Dydaktyka Informatyki 16(2021)
ISSN 2083-3156; e-ISSN 2543-9847
DOI: 10.15584/di.2021.16.19
http://www.di.univ.rzeszow.pl

Uniwersytet Rzeszowski

Artur BARTOSZEWSKI

ORCID: 0000-0002-2648-7924. Dr, Uniwersytet Technologiczno-Humanistyczny w Radomiu, Wydział Transportu, Elektrotechniki i Informatyki, Katedra Informatyki, ul. Malczewskiego 22a, 26-600 Radom; e-mail: artur.bartoszewski@uthrad.pl

WSPÓŁCZESNE TECHNOLOGIE PROGRAMISTYCZNE NA PRZYKŁADZIE TWORZENIA APLIKACJI MOBILNEJ „KALKULATOR”

MODERN PROGRAMMING TECHNOLOGIES ON THE EXAMPLE OF MOBILE APPLICATION “CALCULATOR”

Słowa kluczowe: programowanie, aplikacje mobilne, Android.
Keywords: programming, mobile applications, Android.

Streszczenie

W pracy przedstawiono wybrane problemy dotyczące tworzenia aplikacji mobilnych dla systemu Android na przykładzie programu „kalkulator”. Zademonstrowano możliwości dydaktycznego zastosowania tego rodzaju projektów.

Abstract

The paper presents selected problems concerning the development of mobile applications for Android on the example of the calculator program. The author demonstrated the possibilities of didactic application of this type of projects.

Wstęp

Jednym z problemów w nauczaniu programowania jest płaska krzywa zdobywania kompetencji. W klasycznym podejściu uczeń zdobywa podstawową wiedzę o praktycznym zastosowaniu algorytmów niezbędną każdemu programi-

ście, jednak podawana jest ona w sposób niewiele odbiegający od tego jak programowania uczono 20 i więcej lat temu.

Przepaść pomiędzy działającymi w środowisku tekstowym programami pisanymi przez uczniów na lekcjach informatyki a użytkowymi aplikacjami, z którymi spotykają się oni na co dzień wydaje się być nie do pokonania.

Artykuł stanowi próbę zaprezentowania alternatywnego podejścia do edukacji programistycznej. Podejściem tym jest skonfrontowanie ucznia już na względnie wczesnym etapie nauki ze współcześnie używanymi narzędziami i technologiami programistycznymi. Zaprezentować je można w postaci spójnego projektu, w którego efekcie powstanie w pełni funkcjonalna aplikacja. Taki model nauki spotykany jest często w samokształceniu programistów. W kształceniu formalnym pojawia się on zwykle dopiero na wyższych semestrach studiów. Celowe może być jednak zapoznanie ucznia z nowoczesnymi technologiami już na znacznie wcześniejszym etapie kształcenia. Mowa tu o technikach i liceach profilowanych o specjalności informatyka.

Analiza przedstawionego poniżej przykładu pozwoli określić, jakich umiejętności i jakiej podbudowy teoretycznej będzie potrzebował uczeń przed przystąpieniem do tworzenia pełnoprawnej aplikacji.

Przykłady praktyczne wykorzystywane podczas kształcenia mogą być związane z różnymi dziedzinami i technologiami programistycznymi. Głównymi warunkami wyboru będą dostępność narzędzi ułatwiających pracę oraz możliwość uzyskania spektakularnych efektów, które zmotywują ucznia. Na potrzeby niniejszego artykułu wybrano tworzenie aplikacji mobilnych. Prostym, a jednocześnie ciekawym dla ucznia przykładem może być wykonanie aplikacji typu kalkulator.

