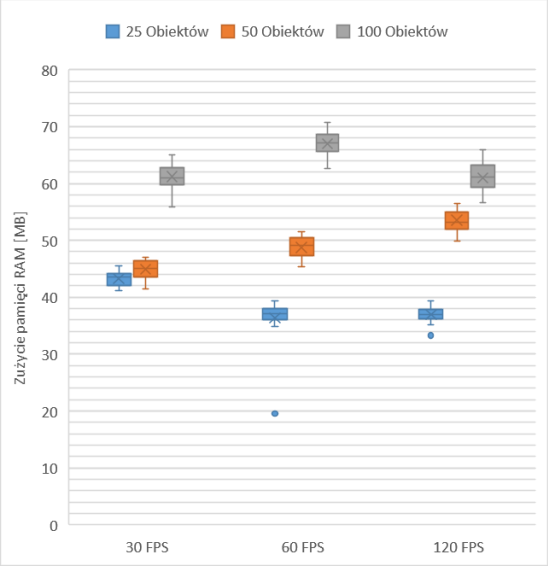
Rysunek 5: Zużycie wątków CPU w wersji aplikacji Kotlin na urządzeniu Samsung Galaxy A3.



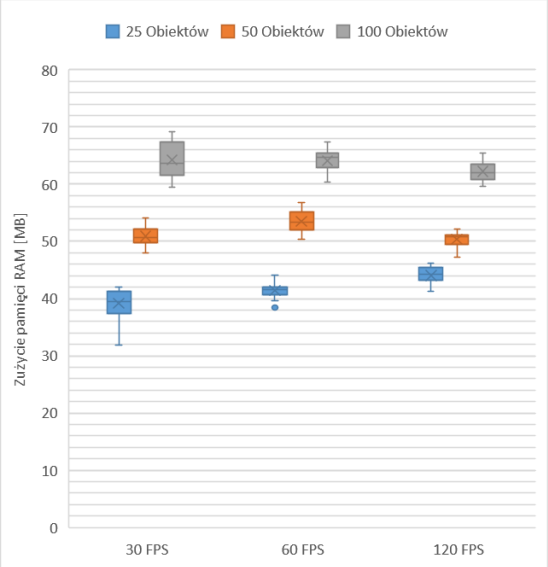
Rysunek 6: Zużycie pamięci RAM w wersji aplikacji Java na urządzeniu Samsung Galaxy J7.



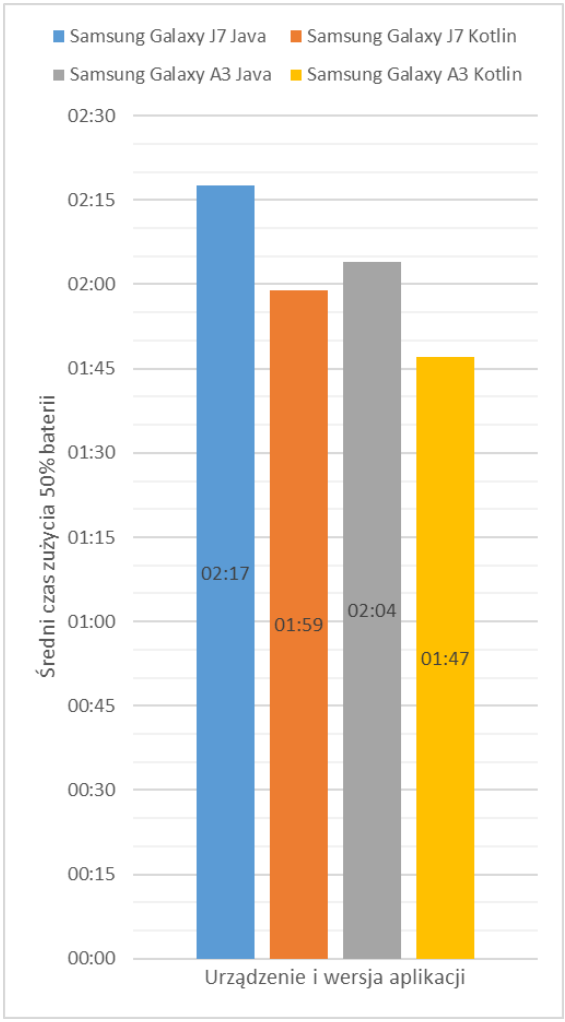
Rysunek 7: Zużycie pamięci RAM w wersji aplikacji Java na urządzeniu Samsung Galaxy A3



Rysunek 8: Zużycie pamięci RAM w wersji aplikacji Kotlin na urządzeniu Samsung Galaxy J7.



