

Programowanie gier i aplikacji na Androida

Opracował: Krzysztof Zawadzki



Czym jest system Android?

Android jest systemem operacyjnym dla urządzeń mobilnych takich jak telefony komórkowe, smartfony, tablety, netbooki itp. Jest to najpopularniejszy system mobilny na świecie. Został opracowany przez niewielką firmę Android Inc. w Kalifornii, która w 2005 roku została przejęta przez firmę Google.

Android jest systemem opartym na jądrze Linuksa i mogącym pracować na platformie ARM, MIPS oraz x86. Aplikacje na Androida tworzy się głównie w języku Java oraz natywnie w C/C++. Jednakże, platforma Android została stworzona głównie z myślą o Java i to właśnie w tym języku można tworzyć pełnoprawne aplikacje.

Przygotowanie środowiska programistycznego

Zainstalowanie pakietu Java Development Kit (JDK).

Odwiędź stronę: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

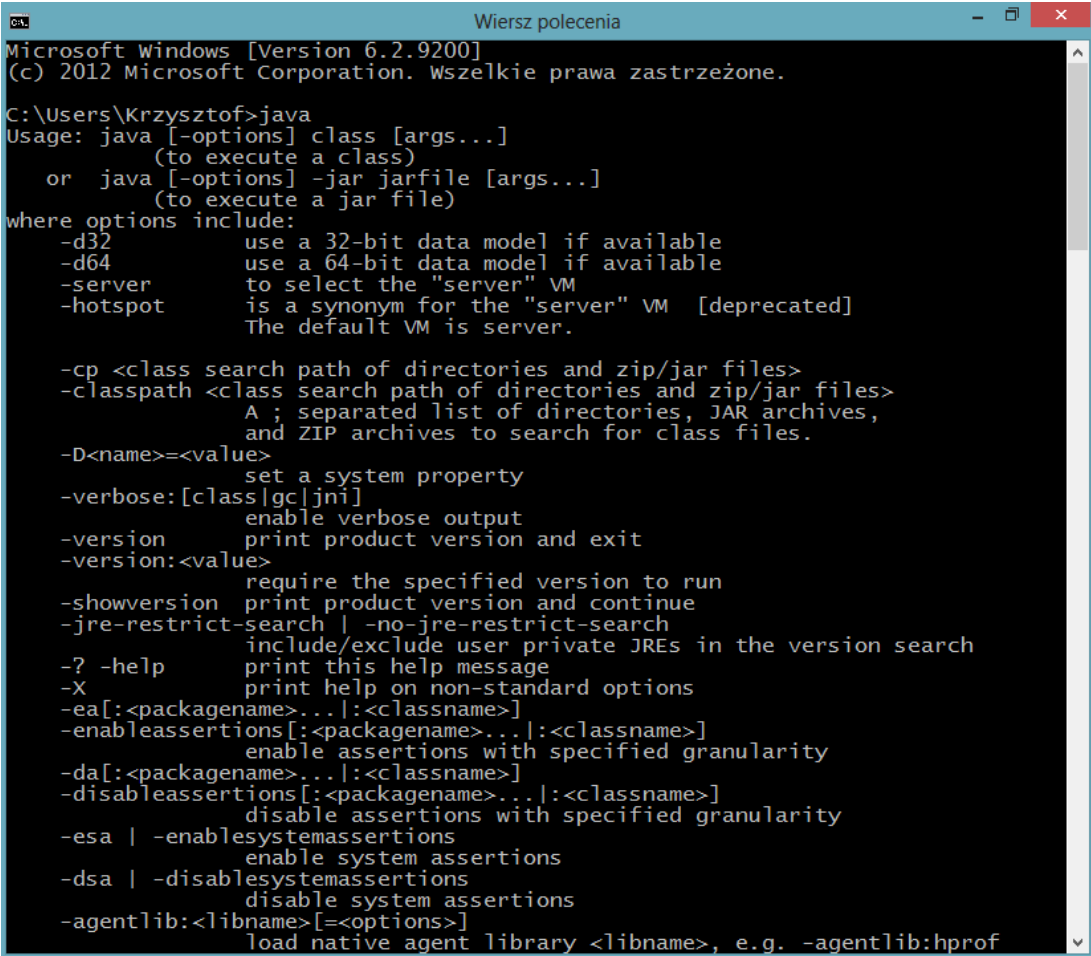
Pobierz wersję JDK odpowiednią dla twojego systemu operacyjnego, a następnie dokonaj instalacji.

Aby IDE mogło odnaleźć ścieżkę do narzędzi Javy (m.in. kompilatora) należy odpowiednio skonfigurować zmienną środowiskową.

Wejdz w **Komputer -> Właściwości -> Zaawansowane ustawienia systemu -> Zakładka Zaawansowane -> Zmienne środowiskowe**

Edytuj bądź utwórz zmienną o nazwie **Path** – jako wartość podaj ścieżkę do katalogu bin pakietu JDK. Przykład: C:\Program Files (x86)\Java\jdk1.7.0_21\bin;

Żeby sprawdzić czy poprawnie wykonałeś instalację JDK uruchom wiersz poleceń (Start -> Uruchom -> cmd). Wpisz kolejno komendy **java** oraz **javac**.



```
Wiersz polecenia
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Krzysztof>java
Usage: java [-options] class [args...]
           (to execute a class)
   or java [-options] -jar jarfile [args...]
           (to execute a jar file)
where options include:
    -d32          use a 32-bit data model if available
    -d64          use a 64-bit data model if available
    -server       to select the "server" VM
                  The default VM is server.
    -cp <class search path of directories and zip/jar files>
    -classpath <class search path of directories and zip/jar files>
                  A ; separated list of directories, JAR archives,
                  and ZIP archives to search for class files.
    -D<name>=<value>
                  set a system property
    -verbose[:<class|gc|jni>]
                  enable verbose output
    -version       print product version and exit
    -version:<value>
                  require the specified version to run
    -showversion   print product version and continue
    -jre-restrict-search | -no-jre-restrict-search
                  include/exclude user private JREs in the version search
    -? -help       print this help message
    -X             print help on non-standard options
    -ea[:<packagename>...[:<classname>]]
    -enableassertions[:<packagename>...[:<classname>]]
                  enable assertions with specified granularity
    -da[:<packagename>...[:<classname>]]
    -disableassertions[:<packagename>...[:<classname>]]
                  disable assertions with specified granularity
    -esa | -enablesystemassertions
                  enable system assertions
    -dsa | -disablesystemassertions
                  disable system assertions
    -agentlib:<libname>[=<options>]
                  load native agent library <libname>, e.g. -agentlib:hprof
```

Komenda java

```
Wiersz polecenia
C:\Users\Krzysztof>javac
Usage: javac <options> <source files>
where possible options include:
  -g               Generate all debugging info
  -g:none          Generate no debugging info
  -g:{lines,vars,source}  Generate only some debugging info
  -nowarn          Generate no warnings
  -verbose         Output messages about what the compiler is doing
  -deprecation     Output source locations where deprecated APIs are used
  -classpath <path> Specify where to find user class files and annotations
  -cp <path>       Specify where to find user class files and annotations
  -sourcepath <path> Specify where to find input source files
  -bootclasspath <path> Override location of bootstrap class files
  -extdirs <dirs>   Override location of installed extensions
  -endorseddirs <dirs> Override location of endorsed standards path
  -proc:{none,only} Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path> Specify where to find annotation processors
  -d <directory>    Specify where to place generated class files
  -s <directory>    Specify where to place generated source files
  -implicit:{none,class} Specify whether or not to generate class files for implicitly referenced files
  -encoding <encoding> Specify character encoding used by source files
  -source <release> Provide source compatibility with specified release
  -target <release>  Generate class files for specific VM version
  -version          Version information
  -help            Print a synopsis of standard options
  -Akey[=value]    Options to pass to annotation processors
  -X              Print a synopsis of nonstandard options
  -J<flag>         Pass <flag> directly to the runtime system
  -Werror          Terminate compilation if warnings occur
  @<filename>      Read options and filenames from file

C:\Users\Krzysztof>
```

Komenda javac

ADT Bundle

ADT Bundle jest przygotowaną przez deweloperów systemu Android gotową paczką zawierającą środowisko **Eclipse**, emulator oraz SDK Androida. Dzięki niej możemy tworzyć aplikacje na Androida, a także aplikacje Javy. Jest to wygodne i szybkie rozwiązanie nie wymagające instalacji – paczkę wystarczy pobrać i wypakować.

