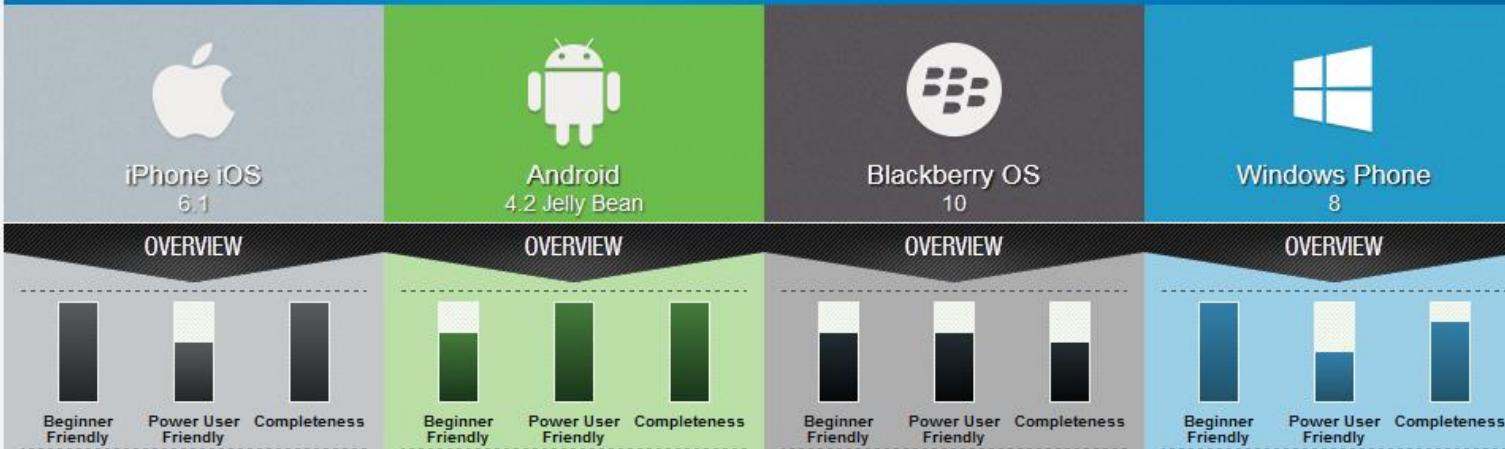


Programowanie urządzeń mobilnych

Wprowadzenie do systemu Android



<http://myphonedeals.co.uk/blog/33-the-smartphone-os-complete-comparison-chart>



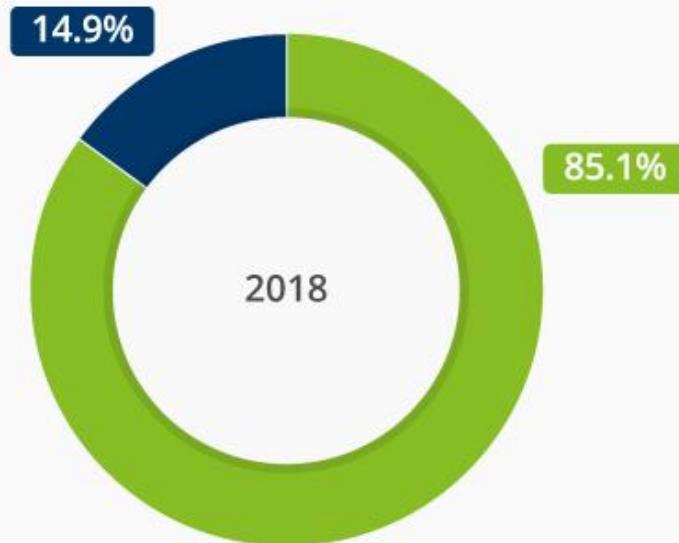
Wojna platform

Segmentacja rynku

The Smartphone Duopoly

Worldwide smartphone market share by operating system (based on unit shipments)