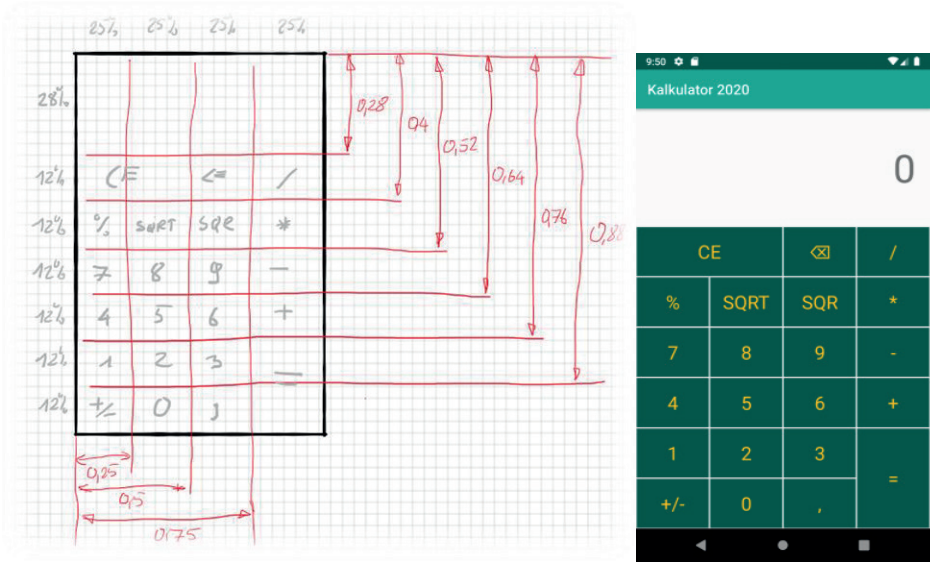
Projekt i graficzny interfejs użytkownika aplikacji

Narzędziem niezbędnym do wykonania aplikacji działającej w systemie Android jest darmowe, zintegrowane środowisko programistyczne Android Studio. Językiem programowania, którym należy się posłużyć są Java lub Kotlin. W omawianym przykładzie wykorzystany będzie język Java. Warto zaznaczyć, że do wykonania opisywanej aplikacji potrzebna jest tylko podstawowa znajomość najważniejszych technik programistycznych wspólnych dla większości powszechnie używanych języków programowania.

Pracę nad projektem rozpoczyna uczeń od określenia niezbędnych funkcjonalności aplikacji. Kolejnym etapem jest zaprojektowanie graficznego interfejsu użytkownika powstającej aplikacji, nazywanego też często jej layoutem. Układ layoutu oparty będzie na systemie linii pomocniczych. Ekran po-

dzielony zostanie na osiem wierszy, z czego dwa górne przeznaczone będą na wyświetlacz. Przyciski ułożone będą w czterech kolumnach. Potrzebnych jest więc sześć poziomych linii pomocniczych i trzy pionowe. Utworzą one siatkę, w której rozmieszczone zostaną wszystkie elementy interfejsu. Projekt layoutu oraz jego docelowy wygląd pokazano na rys. 1. Linie pomocnicze najprościej dodać przy użyciu narzędzia *Designer* będącego częścią środowiska *Android Studio*[1].

W siatce linii pomocniczych osadzone zostaną właściwe elementy interfejsu (w nomenklaturze API Androida nazywane widokami[2]). Widok typu *EditText* będzie wyświetlaczem, a widoki typu *Button* przyciskami kalkulatora.



Rys.1. Projekt layoutu oraz docelowy wygląd aplikacji

Źródło: opracowanie własne.

Na tym etapie pracy niezwykle pomocny będzie Designer. Pozwoli on dodać widoki oraz określić sposób ich pozycjonowania i wygląd. Wynik pracy Designera zapisywany jest w pliku projektu w postaci kodu języka XML (ang. *Extensible Markup Language*, rozszerzalny język znaczników). Wygląd takiego kodu powinien być znajomy dla uczniów, którzy spotkali się wcześniej z projektowaniem stron internetowych i językiem HTML.

[1] S. Hoober, E. Berkman, *Designing Mobile Interfaces. Patterns for Interaction Design*, O'Reilly Media, Sebastopol, Kalifornia 2011.

[2] M. Płonkowski, *Android Studio. Tworzenie aplikacji mobilnych*, Helion, Gliwice 2017.