Rysunek 9: Zużycie pamięci RAM w wersji aplikacji Kotlin na urządzeniu Samsung Galaxy A3.



Rysunek 10: Porównanie średniego czasu zużycia 50% baterii przez obie wersje aplikacji na obu urządzeniach.

Walidacja wyników testów została przeprowadzona z użyciem testu t-Studenta z poziomem istotności 5%. W tabelach 3-6 przedstawiono wyniki testów statystycznych przeprowadzonych na scenariuszach testowych określonych w Tabeli 2. Tabele 3 i 4 odnoszą się do wyników testów przeprowadzonych na urządzeniu Samsung Galaxy J7, a Tabele 5 i 6 do wyników badań na urządzeniu Samsung Galaxy A3. Tabele 3 i 5 zawierają wyniki porównania zużycia wątków CPU, a tabele 4 i 6 porównanie zużycia pamięci RAM. Wiersze tabeli, dla których w obu próbach występuje wartość p-value mniejsza niż 5%, zostały pogrubione.

Tabela 3: Wyniki testów statystycznych dla porównania zużycia wątków CPU na urządzeniu Samsung Galaxy J7

Numer scenariusza	t-value próby 1	p-value próby 1	t-value próby 2	p-value próby 2
1	0,461373	0,325019	1,8348	0,041559
2	0,465961	0,323403	1,2955	0,105758
3	0,818765	0,211805	-0,0777	0,469462
4	-0,93491	0,181107	0,2296	0,410496
5	0,885001	0,193913	1,9911	0,030935
6	0,002507	0,499016	0,7508	0,231238
7	-1,06065	0,151448	1,6336	0,059856
8	0,872614	0,197186	-3,7821	0,000682
9	-0,88577	0,193703	-0,9182	0,185327

Tabela 4: Wyniki testów statystycznych dla porównania zużycia pamięci RAM na urządzeniu Samsung Galaxy J7

Numer scenariusza	t-value próby 1	p-value próby 1	t-value próby 2	p-value próby 2
1	0,4503	0,328935	-3,0867	0,003181
2	0,3981	0,347619	-1,0813	0,146919
3	3,3344	0,001845	-0,9796	0,170134
4	**-8,5922**	**<0,000001**	**-5,7277**	**<0,000001**
5	-0,6918	0,248945	-0,7395	0,234569
6	-0,7457	0,232738	-4,7455	0,000081
7	**-2,0288**	**0,028768**	**-1,8530**	**0,040175**
8	-1,4184	0,086578	-1,5435	0,070054
9	-2,5468	0,010117	-1,0086	0,163267

Tabela 5: Wyniki testów statystycznych dla porównania zużycia wątków CPU na urządzeniu Samsung Galaxy A3

Numer scenariusza	t-value próby 1	p-value próby 1	t-value próby 2	p-value próby 2
1	**-7,0216**	**<0,000001**	**2,0670**	**0,026713**
2	-10,9570	<0,000001	0,2278	0,411185
3	-6,5094	<0,000001	-0,0808	0,468246
4	-1,1844	0,125829	-0,7732	0,22472
5	-0,5311	0,300921	-0,3122	0,379238
6	1,5004	0,075424	0,1729	0,43233
7	1,0389	0,156303	-0,9886	0,167981
8	-1,4314	0,084725	-1,1645	0,129712
9	0,2361	0,40801	5,9878	<0,000001

Tabela 6: Wyniki testów statystycznych dla porównania zużycia pamięci RAM na urządzeniu Samsung Galaxy A3

Numer scenariusza	t-value próby 1	p-value próby 1	t-value próby 2	p-value próby 2
1	**6,1134**	**<0,000001**	**0,0293**	**<0,000001**
2	**-5,3400**	**0,000022**	**-12,5516**	**<0,000001**
3	**-14,1892**	**<0,000001**	**-3,1397**	**<0,000001**
4	**-11,0056**	**<0,000001**	**-11,5291**	**<0,000001**
5	**-9,2934**	**<0,000001**	**-2,0534**	**0,027429**
6	6,2563	<0,000001	0,0997	0,460842
7	**-3,9884**	**0,000431**	**-6,3009**	**<0,000001**
8	**3,8743**	**0,000556**	**-6,6555**	**<0,000001**
9	-1,5951	0,064049	-1,4622	0,080464