● Android ● iOS ● BlackBerry ● Windows Phone ● Others



Total sales

305m

1,405m



@StatistaCharts

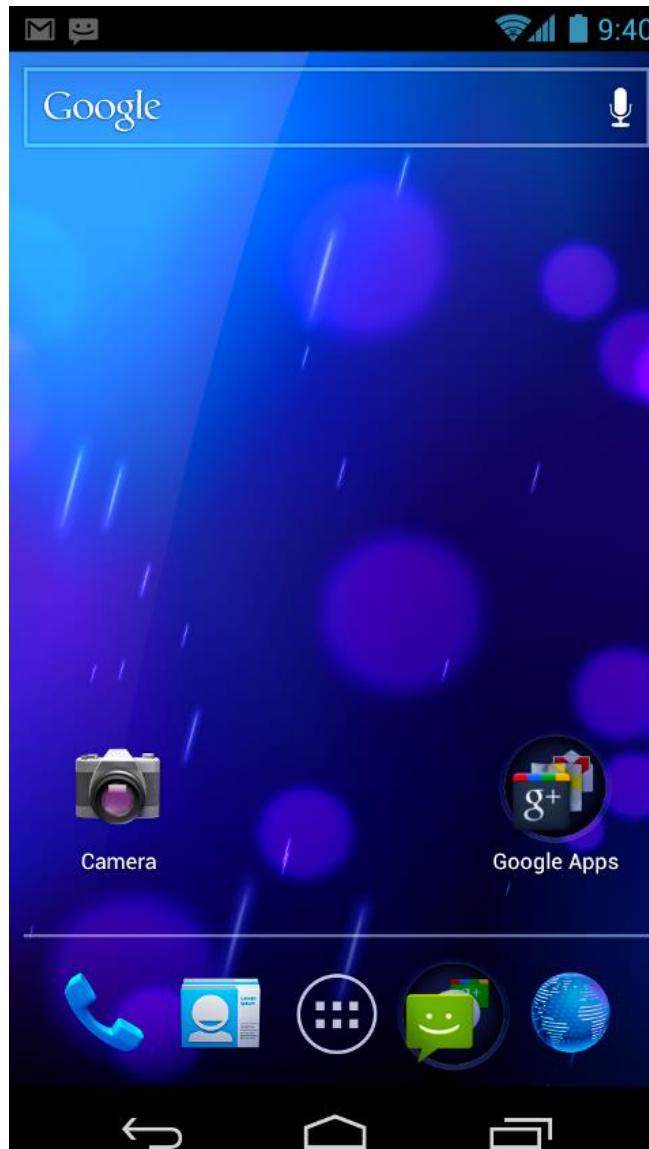
Source: IDC

statista

<https://www.statista.com/chart/3268/smartphone-os-market-share/>

Android – co to takiego?

- ▶ System operacyjny bazujący na jądrze i filozofii Linux'a, przystosowany do urządzeń mobilnych.
- ▶ Tworzony przez Open Handset Alliance (w tym Google Inc.)
- ▶ Projekt typu Open Source.



Krótka historia Androida

2003

- Powstaje **Android Inc.**
- Andy Rubin, Rich Miner, Nick Sears, Chris White

2005

- Android Inc. przejęty przez Google
- Google chciało stworzyć „*a flexible and upgradable system*”

2007

- Świat ujrzał iPhone'a
- Powstaje Open Handset Alliance
- Pierwsza prezentacja Androida

2008

- Pierwsze urządzenie z Androidem – HTC Dream

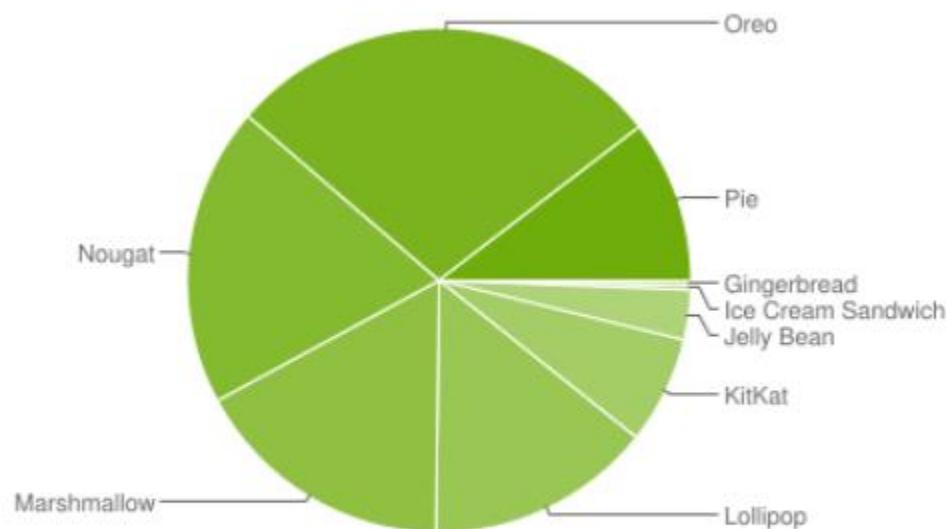


Po co był Android?