Oprogramowanie aplikacji

Oprogramowanie aplikacji rozpocząć należy od przygotowania zmiennych, które będą przechowywać referencje do wszystkich kontrolek. Będą one reprezentować kontrolki w kodzie programu. Referencje należy powiązać z widokami layoutu za pomocą funkcji *findViewById (R.id.nazwaKontrolki).*

Przyciski obsługiwane będą za pomocą słuchacza zdarzeń typu *View.OnClickListener().* Słuchacz zdarzeń jest obiektem reagującym na to, co robi użytkownik[3]. W przypadku kalkulatora zdarzeniem, które należy oprogramować, jest kliknięcie na przycisk.

Możliwe są tu dwa podejścia. Stworzenie kilku słuchaczy zależnie od rodzaju przycisku (inny słuchacz dla cyfr, inny dla działań i tak dalej) lub też stworzenie jednego słuchacza zdarzeń obsługującego wszystkie przyciski. To drugie rozwiązanie jest czytelniejsze i łatwiejsze do ewentualnej rozbudowy. Kod słuchacza zdarzeń dla przycisków prezentuje listing 1.

```
private void utworzSluchaczaZdarzen() {
    sluchaczZdarzen = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            int id = view.getId();
            switch(id) {...}
        }
    };
}
```

Listing 1. Kod słuchacza zdarzeń

Źródło: opracowanie własne.

Słuchacz odczytuje ID widoku, na rzecz którego został wywołany i na tej podstawie wybiera odpowiednią akcję za pomocą instrukcji *switch().* Powinien on zostać przypisany do wszystkich przycisków za pomocą metody *.setOnClickListener(sluchaczZdarzen*).

Zadaniem słuchacza zdarzeń jest więc rozpoznanie, na który przycisk kliknięto i wywołanie odpowiedniej funkcji obsługi zdarzenia. Funkcje te odpowiadają za poszczególne funkcjonalności kalkulatora. Pierwszą z nich, którą należy oprogramować, jest obsługa przycisków dodających cyfry. Każde kliknięcie w przycisk powoduje dopisanie odpowiedniej cyfry do łańcucha znaków przechowywanego w kontrolce *TekstView* będącej wyświetlaczem. Operację tę wykonuje metoda *dodajCyfre()* otrzymująca w parametrze znak, który należy dopisać.

[3] C. Collins, M. Galpin, M. Kaeppler, *Android w praktyce*, Helion, Gliwice 2012.

Zauważyć należy, że w kalkulatorze kliknięcie na przycisk z cyfrą nie zawsze powoduje ten sam efekt. W trakcie wpisywania liczby cyfra zostanie po prostu dopisana do ciągu cyfr na ekranie. Jeżeli jednak wcześniej wybrany był znak równości lub znak działania, rozpoczyna się wpisywanie nowej liczby, a w takim przypadku ekran powinien być najpierw skasowany.

Do rozpoznawania, w którym z trybów wpisywania jest kalkulator (czyli jak ma zachować się przycisk) posłużyć się można flagą, czyli globalną zmienną boolowską. Wartość *true* oznacza, że trwa wpisywanie liczby i każdorazowe kliknięcie przycisku dodaje nową cyfrę. Wartość *false* oznaczała będzie, że rozpoczyna się wpisywanie nowej liczby, czyli zawartość kontrolki *textView* ma zostać skasowana i dopiero wtedy dodana zostanie wybrana cyfra. Kod realizujący mechanizm dodawania cyfr zaprezentowany jest na listingu 2. Warunek *if(temp.equals(„0”))* zapobiega wpisaniu kliku zer na początku liczby.

```
private void dodajCyfre(String s) {
    if (trybPisania) {
        String temp = wyswietlacz.getText().toString();
        if (temp.equals("0")) temp="";
        wyswietlacz.setText(temp+s);
    }
    else
    {
        wyswietlacz.setText(s);
        trybPisania=true;
    }
}
```

Listing 2. Dodawanie cyfr na wyświetlacz

Źródło: opracowanie własne.

