JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 16 (2020) 305–308
Received: 30 June 2020
Accepted: 27 August 2020

Comparison of Objective-C and Swift on the example of a mobile game

Porównanie Objective-C oraz Swift na przykładzie gry mobilnej

Karolina Sylwia Banach*, Maria Skublewska-Paszkowska

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

Abstract

Mobile applications for the iOS platform can be developped using the Swift and Objective-C languages. The article presents a comparison between these languages based on a created mobile game. The structure and performance of these technologies were examined. Based on three devices, languages have been tested. Aspects such as RAM load, time between views, time to save data to the database and time to save data to file were tested as a part of the analysis. Two research hypotheses have been put forward: "Swift has a better performance than Objective-C" and "Swift has a simpler structure than Objective-C". The results obtained confirm that Swift is more efficient than Objective-C. Research into the structure of codes has proven that the newer language has a simpler structure than its predecessor.

Keywords: Swift; Objective-C; performance; structure

Streszczenie

Aplikacje mobilne na platformę iOS można wytwarzać z użyciem języków Swift oraz Objective-C. Tematyką artykułu jest porównanie tych języków na przykładzie utworzonej gry mobilnej. Zbadana została struktura i wydajność omawianych technologii. Na przykładzie trzech iPhone'ów, języki zostały poddane testom. W ramach przeprowadzonej analizy wydajnościowej zostały przebadane takie aspekty jak: obciążenie pamięci RAM, czas przejścia pomiędzy widokami, czas zapisu danych do bazy oraz czas zapisu danych do pliku. Zostały postawione dwie hipotezy badawcze: "Język Swift jest wydajniejszy niż język Objective-C" oraz "Język Swift posiada prostszą strukturę niż język Objective-C". Otrzymane wyniki potwierdzają, że Swift jest wydajniejszy niż Objective-C. Dzięki badaniom struktury kodów udowodniono, że nowszy język posiada prostszą strukturę niż jego poprzednik.

Słowa kluczowe: Swift; Objective-C; wydajność; struktura

*Corresponding author

Email address: Karolina.sylwia.banach@gmail.com

1. Wstęp

W dzisiejszych czasach większość społeczeństwa nie wyobraża sobie funkcjonowania bez smartfonów. Telefony używane są nie tylko w celach komunikacji, ale również do rozrywki, poszerzania wiedzy, a nawet w celach finansowych. Dzięki tym urządzeniom można sprawdzić rozkład jazdy komunikacji, stan swojego konta bankowego, dokonać płatności w sklepach internetowych oraz stacjonarnych lub skontaktować się ze znajomymi poprzez media społecznościowe. Technologie równie szybko muszą się rozwijać i dostarczać coraz więcej rozwiązań, aby nadążyć za postępem, dlatego z każdym rokiem odnawiane są stare języki programowania lub powstają nowe.

W czerwcu 2007 roku została wydana pierwsza wersja systemu iOS, przyjmujący wtedy nazwę iPhone OS. Do czerwca 2014 roku, język Objective-C był jedynym językiem wykorzystywanym do pisania aplikacji mobilnych przeznaczonych dla systemu iOS [1]. Głównym powodem twórców było stworzenie nowego języka programowania, który miał być zastępcą języków C, a zwłaszcza Objective-C. Nowa technologia miała być równie dobrze wydajna co jej poprzednik, ale miała być bardziej bezpieczna. Sam kod miał być bardziej współczesny. Celem jego było przyciągnięcie nowych programistów, którzy byli przyzwyczajeni do nowych technologii takich jak Java, C#, czy JavaScript [2].

Niniejszy artykuł ma na celu porównanie dwóch języków programowania: Swift i Objective-C na przykładzie gry mobilnej. W pierwszej kolejności porównano wydajność stworzonych aplikacji.Dokonano analizy następujących parametrów: obciążenie pamięci RAM, czas przejścia pomiędzy widokami, czas zapisu danych do bazy oraz czas zapisu danych do pliku. W dalszej części zbadano również struktury kodów napisanych w obu językach.

Na podstawie omawianego zagadnienia postawiono dwie hipotezy badawcze: "Język Swift jest wydajniejszy niż język Objective-C" oraz "Język Swift posiada prostszą składnię niż język Objective-C". Zostały one udowodnione na podstawie przeprowadzonych badań.