5. Dyskusja

Podczas analizy przedstawionych wyników dość szybko można zauważyć rozbieżności rezultatów względem urządzenia. Rezultat ten jest oczekiwany i zgodny z wnioskami badań przedstawionych dla konferencji SPLASH w 2017 roku [7]. Według nich obciążenie procesora urządzenia ma zauważalny wpływ na wydajność działającej na nim maszyny wirtualnej JVM, zatem dwa urządzenia o różnych parametrach CPU uzyskają różne poziomy wydajności dla aplikacji napisanych w językach Java i Kotlin.

Na urządzeniu Samsung Galaxy J7, różnice w zużyciu wątków CPU przez wersje aplikacji napisane w językach programowania Java i Kotlin były pomijalnie małe. Natomiast na wykresach przedstawiających porównanie zużycia wątków procesora dla urządzenia Samsung Galaxy A3 łatwo jest zauważyć wyraźne różnice w przypadku scenariuszy zawierających najmniejszą liczbę obiektów na ekranie podczas gry. Różnice te potwierdzają wyniki testów statystycznych zawartych w Tabeli 5. Wersja aplikacji napisana w języku Java wykazuje w tych przypadkach dużo niższy poziom zużycia CPU. Różnice te można wytłumaczyć faktem, iż podczas badań procesor urządzenia Galaxy A3 był mocno obciążony z powodu posiadania mniej wydajnych parametrów niż procesor urządzenia Galaxy J7. Można to zauważyć podczas porównania wykresów zużycia CPU dla tych urządzeń. W przypadku średniej i dużej liczby obiektów na ekranie, poziom obciążenia procesora wystarczył do obniżenia wydajności maszyny JVM dla obu wersji aplikacji, jednak dla małej liczby tych obiektów, wersja aplikacji napisana w języku programowania Java nie spowodowała wystarczająco dużego obciążenia procesora do zmniejszenia wydajności maszyny wirtualnej. Zatem w przypadku tego urządzenia, aplikacja napisana w języku Java była mniej obciążająca dla procesora.

Poziomy zużycia pamięci RAM były dużo bardziej zróżnicowane względem urządzeń niż poziomy zużycia wątków procesora. Urządzenie Samsung Galaxy A3 wykazało ogólnie dużo mniejsze zużycie pamięci RAM, co można wytłumaczyć istnieniem mechanizmu ograniczającego użycie tej pamięci przez aplikacje, wbudowanego w wersję systemu Android obecną na tym urządzeniu. Wykresy zużycia pamięci RAM mają więc różne skale dla tych urządzeń.

W przypadku urządzenia Samsung Galaxy J7, w scenariuszach o małej liczbie obiektów wyświetlanych na ekranie wersja aplikacji napisana w języku programowania Kotlin wykazała nieznacznie mniejsze zużycie pamięci RAM niż wersja napisana w języku Java. Największa różnica widoczna jest w przypadku scenariusza nr. 3, w którym aplikacja działała z szybkością odświeżania 120 klatek na sekundę. Scenariusz ten uzyskał bardzo wysoką wartość t w testach statystycznych. Natomiast dla scenariuszy o średniej i dużej liczbie obiektów, to wersja aplikacji napisana w języku Java wykazała mniejsze zużycie pamięci RAM. Największa różnica wystąpiła dla scenariusza 4, gdzie aplikacja działała z szybkością 30 klatek na sekundę. Wartość p-value dla tego scenariusza wyniosła mniej niż 0,000001. Duże różnice wystąpiły także w przypadku scenariuszy 7 i 9. Można zatem stwierdzić, że przy małym obciążeniu procesora wersja aplikacji napisana w języku Kotlin zużywa mniej pamięci RAM na tym urządzeniu, natomiast przy dużym obciążeniu zużywa ona więcej pamięci niż wersja napisana w języku Java.