ADT Bundle można znaleźć pod adresem: <http://developer.android.com/sdk/index.html>

Android Developers ▾ | Design Develop Distribute

Training API Guides Reference Tools Google Services

Developer Tools

- Download ▴
- Setting Up the ADT Bundle
- Setting Up an Existing IDE ▾
- Android Studio ▾
- Exploring the SDK
- Download the NDK


Get the Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in **ADT (Android Developer Tools)** to streamline your Android app development.

With a single download, the ADT Bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator



Download the SDK
ADT Bundle for Windows

Pobierz paczkę ADT Bundle i wypakuj do odpowiadającej Ci lokalizacji np. C:\

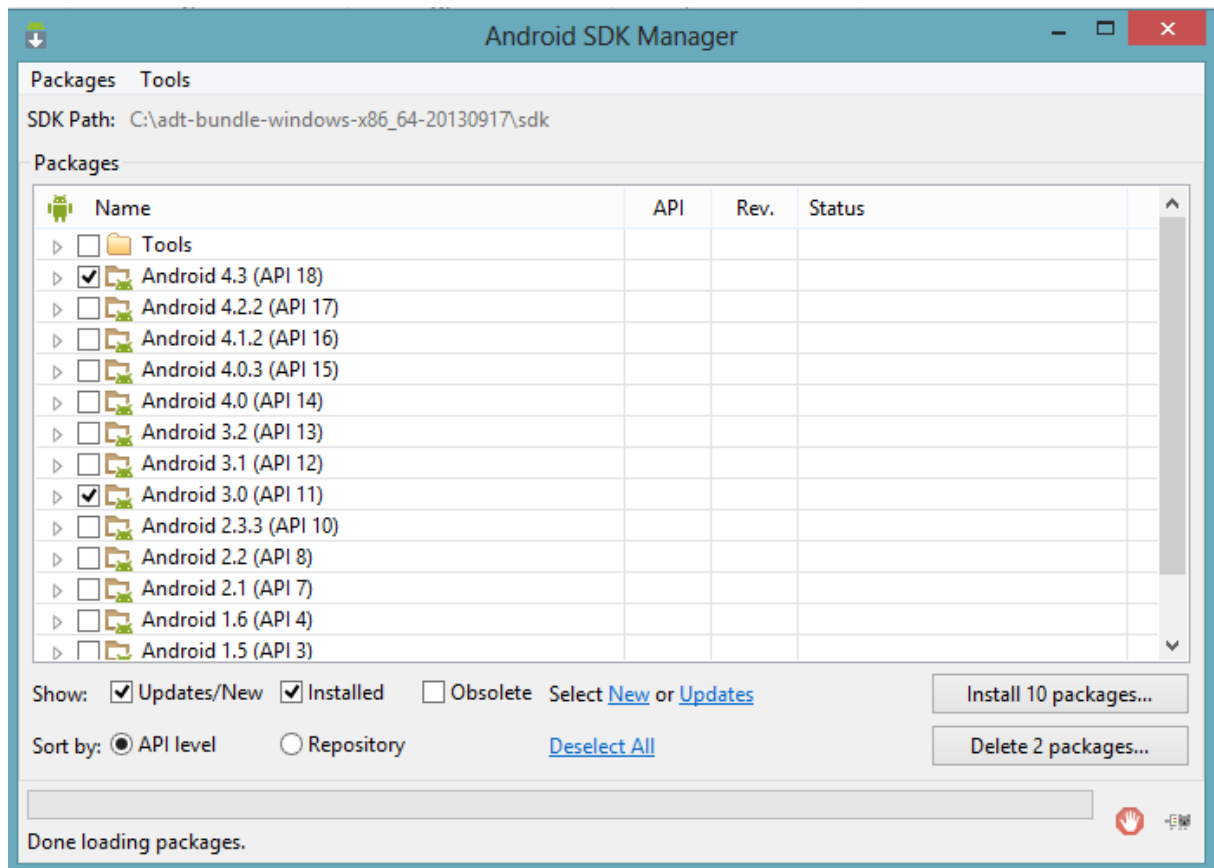
ADT Bundle jest przenośne dlatego też możesz uruchamiać go z pamięci zewnętrznej – wymogiem jest aby na uruchamianym komputerze znajdowało się JDK.

Konfiguracja Android SDK

Uruchom **SDK Manager**, który znajduje się w folderze ADT Bundle np.

C:\adt-bundle-windows-x86_64-20130917\SDK Manager.exe

Żeby móc zacząć programować na platformę Android należy pobrać pliki API Androida. Nie musimy pobierać wszystkich wersji – na potrzeby tej instrukcji pobierzemy wersję Android 3.0 (API 11) oraz najbardziej aktualną Android 4.3 (API 18).



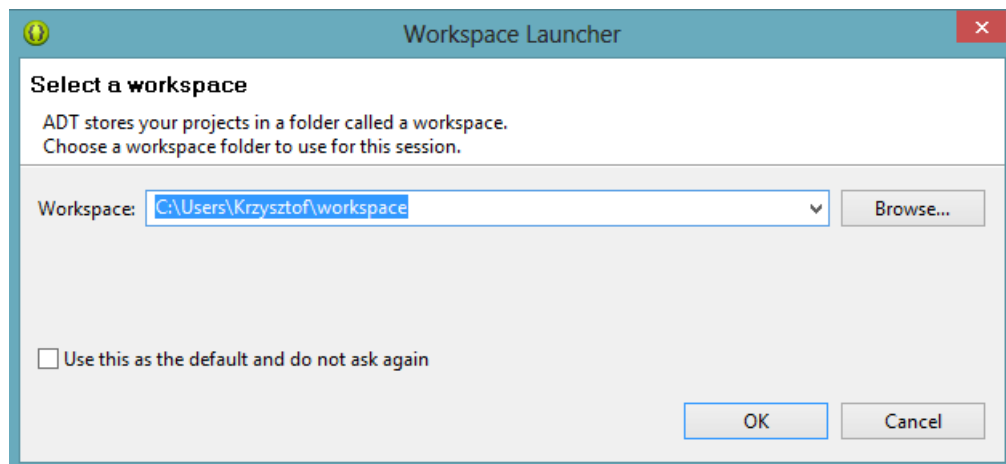
Po wybraniu wersji API kliknij **Install packages** i poczekaj aż wszystkie pliki zostaną ściągnięte.

Konfiguracja środowiska Eclipse

Uruchom program **Eclipse** – znajdziesz go w folderze eclipse znajdującym się w katalogu ADT.

Przykład: C:\adt-bundle-windows-x86_64-20130917\eclipse\eclipse.exe

Po uruchomieniu zostaniesz zapytany o podanie ścieżki do przestrzeni roboczej (workspace). Jest to folder, w którym przechowuje się nasze projekty.