Wersje Androida

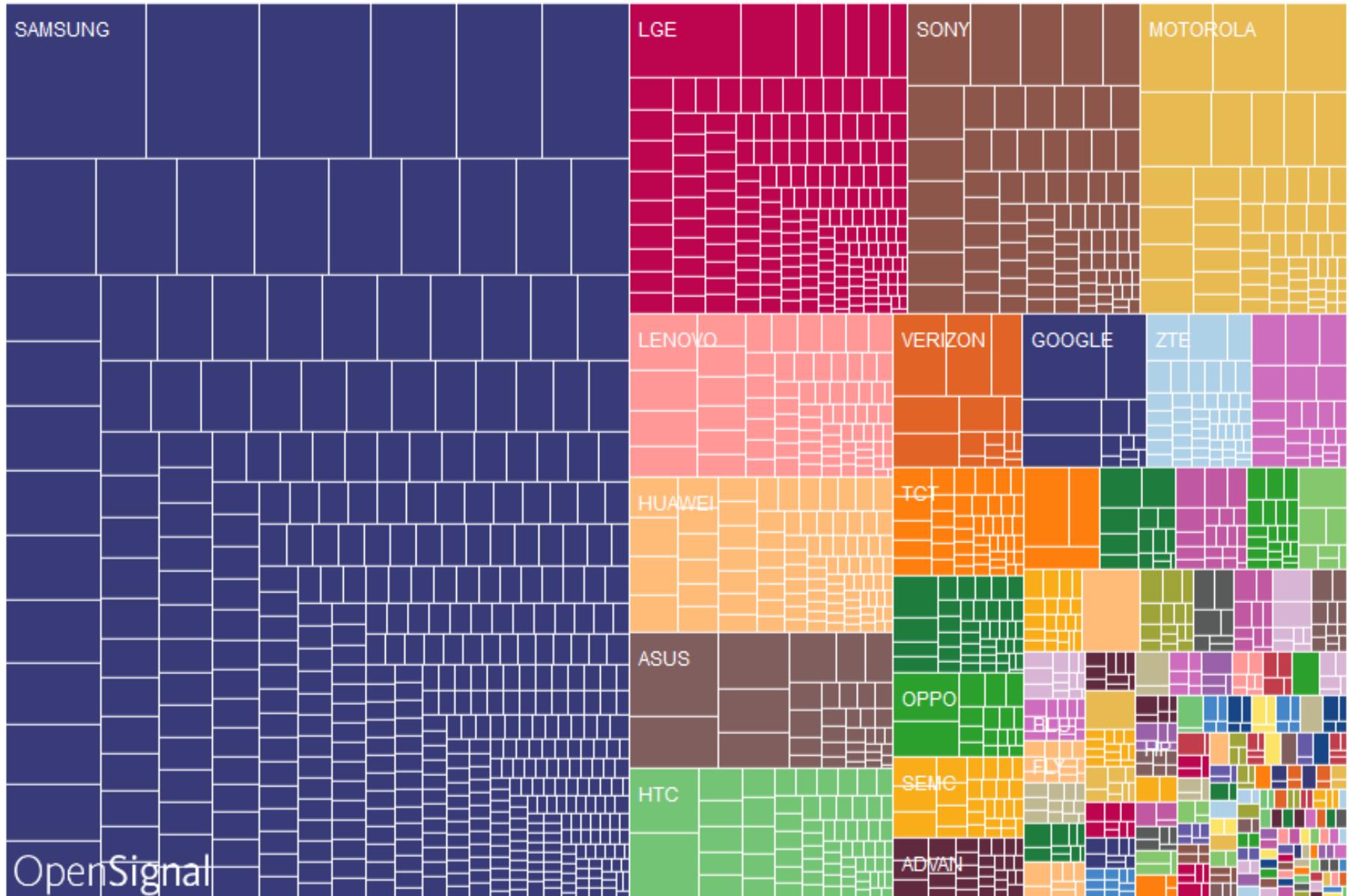
Na którą wersję systemu pisać aplikacje?

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%



Data collected during a 7-day period ending on May 7, 2019.
Any versions with less than 0.1% distribution are not shown.