Na urządzeniu Samsung Galaxy A3, poziomy zużycia pamięci RAM przez obie wersje aplikacji są trudne do porównania. W przypadku niektórych scenariuszy, wersja aplikacji napisana w języku Java radzi sobie lepiej niż wersja w języku Kotlin, w przypadku innych jest odwrotnie. Można to wyjaśnić istnieniem wspomnianej wyżej blokady zużycia pamięci RAM znajdującej się na tym urządzeniu. W przypadku wykrycia dużego zużycia tej pamięci przez aplikację, jej nieużywana część jest czyszczona. Poziomy wykrycia dużego zużycia pamięci wydają się być różne dla obu wersji aplikacji. Można jednak stwierdzić, że ogólnie średnie zużycie

pamięci RAM jest niższe w przypadku wersji aplikacji napisanej w języku Java, ale wzrasta wraz ze wzrostem obciążenia urządzenia bardziej stabilnie w przypadku wersji napisanej w języku Kotlin.

Porównując czasy zużycia 50% baterii przez obie wersje aplikacji na obu urządzeniach, widoczne na wykresie umieszczonym na rysunku 10, można jednoznacznie stwierdzić, że wersja aplikacji napisana w języku Kotlin zużywa baterię urządzenia szybciej niż wersja napisana w języku Java w przypadku tych urządzeń. Różnica jest większa dla urządzenia Samsung Galaxy J7.

6. Wnioski

Biorąc pod uwagę wszystkie zebrane do tej pory wnioski, można stwierdzić, iż wersja aplikacji napisana w języku programowania Java jest ogólnie nieznacznie bardziej wydajna niż wersja napisana w języku Kotlin. Obie wersje wykazują porównywalne obciążenie procesora na urządzeniu Galaxy J7, natomiast na Galaxy A3 wersja napisana w języku Java powoduje mniejsze obciążenie CPU. Na urządzeniu Galaxy J7, wykazuje ona mniejsze zużycie pamięci RAM dla większości scenariuszy, z wyjątkiem tych, w których występuje niskie zużycie czasu procesora. Natomiast na urządzeniu Galaxy A3 zużywa ona więcej pamięci RAM, jednak wzrost zużycia tej pamięci w stosunku do wzrostu obciążenia urządzenia jest bardziej stabilny. Wersja aplikacji w języku Java zużywa także mniej energii na obu urządzeniach. Hipotezę badawczą można zatem uznać za potwierdzoną.

Literatura

[1] S. Bose, A. Kundu, M. Mukherjee, M. Banerjee, A Comparative Study: Java vs Kotlin Programming in Android Application Development, International Journal of Advanced Research in Computer Science 9 (2018) 41-45, https://doi.org/10.26483/ijarcs.v9i3.5978.

[2] K. Wasilewski, W. Zabierowski, A Comparison of Java, Flutter and Kotlin/Native Technologies for Sensor Data-Driven Applications, Sensors 21 (2021) 3324 - 3340, https://doi.org/10.3390/s21103324.

[3] L. Corral, A. Sillitti, G. Succi, Mobile multiplatform development: An experiment for performance analysis, Procedia Computer Science 10 (2012) 736–743, http://doi.org/10.1016/j.procs.2012.06.094.

[4] T. Bryksin, V. Petukhov, K. Smirenko, N. Povarov, Detecting Anomalies in Kotlin Code, International Symposium on Software Testing and Analysis (2018) 10-12, http://doi.org/10.1145/3236454.3236457.

[5] N. Everlönn, S. Gakis, Java and Kotlin, a Performance Comparison, Bachelor thesis, Kristianstad University, 2020.

[6] W. H. Li, D. R. White, J. Singer, JVM-hosted Languages: They Talk the Talk, but Do They Walk the Walk?, In Principles and Practice of Programming in Java Conference (2013), 101-112.

[7] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, J. Saraiva, Energy efficiency across programming languages: how do energy, time, and memory relate?, In SPLASH: Systems, Programming, and Applications Conference (2017), 256–267.

[8] J. Espitia-Acero, M. Linares-Vásquez, Empirical Testing for Establishing Benchmarks: Process Review and Comparison Between Java, Kotlin and Dart's Performance, Bachelor thesis, Universidad de los Andes, 2020.

[9] J. A. E. Gonzalez, M. Linares-Vásquez, Comparative Performance Evaluation for Android Programming Languages, Bachelor thesis, Universidad de los Andes, 2019.

[10] Edytor grafiki wektorowej Inkscape, https://inkscape.org/.

[11] Oficjalne zintegrowane środowisko programistyczne Android Studio, https://developer.android.com/studio/.

[12] Dokumentacja narzędzi Android Profiler, https://developer.android.com/studio/profile.