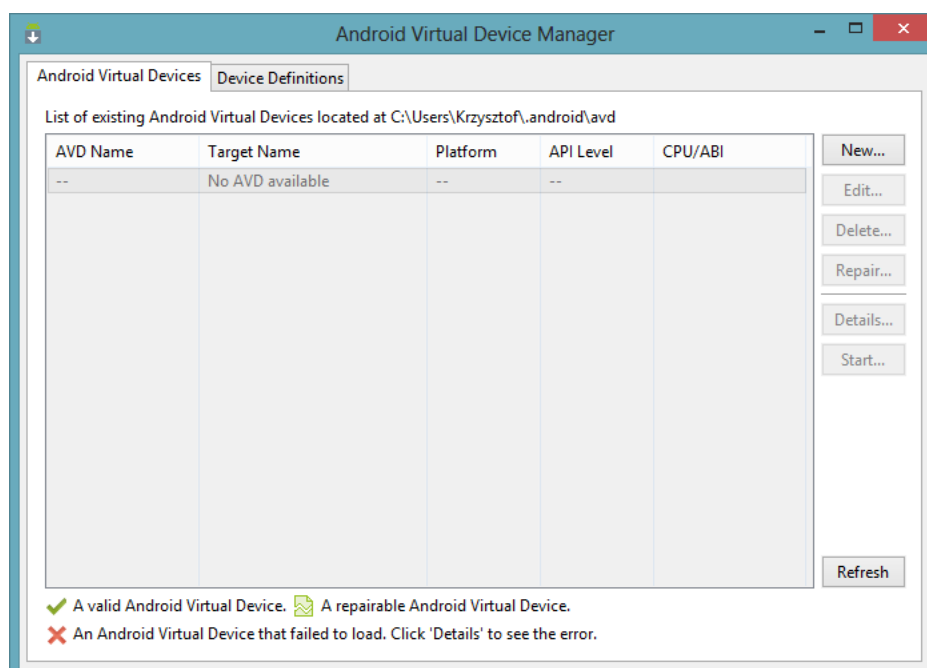


Android Virtual Device Manager – standardowy emulator Androida

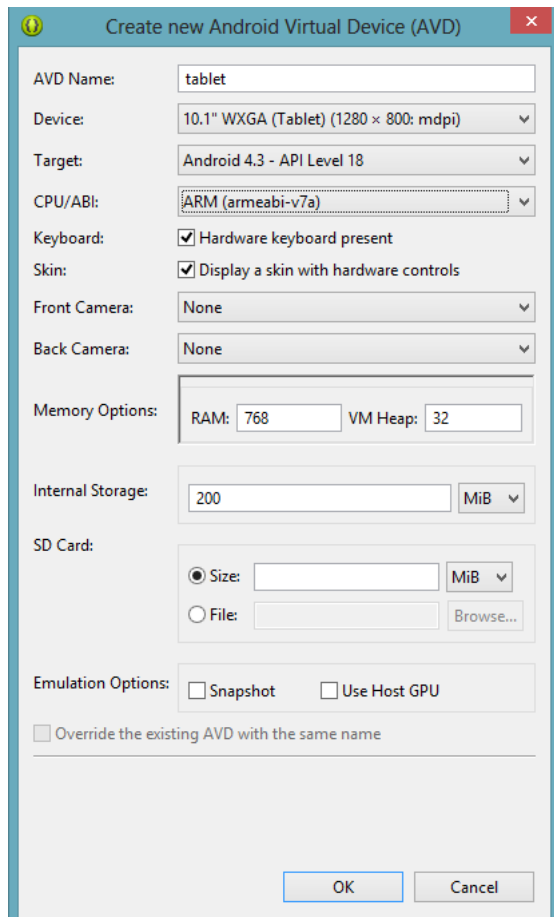
Utworzenie maszyny wirtualnej Androida

Emulator dostarczony razem z ADT Bundle, mimo że został opracowany przez Google posiada swoje wady – pracuje wolno i niestabilnie. Dlatego też polecam skorzystanie z darmowego emulatora **Genymotion**. Jeśli nadal chcesz używać oryginalnego emulatora – wykonaj instrukcje poniżej.

Z poziomu **Eclipse** klikamy na ikonę **Android Virtual Device Manager** znajdującą się na pasku narzędzi. Możemy także wybrać **Window -> Android Virtual Device Manager**.



W celu utworzenia nowego urządzenia kliknij **New**



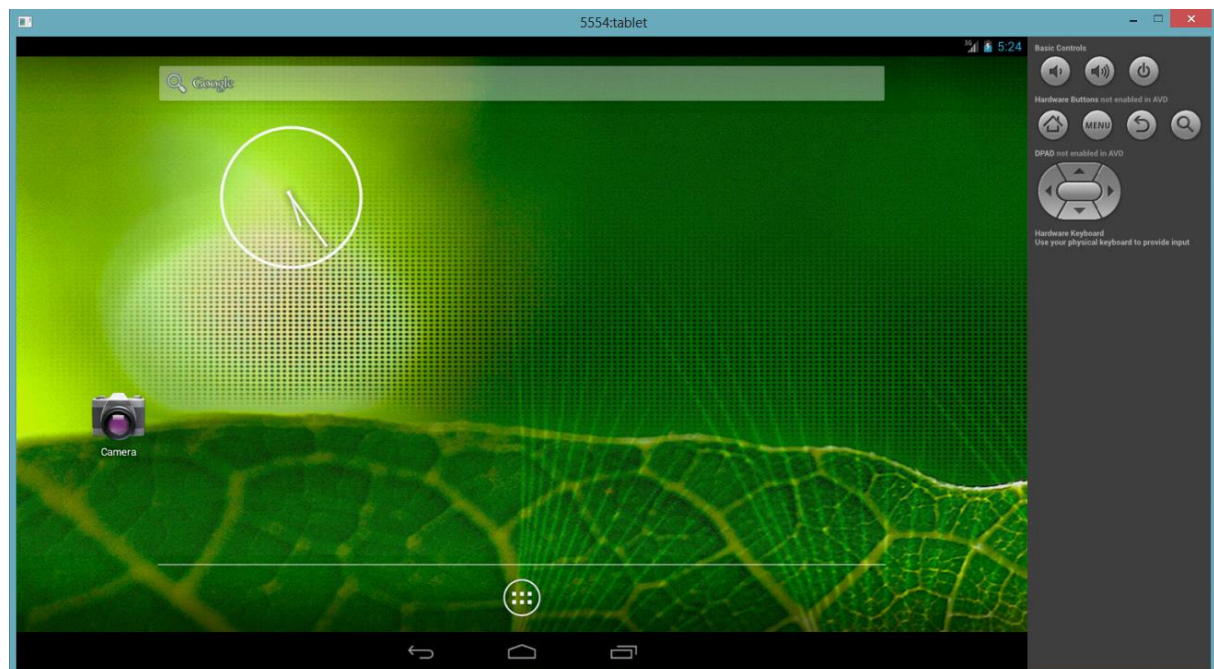
Przypiszmy naszemu urządzeniu parametry widoczne na obrazku. Niech to będzie tablet pracujący pod najnowszym systemem Android 4.3, posiadający 10.1" calowy ekran o rozdzielczości 1280 x 800.

Niestety, przypisanie ilości RAMu większej niż 768 MB na systemie Windows może zakończyć się niepowodzeniem.

Jeżeli posiadamy wydajną kartę graficzną, możemy zaznaczyć opcję **Use Host GPU**, aby emulator mógł korzystać z procesora graficznego – zaznaczenie tej opcji daje zdecydowanie lepszą wydajność.

Możliwe jest utworzenie wielu wirtualnych urządzeń o różnych parametrach.

Aby uruchomić urządzenie – zaznacz go w oknie Android Virtual Device Manager i kliknij **Start**.