Kluczowym etapem pracy nad projektem jest oprogramowanie działań. Projektowany kalkulator potrafi wykonywać tylko podstawowe obliczenia arytmetyczne, jednak nawet w tym przypadku jego algorytm nie jest aż taki prosty, jak mogłoby się to wydawać. Praca z kalkulatorem rzadko ogranicza się do wykonywania dwuargumentowych działań w rodzaju 2+2=4. Zwykle mamy do czynienia z ciągami działań (na przykład wielokrotne dodawanie).

Analiza sposobu działania dowolnego kalkulatora pozwoli uczniowi na sformułowanie kilku wniosków.

1. Jeżeli znak działania występuje w ciągu działań, powoduje jedną z dwu operacji: kończy poprzednie działanie i wyświetla wynik lub zapamiętuje, jakie będzie następne działanie do wykonania i czeka na drugi argument tego działania.

2. Inaczej zachowuje się program, gdy przycisk działania jest użyty po raz pierwszy w ciągu działań (rozpoczyna nowy ciąg). Wtedy kliknięcie przycisku nie zmienia liczby wyświetlanej na ekranie, a tylko powoduje zapamiętanie wybranego działania.

3. Znak równości powoduje dokończenie rozpoczętego obliczenia, wyświetla wynik oraz kończy ciąg działań. Następne użycie przycisku operatora działania będzie oznaczało rozpoczęcie nowego ciągu, gdzie pierwszym argumentem jest liczba aktualnie wyświetlana na ekranie

4. Operatory działań jednoargumentowy, takich jak pierwiastek, kwadrat, procent oraz zmiana znaku, modyfikują tylko wartość wyświetlaną aktualnie na ekranie. Reakcja na ich naciśnięcie jest natychmiastowa (nie wymagają znaku równości ani nie oczekują na kolejny argument).

5. Operatory jednoargumentowe nie kończą ciągu operacji.

Sformułowanie powyższych warunków pozwala napisać kod metody obsługi zdarzenia dla przycisków działań oraz przycisku równości zaprezentowana na listingu 3.

```
private void dzialanie(int kodPrzycisku) {
    double wynik=0;
    if (pierwszeDzilanie)
    {
        x1= Double.parseDouble(wyswietlacz.getText().toString());
        pierwszeDzilanie=false;
    }
    else
    {
        x2 = Double.parseDouble(wyswietlacz.getText().toString());
        switch (kodDzialania)
        {
            case 1: wynik = x1+x2; break;
            case 2: wynik = x1-x2; break;
            case 3: wynik = x1*x2; break;
            case 4: wynik = x1/x2; break;
        }
        x1=wynik;
        wyswietlacz.setText(String.valueOf(wynik));
    }
    kodDzialania= kodPrzycisku;
    if (kodPrzycisku==0) {
        pierwszeDzilanie=true;
    }
    trybPisania=false;
}
```

Listing 3. Obsługa przycisków działań

Źródło: opracowanie własne.

Zakończenie

Zaprezentowany przykład nie stanowi alternatywy dla klasycznego podejścia do nauki programowania. Jest raczej jej uzupełnieniem. Dzięki wykonaniu tego rodzaju projektu uczeń zapozna się ze współczesnymi technikami wytwarzania oprogramowania. Będzie mógł przejść przez cały proces produkcyjny aplikacji mobilnej. Nauka programowania wzbogacona o tego rodzaju projekty może lepiej przygotować ucznia do samodzielnego poszerzania swojej wiedzy oraz do przyszłej pracy zawodowej.

Bibliografia

Collins C., Galpin M., Kaeppler M., *Android w praktyce*, Helion, Gliwice 2012.

Hoober S., Berkman E., *Designing Mobile Interfaces. Patterns for Interaction Design*, O'Reilly Media, Sebastopol, Kalifornia 2011.

Płonkowski M., *Android Studio. Tworzenie aplikacji mobilnych*, Helion, Gliwice 2017.