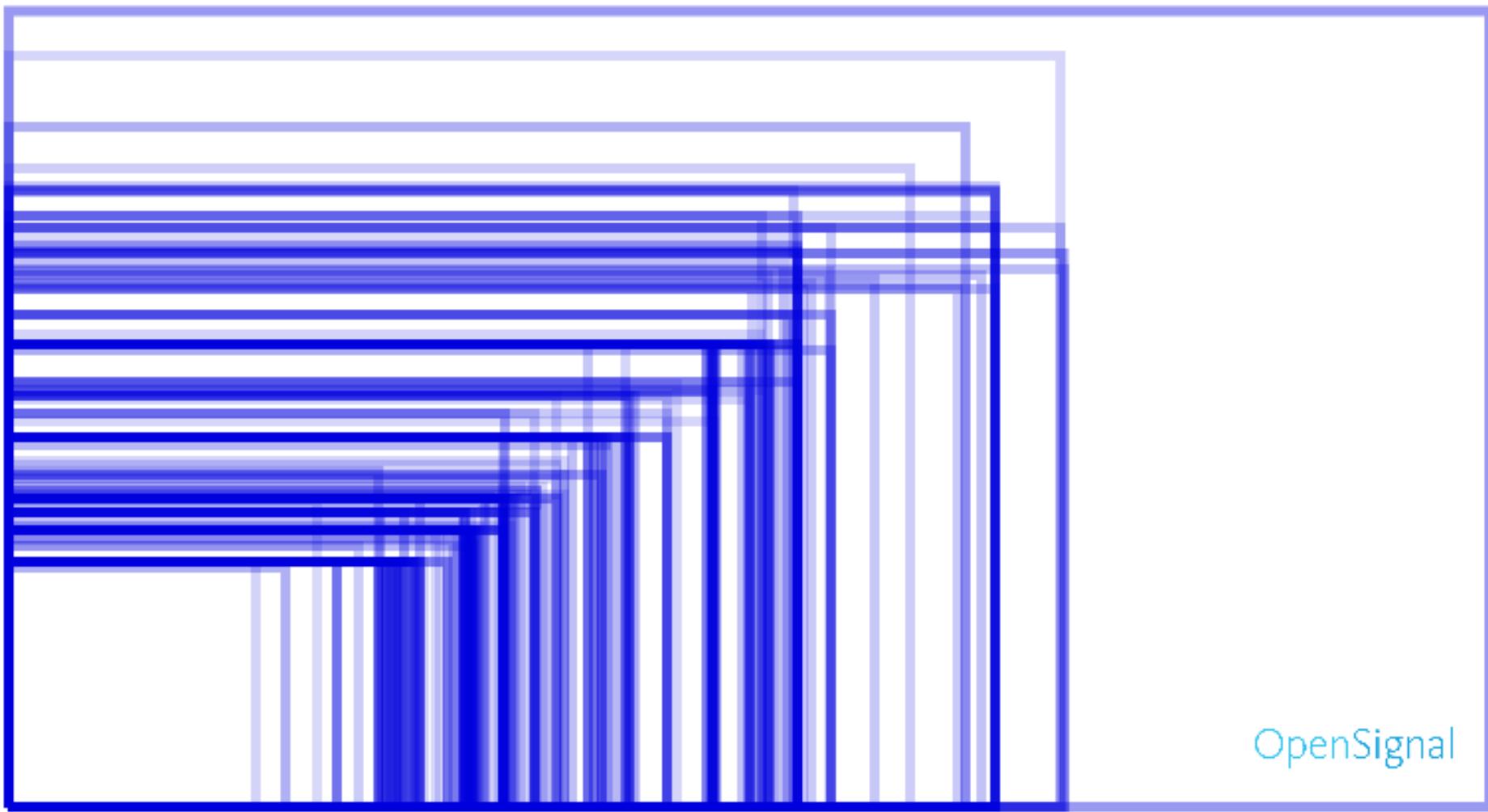
Fragmentacja – producenci



Fragmentacja – urządzenia

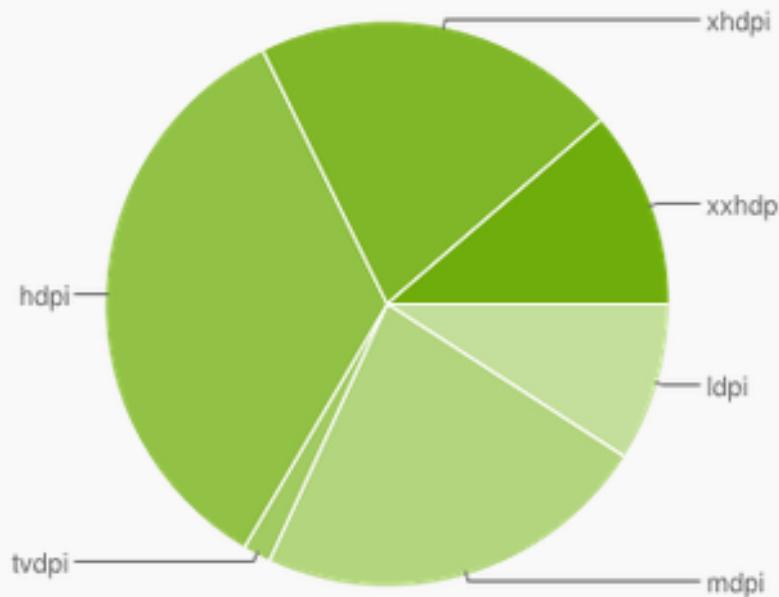
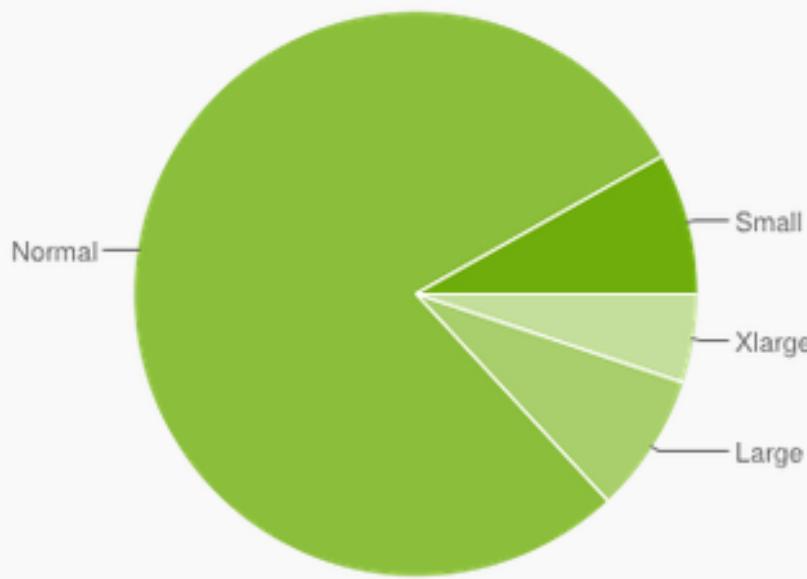


Fragmentacja – rozmiar ekranu



Fragmentacja – rozmiar ekranu c.d.

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	8.1%						8.1%
Normal	0.1%	13.9%		33.3%	20.2%	11.3%	78.8%
Large	0.8%	4.4%	1.6%	0.6%	0.6%		8.0%
Xlarge	0.1%	4.5%		0.3%	0.2%		5.1%
Total	9.1%	22.8%	1.6%	34.2%	21.0%	11.3%	



Data collected during a 7-day period ending on February 4, 2014.

Any screen configurations with less than 0.1% distribution are not shown.

Niezależność od gęstości



Figure 2. Example application without support for different densities, as shown on low, medium, and high density screens.