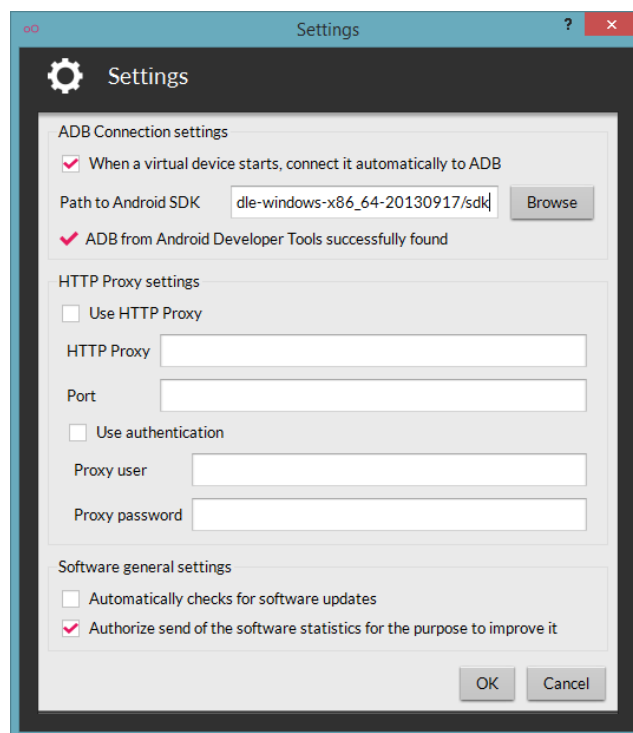


Uruchomione wirtualne urządzenie Android.

Genymotion – znacznie szybszy i wydajniejszy emulator Androida

Genymotion to emulator oparty na programie VirtualBox i pracujący w architekturze x86. Aplikacje Androida działają wyraźnie szybciej w porównaniu do AVD – emulator ten zbiera bardzo dobre recenzje wśród deweloperów. Instalacja i korzystanie z Genymotion jest darmowe.

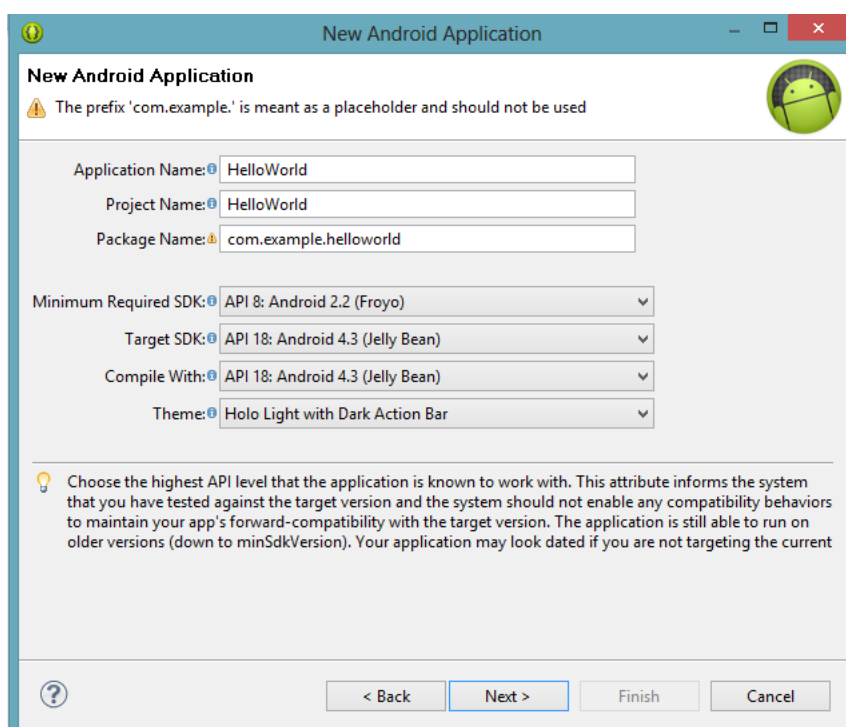
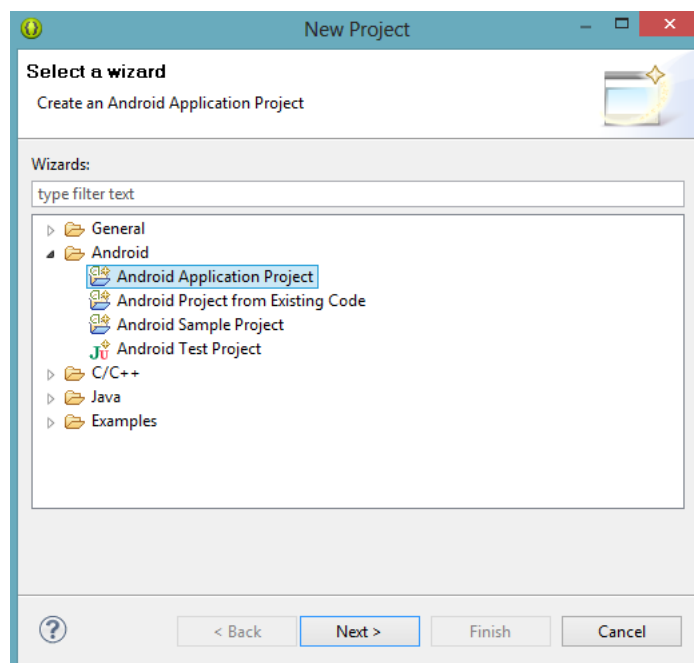
1. Załóż konto na <http://www.genymotion.com/>
2. Zaloguj się i pobierz program
<https://cloud.genymotion.com/page/launchpad/download/>
3. Zainstaluj i uruchom **Genymotion**
4. Kliknij **Add**, potem **Connect**
5. Podaj login i hasło
6. Wybierz typ urządzenia: **WXGA 10.1” Tablet – 4.3 – API 18 – 1280x800**
7. Kliknij **Next**, poczekaj aż Genymotion pobierze dane maszyny wirtualnej
8. Naciśnij **Create**
9. Wejdź do **Settings** – podaj ścieżkę do Android SDK
10. Aby uruchomić maszynę – wybierz urządzenie na liście i naciśnij **Play**
11. Więcej informacji: <https://cloud.genymotion.com/page/doc/>



Tworzenie projektu

Stwórzmy nowy projekt – w tym celu wybierz File -> New -> Project

Wybierz **Android Application Project** i przejdź dalej



Application Name – jest to nazwa naszej aplikacji – nadajmy jej nazwę HelloWorld

Project Name – nazwa folderu dla naszego projektu w przestrzeni roboczej

Package Name – jest to nazwa naszej przestrzeni nazw. Powinna być unikatowa i utworzona na zasadzie odwrócenia domeny.

Dla przykładu: *com.nazwafirmy.nazwaprojektu*

Na razie pozostawmy to pole z domyślną wartością. W późniejszym czasie będziemy nazywać nasze pakiety według zasady: *nazwisko.imie.nazwaprojektu* (bez polskich znaków)

Minimum Required SDK – wybieramy najniższą wersję Androida jaką nasza aplikacja będzie wspierać

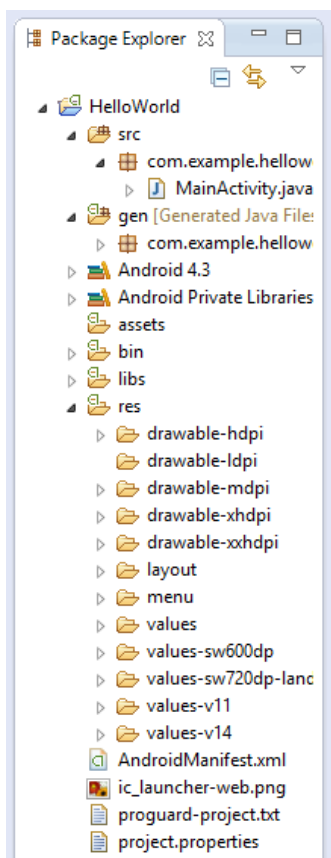
Target SDK – wskazuje najwyższą wersję Androida z jaką testowaliśmy naszą aplikację.