Istnieje niewiele publikacji naukowych dotyczących porównań języka Objective-C z językiem Swift. Większość z nich skupia się jedynie na porównaniu ich pod względem struktury, jak np. pozycja [3], w której autorzy udowadniają, że składnia Swift jest krótsza, prostsza oraz bardziej przypominająca języki współczesne. Kolejnym przykładem, jest pozycja [4], w której autorzy wyraźnie podkreślają różnice i podobieństwa w strukturze obydwu technologii. Istnieją dokumentacje badające same bezpieczeństwo języków [5]. W artykule [6], porównana jest wydajność Objective-C oraz Swift,

pod względem szybkości wykonywania algorytmów sortowania oraz struktur danych. Po analizie prac literackich, nie spotkano się z analizą, która poruszana jest w niniejszym artykule.

2. Metodyka badań

W celu przeprowadzenia badań zarówno wydajnościowych, jak i strukturalnych, napisano specjalne aplikacje mobilne, które działają identycznie w językach Swift oraz Objective-C. Gra ping-pong polega na odbijaniu piłeczki paletką poprzez ruch dotykowy na ekranie iPhone. Rozgrywka toczy się między graczem a iPhone.

Do utworzenia gry mobilnej ping-pong, potrzebnej do przeprowadzenia badań wykorzystano następujące narzędzia i technologię:

- środowisko programistyczne X-code 11.5 [7];
- system iOS 13.3;
- język programowania Swift 5.0;
- język programowania Objective-C 2.1;
- framework SpriteKit [8];
- system zarządzania bazą danych MySQL;
- format JSON;
- język programowania PHP 7.1.

Obciążenie pamięci RAM zostało zbadane podczas przeprowadzania rozgrywki gracza. Czas przejścia pomiędzy widokami, był pobrany podczas przejścia z widoku rejestracji do widoku logowania. Czas zapisu danych do zewnętrznej bazy został pobrany podczas zapisu wyniku rozgrywki. Ostatnia analiza została przebadana w momencie zapisu danych do pliku.

Badania wydajności przeprowadzono na trzech urządzeniach z systemem iOS:

- iPhone SE 2nd generation;
- iPhone X;
- iPhone 8.

3. Badania wydajności

Na podstawie wyników testów wyznaczono wartość minimalną, wartość maksymalną, średnią i medianę. Każda operacja została wykonana po 20 razy, na każdym z urządzeń oraz technologii.

3.1. Obciążenie pamięci RAM

W tabeli 1 zaprezentowano, jak przedstawia się obciążenie pamięci RAM na badanych smartfonach, uwzględniając omawiane technologie.

Na podstawie średniej i mediany można zauważyć, że obciążenie RAM nie zależy jedynie od danego języka. Widać, że im nowsze urządzenie tym Objective-C lepiej sobie radzi niż język Swift. W przypadku iPhone'a X obie technologie zużywają podobną ilość pamięci RAM.

Tabela 1: Uśrednione obciążenie pamięci RAM

Technologia	Urządzenie	Wartość minimalna [MB]	Wartość maksymalna [MB]	Średnia [MB]	Mediana [MB]
Swift	iPhone SE 2nd generation	108,05	142,77	125,91	127,36
	iPhone X	115,07	132,04	118,59	115,74
	iPhone 8	121,43	124,74	124,32	124,63
Objective-C	iPhone SE 2nd generation	98,61	106,64	105,53	106,37
	iPhone X	112,00	153,47	116,80	115,05
	iPhone 8	134,78	135,53	135,39	135,42

3.2. Czas przejścia pomiędzy widokami

W tabeli 2 przedstawiono uśrednione wyniki czasu przejścia pomiędzy widokami uwzględniając badane smartfony oraz języki programowania.

Tabela 2: Uśredniony czas przejścia pomiędzy widokami

Technologia	Urządzenie	Wartość minimalna [ms]	Wartość maksymalna [ms]	Średnia [ms]	Mediana [ms]
Swift	iPhone SE 2nd generation	4,8150	6,4710	5,9566	6,0075
	iPhone X	2,5610	6,8600	5,4074	5,4790
	iPhone 8	3,0310	6,8340	5,3835	5,4851
Objective-C	iPhone SE 2nd generation	4,2089	6,8970	5,9865	6,2345
	iPhone X	4,6780	8,6850	5,9301	5,9845
	iPhone 8	2,7441	7,6700	5,9412	6,0365

Na podstawie danych z 2 tabeli można stwierdzić, że aplikacja napisana w języku Swift znacznie szybciej przechodzi między widokami aplikacji. Wartości w urządzeniu iPhone SE, są do siebie o wiele bardziej zbliżone, niż w pozostałych smartfonach.