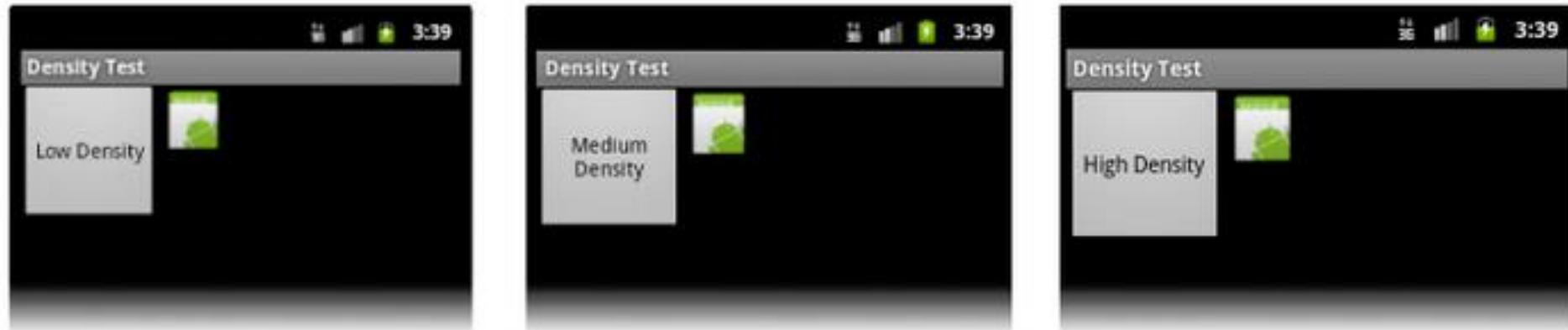
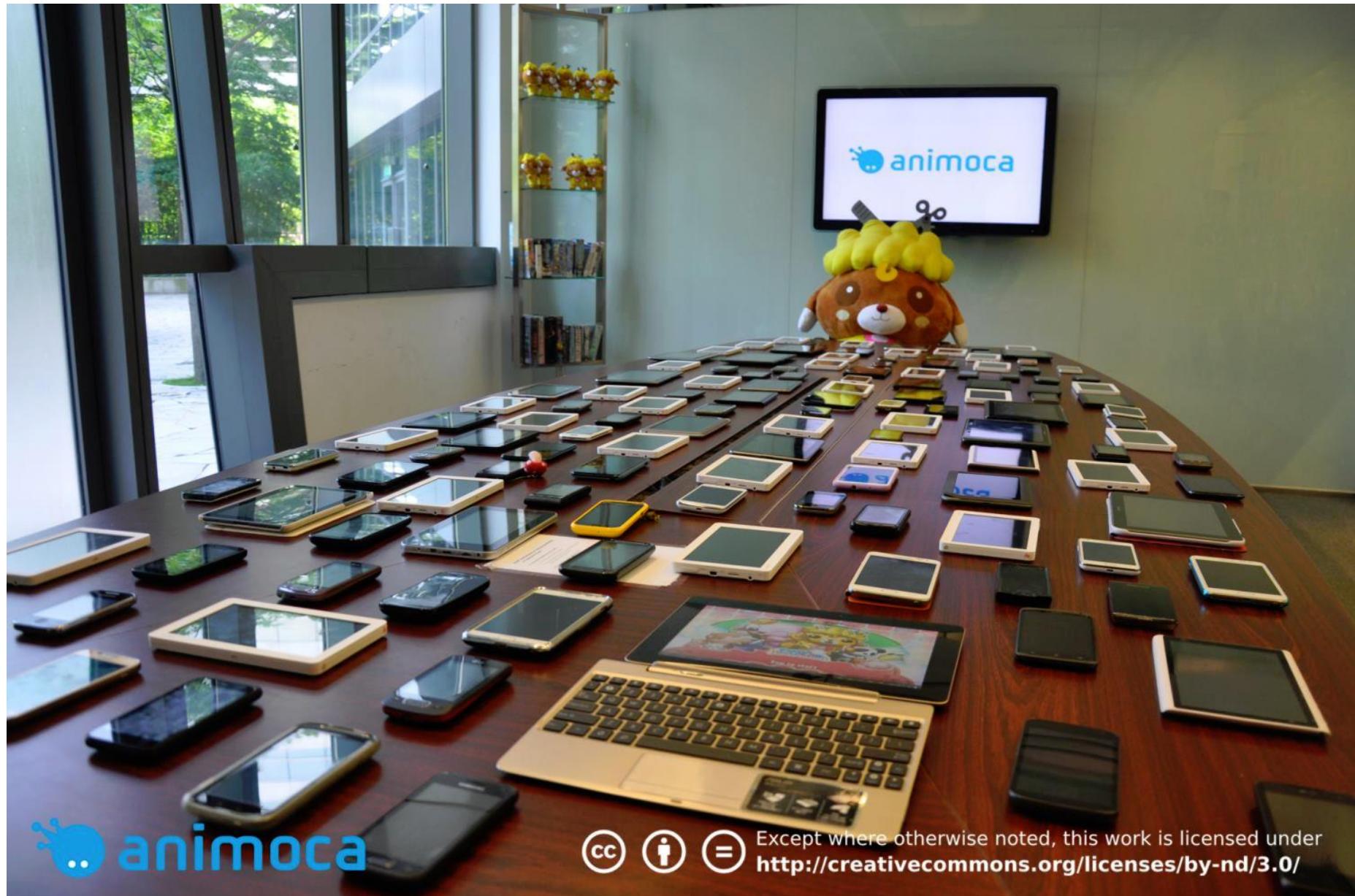


Figure 3. Example application with good support for different densities (it's density independent), as shown on low, medium, and high density screens.

Zbiór urządzeń testowych



APPLICATIONS

Home

Contacts

Phone

Browser

...