Compile With – wskazujemy wersję Androida, z którą nasza aplikacja się kompiluje. Domyślnie to pole ustawione jest na najnowszą wersję API.

Theme – określa schemat wyglądu interfejsu aplikacji

Kolejne etapy tworzenia projektu możemy pozostawić domyślnie – zajmiemy się tym później.

Struktura aplikacji



src/ - katalog, w którym przechowuje się pliki źródłowe. Domyślnie zawiera on klasę Activity, która zostaje wywołana tuż po uruchomieniu aplikacji

gen/ - zawiera plik R.java, który jest generowany automatycznie – łączy on część programistyczną z zasobami

res/ - katalog, w którym przechowujemy wszelkie zasoby na potrzeby aplikacji np. obrazki, bitmapy, dźwięki, muzykę i inne.

W katalogu **res/layout** znajdują się pliki definiujące wygląd interfejsu użytkownika.

Plik **AndroidManifest.xml** definiuje wiele parametrów takich jak numer wersji aplikacji, wersję SDK, także która Aktywność (Activity) powinna zostać uruchomiona jako pierwsza oraz jakich uprawnień wymaga aplikacja.

Interfejs użytkownika – kod XML

Podstawowe layouty

Linear Layout (Horizontal) – elementy będą się układać od lewej do prawej

Linear Layout (Vertical) – elementy układają się z góry na dół

Relative Layout – elementy ustawiają się względem siebie

Otwórz plik **activity_main.xml** z katalogu **/res/layout**

Otworzy ci się graficzny edytor interfejsu aplikacji. Żeby przejść do kodu wybierz na dole zakładkę **activity_main.xml**

Usuniemy pole tekstowe `<TextView>` oraz zmienimy `<RelativeLayout>` na `<LinearLayout>` i dodamy parametr `android:orientation="horizontal"`

Rezultat:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
</LinearLayout>
```

Dodajemy pole tekstowe

Żeby utworzyć pole tekstowe należy dodać element `<EditText>` wewnątrz `<LinearLayout>`.

Dodatkowo, zajdzie potrzeba dodania pewnych atrybutów definiujących nasze pole tekstowe.

Kod:

```
<EditText android:id="@+id/edit_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="@string/edit_message" />
```

`android:id` – ten atrybut zapewnia unikalny identyfikator dla zasobu.

Znak `@` jest wymagany przy każdym odwołaniu do konkretnego zasobu z poziomu XML.

Znak `+` służy do kreacji (w tym przypadku identyfikatora dla pola tekstowego) i jest potrzebny tylko podczas definiowania elementu po raz pierwszy.

Po znaku `/` podajemy wartość dla naszego `id`, czyli `edit_message` (taki identyfikator otrzyma nasze pole tekstowe)

`android:layout_width` oraz `android:layout_height`

Zamiast definiowania wymiarów pola tekstowego „na sztywno” zastosowaliśmy wartość `"wrap_content"`. Dzięki temu nasz element dostosuje się i będzie na tyle duży, aby pomieścić swoją zawartość. Jeżeli użylibyśmy wartości `"match_parent"` element przejął by wymiary po elemencie nadrzędnym czyli `<LinearLayout>`. W tym wypadku pole tekstowe zajęło by cały ekran.

`android:hint`

Określa on domyślny tekst, który wyświetli się w polu kiedy jest ono puste.

W aplikacjach Android używanie stringów zdefiniowanych na czysto w kodzie jest niezalecane. Zamiast tego używa się zasobów tekstowych (string resources), które tworzy się w pliku `/res/values/strings.xml`.

Wartość `"@string/edit_message"` wskazuje na zasób tekstowy o nazwie `edit_message`, który zaraz utworzymy.

Zauważ: Nadaliśmy taką samą nazwę dla naszego stringa oraz pola tekstowego (`edit_message`). Jednakże nie zachodzi konflikt, ponieważ w Androidzie najpierw odwołujemy się przez typ (np. `@id` lub `@string`).

Dodawanie zasobów tekstowych (string resources)

Przejdź do `res/values/strings.xml`

Kliknij **Add...** Wybierz typ zasobu (String)

W polu **Name** podaj: `enter_message`

W polu **Value** podaj np. *Wpisz wiadomość*

W ten sam sposób utwórz String o nazwie `button_send` i nadaj mu wartość *Wyślij*.

Kod wynikowy:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">HelloWorld</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="edit_message">wpisz wiadomość</string>
    <string name="button_send">wyślij</string>

</resources>
```

Trzymanie wszystkich stringów w jednym pliku niesie ze sobą wiele korzyści np. ułatwia późniejsze przetłumaczenie aplikacji na inny język (wczytujemy wtedy odpowiedni plik strings.xml) oraz znacznie ułatwia zarządzanie nimi i ogranicza szukanie konkretnych wartości w kodzie w razie potrzeby ich zmiany.

Dodajemy przycisk

W pliku **res/layout/activity_main.xml** wklej kod:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send" />
```

Umieść go między znacznikami **<LinearLayout>** zaraz po znaczniku zamykającym element **<EditText>**.

Końcowy kod pliku activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >

    <EditText android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />

</LinearLayout>
```



Uruchamianie kolejnej Aktywności (okna aplikacji)

Activity (Aktywność) to jeden z podstawowych komponentów systemu Android. Klasa ta (a raczej jej podklasy) odpowiedzialna jest za interakcję z użytkownikiem, tworzenie okna naszej aplikacji i uruchamianie innych podstawowych komponentów systemowych. **Zazwyczaj jedna podklasa Activity reprezentuje jedno okno naszej aplikacji.**

Źródło definicji: <http://www.android4devs.pl/2011/07/activity-podstawowe-informacje-cykl-zycia/>

Reakcja na przycisk

Do kodu XML naszego przycisku w pliku **res/layout/activity_main.xml** dodamy atrybut **android:onClick="sendMessage"**

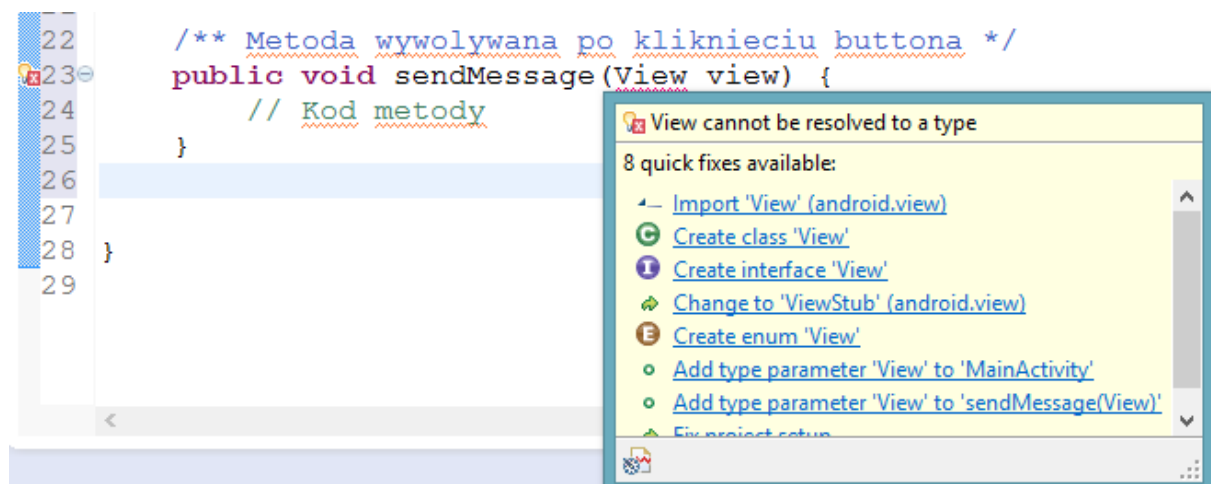
Kod:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

Wartość **"sendMessage"** to nazwa metody, która zostanie wykonana kiedy klikniemy na przycisk.