3.3. Czas zapisu danych do zewnętrznej bazy danych

W tabeli 3 przedstawiono wyniki statystyczne czasu zapisu danych do zewnętrznej bazy na badanych urządzeniach w językach Objective-C oraz Swift. Do łączności z bazą MySQL aplikacja wykorzystuje format JSON oraz pliki php.

Wyniki badań przeprowadzonych na iPhone'ach dowodzą, że Objective-C w szybszy sposób zapisuje dane do zewnętrznej bazy danych. Urządzenie nie ma wpływu na wyniki testu, ponieważ średnia wartość danej technologii, dla badanych modeli iPhoneó'w jest do siebie bardzo podobna.

Tabela 3: Uśredniony czas zapisu do zewnętrznej bazy danych

Technologia	Urządzenie	Wartość minimalna [ms]	Wartość maksymalna [ms]	Średnia [ms]	Mediana [ms]
Swift	iPhone SE 2nd generation	0,5180	0,6050	0,5441	0,5420
	iPhone X	0,4660	0,7080	0,5789	0,5710
	iPhone 8	0,4209	0,6930	0,5507	0,5400
Objective-C	iPhone SE 2nd generation	0,2650	0,5200	0,3143	0,3061
	iPhone X	0,2450	0,3549	0,3046	0,3026
	iPhone 8	0,1940	0,4870	0,3530	0,3515

3.4. Czas zapisu danych do pliku

W tabeli 4 przedstawiono uśrednione wyniki czasu zapisu danych do pliku uwzględniając badane iPhone oraz języki programowania.

Tabela 4: Uśredniony czas zapisu danych do pliku

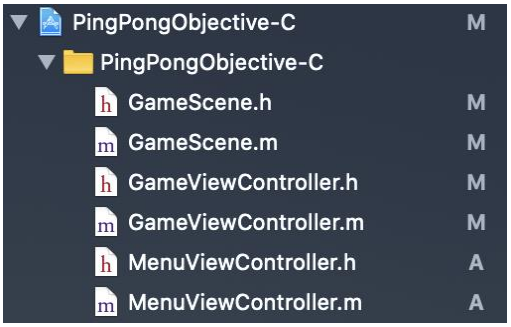
Technologia	Urządzenie	Wartość minimalna [ms]	Wartość maksymalna [ms]	Średnia [ms]	Mediana [ms]
Swift	iPhone SE 2nd generation	0,7019	1,5550	0,9293	0,8635
	iPhone X	0,6330	1,5190	0,8640	0,7970
	iPhone 8	0,4190	1,0331	0,8183	0,8135
Objective-C	iPhone SE 2nd generation	0,7960	1,7580	1,3254	1,3396
	iPhone X	0,6371	1,6630	1,2320	1,3075
	iPhone 8	0,8920	1,8040	1,2518	1,2925

Na podstawie średniej i mediany można zauważyć, że czas zapisu danych do pliku w języku Swift jest znacznie krótszy, niż w języku Objective-C. Wartości minimalne i maksymalne w obu technologiach są bardziej zbliżone do siebie, niż wartości średnie oraz mediany.

4. Analiza struktury

Nowa składnia języka Swift pozwala pisać aplikacje w zdecydowanie mniejszej liczbie plików, co można zauważyć na przykładach 1 oraz 2. Klasa języka Swift składa się z jednego pliku z rozszerzeniem ".swift" (rys 1) [9]. W przypadku Objective-C struktura jest 2 razy dłuższa, ponieważ każda klasa jest zbudowana z dwóch plików (przykład 2). Jeden z nich posiada rozszerzenie ".h"i jest on interfejsem klasy, a drugi z rozszerzeniem ".m" jest częścią implementacyjną [10].

Rysunek 1. Fragment struktury gry ping-pong w języku Swift.



Rysunek 2. Fragment struktury gry ping-pong w języku Objective-C.

Składnia omawianych języków programowania jest bardzo podobna do siebie, natomiast różnice występują w strukturze kodu. Język Swift ma zdecydowanie prostszą i bezpieczniejszą strukturę, co dowodzi brak konieczności stawiania średników na końcu linijki. Dodatkowo same odwołanie się do metod odbywa się podobnie, jak we współczesnych językach programowania. Przykładem tego jest obiekt touches, prezentowany w metodzie touchesBegan (listing 1). W przypadku technologii Swift wywołanie metody odbywa się poprzez dodanie do obiektu kropki a następnie nazwy metody, co widać na listingu 1. W Objective-C ta sama czynność odbywa się poprzez dodanie nawiasów kwadratowych przed obiektem, następnie oddzielenie spacją obiektu od metody wywołującej zamknięcie nawiasu oraz postawienie średnika po zakończonej czynności (Listing 2).