APPLICATION FRAMEWORK

Activity Manager

Window Manager

Content Providers

View System

Notification Manager

Package Manager

Telephony Manager

Resource Manager

Location Manager

XMPP Service

LIBRARIES

Surface Manager

Media Framework

SQLite

OpenGLES

FreeType

WebKit

SGL

SSL

libc

ANDROID RUNTIME

Core Libraries

Dalvik Virtual Machine

LINUX KERNEL

Display Driver

Camera Driver

Bluetooth Driver

Flash Memory Driver

Binder (IPC) Driver

USB Driver

Keypad Driver

WiFi Driver

Audio Drivers

Power Management

Anatomia Androida

Najniższa warstwa - kernel

- ▶ Android kernel = Linux kernel po drobnych modyfikacjach
- ▶ Kernel zapewnia m.in.:
 - Warstwę abstrakcji sprzętowej (HAL)
 - Zarządzanie pamięcią
 - Zarządzanie procesami
 - Zarządzanie siecią
 - Zarządzaniem zasilaniem
- ▶ Linux shell komendą: adb shell
- ▶ Czemu właśnie jądro Linux'a?



Biblioteki natywne

- ▶ Kod napisany w C lub C++

Przykładowe elementy:

- ▶ **Surfaces Manager** – zarządzanie wyświetlaniem ekranów
- ▶ **OpenGL|ES** – zarządzanie grafiką 2D i 3D
- ▶ **Media Framework** – kodeki (MPEG 4, H.264, MP3, AAC)
- ▶ **FreeType** – rendering czcionek
- ▶ **SQLite** – przechowywanie danych
- ▶ **Webkit** – silnik przeglądarki internetowej



Dalvik

- ▶ Android Runtime – dostosowano do warunków urządzeń wbudowanych (ograniczone zasoby procesora, pamięci, baterii)
- ▶ Dalvik – własna implementacja Wirtualnej Maszyny (Javy)

Różnice względem JVM:

- ▶ Dalvik operuje na plikach .dex
- ▶ Inny zestaw bibliotek niż w JDK
- ▶ Bardziej zwarta i efektywna implementacja
- ▶ Dalvik jest „register-based VM”

Po co tworzyć własną WM?



Art (od 4.4 i L)

Włączenie:

- ▶ W Androidzie 4.4 ART (Android Runtime) należy uaktywnić w menu Ustawienia -> Opcje programistyczne.

Zalety:

- ▶ (AOT) Ahead-of-time compilation
- ▶ Ulepszony mechanizm garbage collector
- ▶ Ulepszone mechanizmy odplukwania (debuger)

```
java.lang.NullPointerException: Attempt to write to field 'int  
android.accessibilityservice.AccessibilityServiceInfo.flags' on a  
null object reference
```

```
java.lang.NullPointerException: Attempt to invoke virtual method  
'java.lang.String java.lang.Object.toString()' on a null object reference
```

Wady:

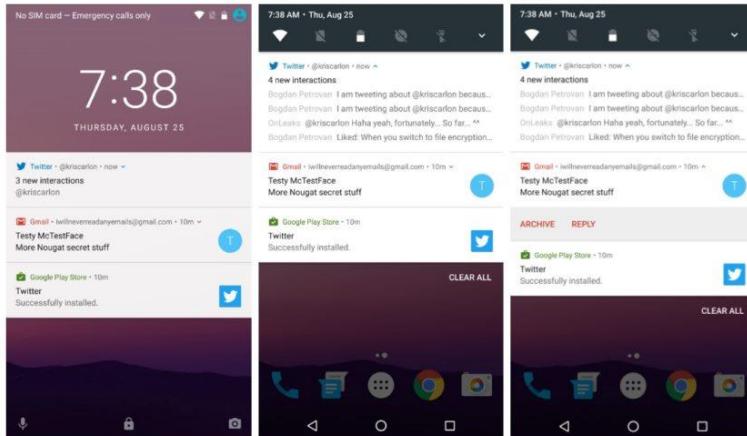
- ▶ Nie wszystkie aplikacje będą w 100% funkcjonować poprawnie

Marshmallow czyli Android 6.0

- ▶ Zmiany w uprawnieniach aplikacji – system pyta o dane uprawnienie (za pierwszym razem) dokładnie w momencie kiedy jest potrzebne.
- ▶ Wsparcie dla czytników linii papilarnych
- ▶ Wsparcie dla USB typu C
- ▶ Usprawnienia asystenta głosowego
- ▶ Wprowadzenie obsługi płatności Android Pay
- ▶ Ulepszenia w systemie oszczędzania energii.