Otwórz klasę **MainActivity.class** znajdującą się w **src/** i dodaj metodę:

```
/** Metoda wywoływana po kliknięciu buttona */
public void sendMessage(View view) {
    // Kod metody
}
```



Należy zaimportować klasę 'View', aby metoda mogła się poprawnie wykonać. W tym celu najedź myszką na podkreślone na **czerwono** słowo kluczowe **View** i w menu

kontekstowym wybierz: **Import 'View' (android.view)**. Możesz także użyć **skrótów Ctrl + Shift + O** lub ręcznie dopisać liniijkę na początku dokumentu:

```
import android.view.View;
```

Klasa **View** jest to klasa, od której dziedziczą wszelkie podstawowe elementy interfejsu użytkownika takie jak przyciski i pola tekstowe.

Budowanie Intent'u – wysłanie wiadomości

Intent (ang. intencja, zamiar, cel) – jest to obiekt, który zapewnia komunikację między dwoma komponentami np. dwiema Aktywnościami. Intent zazwyczaj jest używany do uruchomienia kolejnej Aktywności i przesłania do niej danych.

W środku metody `sendMessage()` wklej:

```
Intent intent = new Intent(this, DisplayMessageActivity.class);
```

Zaimportuj potrzebne klasy skrótem klawiaturowym. Potrzebny jest:

```
import android.content.Intent;
```

Konstruktor klasy Intent przyjmuje dwa parametry:

Context – definiujemy obiekt, który wywołuje Intent – wpisując `this` wskazujemy na przesłany do funkcji obiekt klasy View (nasz button).

Class – wskazujemy klasę (Aktywność), którą uruchomimy –

DisplayMessageActivity.class – tak będzie się nazywać Aktywność, którą potem utworzymy.

Kod (metoda `sendMessage()`):

```
EditText editText = (EditText) findViewById(R.id.edit_message); // zmienna przechowująca wpisany tekst
```

```
String message = editText.getText().toString(); // wartość pola tekstowego przekonwertowana do stringa
```

```
intent.putExtra(EXTRA_MESSAGE, message); // umieszczamy naszą wiadomość w Intencie
```

Intent może przenosić dane różnych typów jako wartości-klucze zwane **extras**.

Metoda `putExtra()` przyjmuje nazwę klucza jako pierwszy parametr i jego wartość jako drugi argument.

Klucz definiujemy jako zmienną, która jest stała, statyczna oraz publiczna.

Kod (umieszczamy w ciele klasy `MainActivity`):

```
public final static String EXTRA_MESSAGE = "com.example.helloworld.MESSAGE";
```

Definiowanie kluczy z nazwą pakietu jako prefix jest dobrą praktyką. W ten sposób jesteśmy pewni, że są one unikalne i nie kolidują z innymi aplikacjami.

Wywołanie aktywności

Aby uruchomić kolejne okno naszej aplikacji dodaj w metodzie `sendMessage()` linijkę:

```
startActivity(intent); // uruchamiamy aktywnosc
```

Utworzenie aktywności

1. Wybierz **File -> New -> Other...** lub kliknij przycisk **New** na pasku narzędzi.
2. W nowym oknie wybierz folder **Android** i zaznacz **Android Activity**. Kliknij **Next**.
3. Wybierz *BlankActivity*.
4. Wypełnij pola:

Project: *HelloWorld*

Activity Name: *DisplayMessageActivity*

Layout Name: *activity_display_message*

Title: *Moja wiadomość*

5. Kliknij **Finish**

Odbiór Intent'u – odbiór wiadomości

W pliku **src/DisplayMessageActivity.class** w metodzie `onCreate()` dodaj:

```
//odbior wiadomosci z intentu
Intent intent = getIntent();
String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

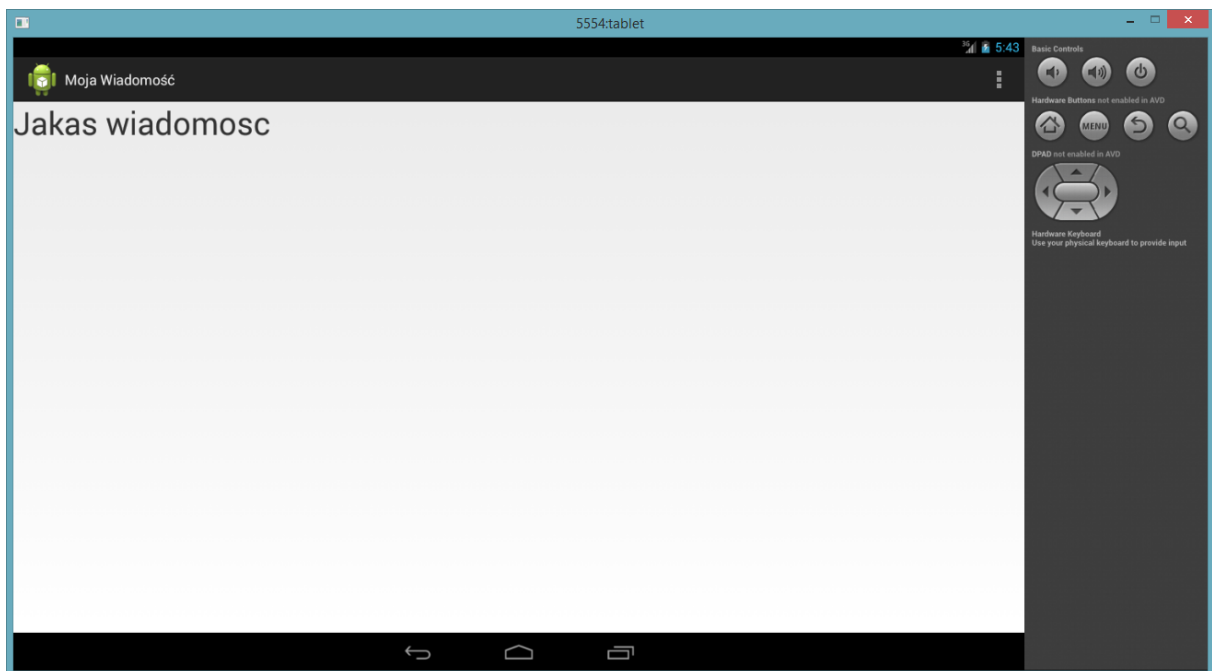
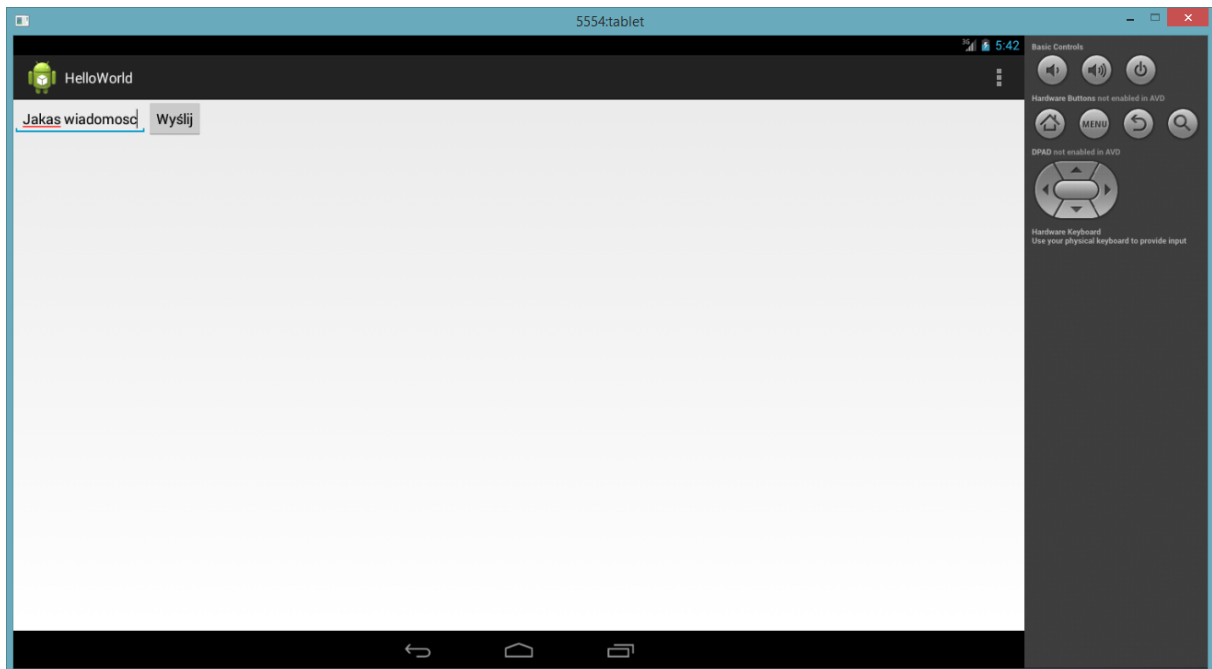
Wyświetlenie wiadomości

Nadal znajdując się w metodzie `onCreate()` dodaj:

```
//tworzenie etykiety
TextView textView = new TextView(this); // utworzenie etykiety
textView.setTextSize(40); //ustawienie rozmiaru tekstu
textView.setText(message); // ustawienie wiadomosci jako tekst
etykiety
// dodanie etykiety do okna aplikacji
setContentView(textView);
```

Uruchomienie i test aplikacji

1. Uruchom maszynę wirtualną
2. Kliknij ikonę Run lub naciśnij Ctrl + F11



Zadanie 1

Przerób aplikację tak, aby pytała o imię użytkownika i wyświetlała komunikat: Witaj + imię

Zadanie 2

Dodaj drugie pole tekstowe. Program ma dodawać dwie liczby i wyświetlać wynik w nowym oknie.

Przykład: Konwersja wartości z pola tekstowego na liczbę całkowitą:

```
EditText pole_tekstowe = (EditText) findViewById(R.id.identyfikator);  
int liczba = Integer.parseInt(pole_tekstowe.getText().toString());
```

Aplikacja – Kalkulator BMI

Nasza kolejna aplikacja będzie prosiła użytkownika o podanie wagi (kg) oraz wzrostu (cm). Na podstawie podanych danych obliczy wskaźnik BMI (Body Mass Index) oraz wyświetli odpowiedni komentarz w zależności od otrzymanej wartości wskaźnika (np. „waga w normie”).

Wzór na **BMI** = masa (w kg) / (wzrost)² (w metrach!)

Utwórz projekt: Android Application Project – nazwa aplikacji: **BMICalc** – nazwa pakietu: **nazwisko.imię.bmicalc** – reszta tak jak poprzednio.

Tworzenie interfejsu

Plik /res/layout/activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context=".MainActivity" >  
    <EditText  
        android:id="@+id/massText"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_alignParentTop="true"  
        android:layout_centerHorizontal="true"  
        android:layout_marginTop="44dp"  
        android:ems="10"  
        android:inputType="number" >
```

```
        <requestFocus />
    </EditText>
```

```
<EditText
    android:id="@+id/heightText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/massText"
    android:layout_below="@+id/massText"
    android:layout_marginTop="47dp"
    android:ems="10"
    android:inputType="number" />
```

```
<TextView
    android:id="@+id/textview2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/massText"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="32dp"
    android:text="@string/height_label" />
```

```
<TextView
    android:id="@+id/textview1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/massText"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="34dp"
    android:text="@string/mass_label" />
```

```
<TextView
    android:id="@+id/resultLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/calculateButton"
    android:layout_centerHorizontal="true"
```

```
        android:layout_marginTop="21dp" />
```

```
<Button
```

```
    android:id="@+id/calculateButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/heightText"
    android:layout_centerHorizontal="true"
    android:onClick="calculate"
    android:text="@string/calculateButton_label" />
```

```
<TextView
```

```
    android:id="@+id/commentLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/resultLabel"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="22dp" />
```

```
</RelativeLayout>
```

Zasoby tekstowe

Plik /res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
```

```
    <string name="app_name">BMICalc</string>
    <string name="action_settings">Ustawienia</string>
    <string name="mass_label">Masa (kg)</string>
    <string name="height_label">Wzrost (cm)</string>
    <string name="calculateButton_label">Oblicz BMI</string>
    <string name="alert">Podaj poprawne dane!</string>
    <string name="underweight">Niedowaga</string>
    <string name="normal">Waga w normie</string>
    <string name="overweight">Nadwaga</string>
    <string name="obeseClassI">I stopień otyłości</string>
    <string name="obeseClassII">II stopień otyłości (otyłość
kliniczna)</string>
```

```
<string name="obeseClassIII">III stopień otyłości (otyłość  
skrajna)</string>  
</resources>
```

Kod aplikacji

Plik /src/MainActivity.java

```
package zawadzki.krzysztof.bmicalc;  
  
import android.app.Activity;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.View;  
import android.widget.EditText;  
import android.widget.TextView;
```

```
public class MainActivity extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }
```

```
    @Override
```

```
    public boolean onCreateOptionsMenu(Menu menu) {  
        getMenuInflater().inflate(R.menu.main, menu);  
        return true;  
    }
```

```
    public void calculate(View view) { // metoda obliczająca nasze BMI po  
kliknięciu buttona
```

```
        int mass = 0; // waga  
        int height = 0; // wzrost
```

```
        //Pobranie obiektów interfejsu
```

```
        TextView result = (TextView) findViewById(R.id.resultLabel);  
        TextView comment = (TextView) findViewById(R.id.commentLabel);  
        EditText mt = (EditText) findViewById(R.id.massText);  
        EditText ht = (EditText) findViewById(R.id.heightText);
```



```

        //Pobranie stringa z zasobow (strings.xml)
        String alert = getResources().getString(R.string.alert); //
komunikat o bledzie: "Podaj poprawne dane!"

        //wyczyszczenie etykiet i zmiana rozmiaru czcionki
        comment.setText("");
        comment.setTextSize(30);
        result.setText("");
        result.setTextSize(30);

        // parsowanie wartosci z pol tekstowych na wartosc liczbową z
obsługa wyjątków (błędów)
        // aplikacja nie zamyka się gdy nie podamy żadnych danych
        try { // proba parsowania
            mass = Integer.parseInt(mt.getText().toString());
        } catch (NumberFormatException nfe) {} // łapiemy wyjątek

        try {
            height = Integer.parseInt(ht.getText().toString());
        } catch (NumberFormatException nfe) {}

        // masa i wzrost nie mogą być ujemne ani zerowe
        if (mass <= 0 || height <= 0) {
            result.setText(alert); // wyświetl komunikat o błędzie
        } else {
            double h = height / 100.00; // wagę w cm zamieniamy na
metry
            double BMI = mass / Math.pow(h, 2); // wzór na BMI
            String r = String.format("%.2f", BMI); //formatujemy
stringa
            result.setText(r); // i wyświetlamy wynik