Listing 1. Metoda touchesBegan w języku Swift

```
override func touchesBegan(_ touches: Set<UITouch>,
    with event: UIEvent?) {

    for touch in touches{
        let location = touch.location(in: self)
        rocketPlayer.run(SKAction.moveTo(x:
            location.x, duration: 0.1))
    }
}
```

Listing 2. Metoda touchesBegan w języku Objective-C

```
- (void)touchesBegan:(NSSet *)touches
    withEvent:(UIEvent *)event {
    for (UITouch *touch in touches){
        CGPoint location = [touch
            locationInNode:(self)];
        [rocketPlayer runAction:[SKAction
            moveToX:location.x duration:0.1]];
    }
}
```

5. Wnioski

W niniejszym artykule przedstawiono analizę porównawczą języków programowania Swift oraz Objective-C na przykładzie stworzonej gry mobilnej ping-pong. Aplikacja przeszła testy pod względem wydajnościowym oraz strukturalnym.

Na podstawie otrzymanych wyników, można stwierdzić, że wybór technologii jest uzależniony od funkcjonalności przyszłej aplikacji. Język Objective-C w szybszy sposób przekazuje dane do zewnętrznej bazy MySQL. Technologia Swift natomiast szybciej przechodzi między widokami oraz znacznie szybciej zapisuje dane do plików. Język Swift jest w dalszym ciągu ulepszany i dopracowywany, być może w przyszłości poprawią czas pracy z formatem JSON, wtedy wybór będzie bezproblemowy. Język Swift jest wydajniejszy niż język Objective-C, ponieważ więcej testów przeszedł znacznie szybciej.

Struktura kodu Objective-C jest dużo bardziej złożona i precyzyjna, dużo łatwiej popełnić błąd podczas pisania kodu. W strukturze kodu widać dużą przewagę języka Swift nad jego poprzednikiem, jest bardziej intuicyjny oraz współczesny. Sam fakt, że do stworzenia klasy wystarczy tylko jeden plik daje mu ogromną przewagę nad językiem Objective-C. Dodatkowym atutem języka Swift jest fakt, że trudniej popełnić błąd podczas pisania kodu, ponieważ składnia została uproszczona do minimum, np. nie ma potrzeby stawiania średnika po zakończonym poleceniu.

Znacznie lepszym wyborem dla początkujących programistów przy tworzeniu gier dedykowanych na system iOS jest język Swift, ze względu na swoją strukturę, ponieważ jest ona prostsza niż w języku Objective-C.

Literatura

[1] The History of iOS, from Version 1.0 to 13.0, https://www.lifewire.com/ios-versions-4147730 [16.06.2020].

[2] About Swift, https://swift.org/about [16.06.2020].

[3] C. G. Garcia, J. P. Espada, B. C. Pelayo G-Bustelo, J. M. Cueva Lovelle, Swift vs. Objective-C: A New Programming Language,Regular Issue, https://www.researchgate.net/publication/277142254_Swift_vs_Objective-C_A_New_Programming_Language [30.06.2020].

[4] K.E. Sienkiewicz, E. Łukasik, Porównanie aplikacji mobilnej w językach Swift i Objective-C, Journal of Computer Sciences Institutes, http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-875445e3-044-6b5-b16d-880f12cc7ab3 [30.06.2020].

[5] V. M. Santana, P. Centoze, Seciurity mechanisms and analysis for insecure data storage and unintended data leakage for mobile applications, International Journal of Computer and Technology https://www.researchgate.net/publication/324985466_SECURITY_MECHANSMS_AND_ANALYSIS_FOR_INSECURE_DATA_STORAGE_AND_UNINTENDED_DATA_LEAKAGE_FOR_MOBILE_APPLICATIONS [30.06.2020].

[6] K. Gut, M. Skublewska-Paszkowska, E. Łukasik, J. Smołka, Comparison of programming languages on the iOS platform in terms of performance, IAPGOŚ http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-0bbfbb45-b7bf-4703-91aa-169a26d70236 [30.06.2020].

[7] Xcode, https://developer.apple.com/xcode/ [16.06.2020].

[8] SpriteKit, https://developer.apple.com/documentation/spritekit/ [16.06.2020].

[9] M. Lassoff, T. Stachowitz, Swift Fundamentals: The Language of iOS Development. 2014.

[10] D. Chisnall, Objective-C Phrasebook, Second Edition. 2012.