            // wyświetlamy odpowiednią informację w zależności od
wartości BMI
            if (BMI < 18.5) {

                comment.setText(getResources().getString(R.string.underweight)); //
niedowaga

            } else if (BMI > 18.5 && BMI < 25.00) {

```

```

        comment.setText(getResources().getString(R.string.normal)); // waga w
normie

        } else if (BMI > 25.00 && BMI < 30.00) {

        comment.setText(getResources().getString(R.string.overweight)); //
nadwaga

        } else if (BMI > 30.00 && BMI < 35.00) {

        comment.setText(getResources().getString(R.string.obeseClassI)); // I
stopien otylosci

        } else if (BMI > 35.00 && BMI < 40.00) {

        comment.setText(getResources().getString(R.string.obeseClassII)); //
II stopien otylosci

        } else
        {

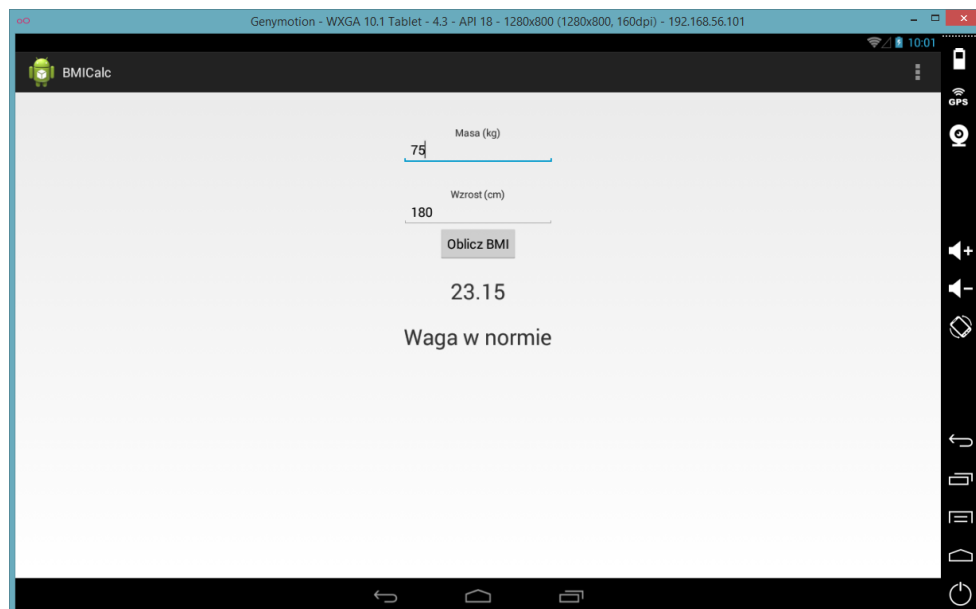
        comment.setText(getResources().getString(R.string.obeseClassIII)); //
III stopien otylosci

        }

    }

}
}

```



Zadanie 3

Zaprojektuj aplikację, która obliczy i wyświetli pole całkowite i objętość stożka.

Aplikacja – Konwerter Liczb

Aplikacja będzie konwertować liczbę dziesiętną na liczbę w systemie binarnym, szesnastkowym i ósemkowym.

Utwórz nowy projekt o nazwie **KonwerterLiczb**, nadaj odpowiednią nazwę pakietu.

Zmiana motywu aplikacji

Przejdź do res/values-v14 i otwórz plik **styles.xml**

Zamień linijkę:

```
<style name="AppBaseTheme" parent="android:Theme.Holo.Light.DarkActionBar">
```

na:

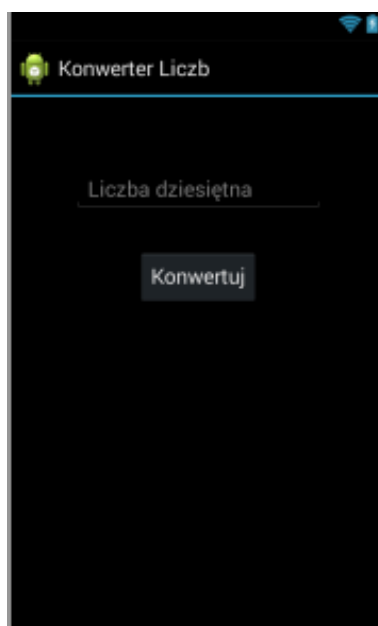
```
<style name="AppBaseTheme" parent="android:Theme.Holo">
```

Od tego momentu aplikacja będzie posiadała ciemny motyw kolorystyczny.

Interfejs

W naszej aplikacji będą potrzebne:

- Element **EditText** o id = editText1 i tekstem (hint) „Liczba dziesiętna” (utwórz i przypisz odpowiednie zasoby tekstowe w strings.xml)
- Element **Button** o id = button1 i tekstem „Konwertuj” i atrybutem **android:onClick="convert"**
- Element **TextView** o id = textView1 i pustym tekście



Funkcja convert

Wklej kod funkcji do pliku src/MainActivity.java

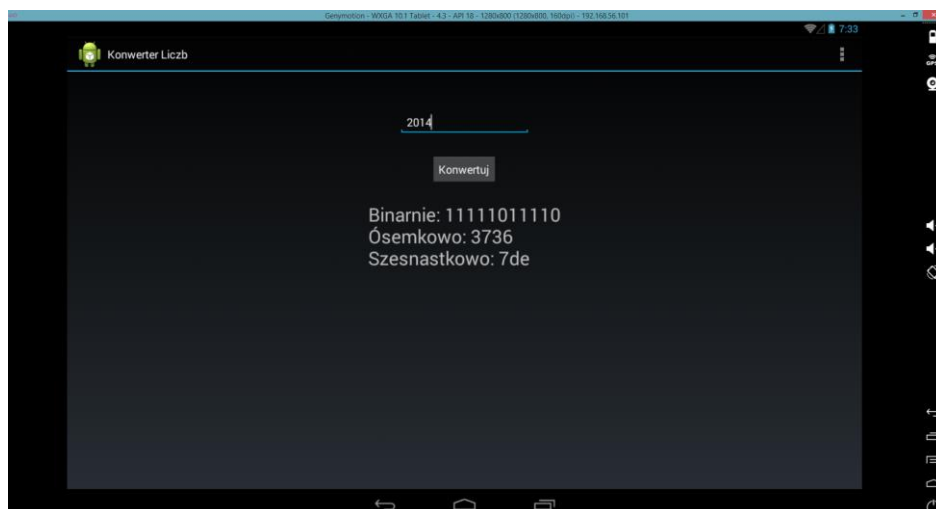
```
public void convert(View view) { // funkcja wywołana po naciśnięciu
    buttona Konwertuj
        EditText editText1 = (EditText) findViewById(R.id.editText1);
    // łapiemy obiekt editText1
        String decimal = editText1.getText().toString(); // pobieramy
    jego zawartość
        int d = 0;

        try { // próba wykonania kodu, w razie niepowodzenia 'łapiemy'
    wyjątek (błąd)
            d = Integer.parseInt(decimal); // konwertujemy
    wprowadzoną liczbę ze stringa na int

                String binary = Integer.toBinaryString(d); // system
    dwójkowy
                String octal = Integer.toOctalString(d); // system
    osemkowy
                String hex = Integer.toHexString(d); // system
    szesnastkowy

                TextView textView1 = (TextView)
    findViewById(R.id.textView1); // łapiemy obiekt textView1
                textView1.setTextSize(30); // ustawiamy rozmiar czcionki
                //wyswietlamy wynik na ekranie
                textView1.setText("Binarnie: " + binary + "\nÓsemkowo: "
    + octal + "\nSzesnastkowo: " + hex);

        } catch (NumberFormatException nfe) {} // łapiemy wyjątek w
    razie błędu
```



Zadanie 4

Wykonaj aplikację, która po podaniu długości w metrach wyświetli daną odległość w calach, stopach, jardach, milach.