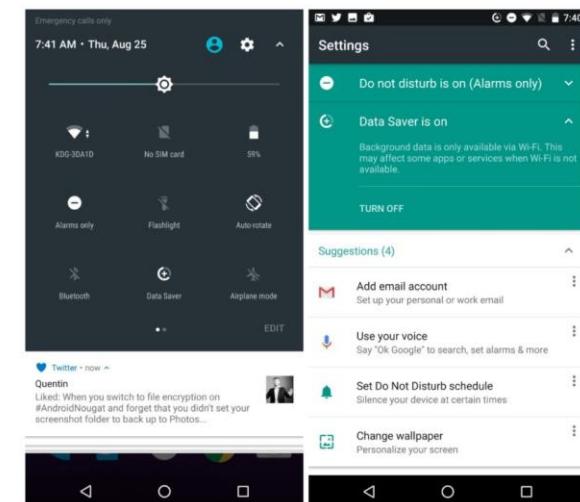
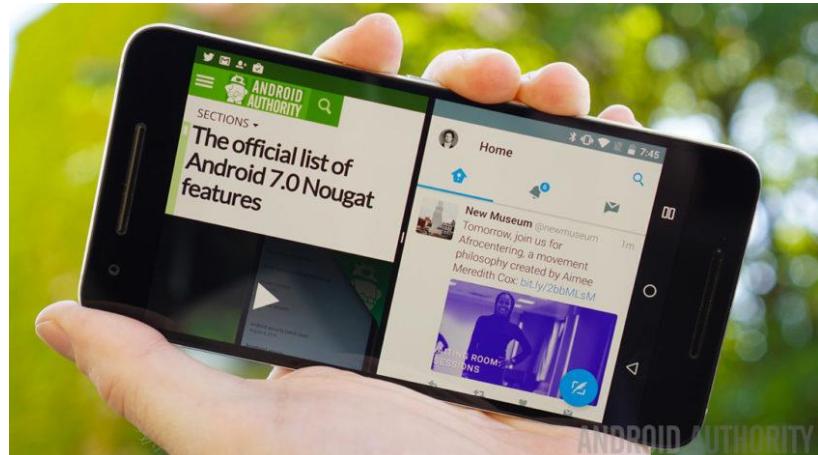
Nugat – co nowego

- ▶ Tryb split screen
- ▶ Priorytety w powiadomieniach



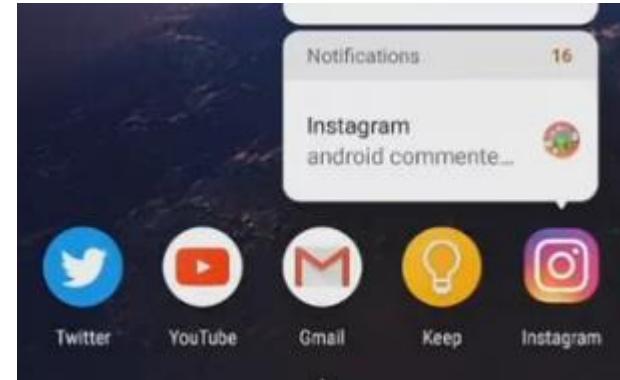
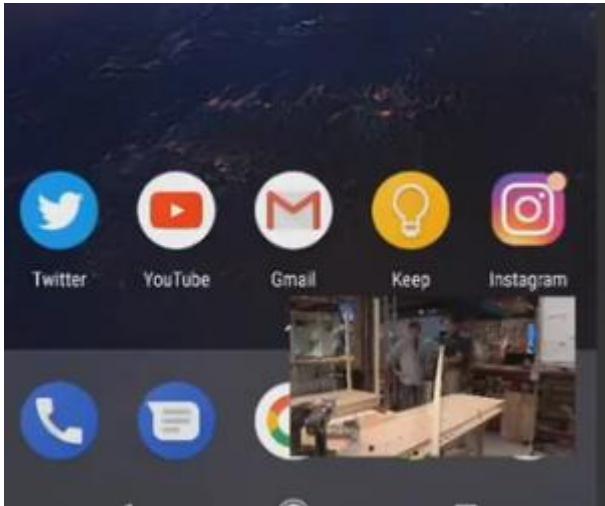
Ulepszony tryb drzemki
Zmiana DPI

- ▶ Tryb nie przeszkadzać
- ▶ Wsparcie dla Vulkan, Java 8



Oreo – co nowego

- ▶ Większe restrykcje dla procesów w tle
- ▶ Notification Dots
- ▶ Tryb Obraz-w-obrazie (PiP)
- ▶ Ambient display



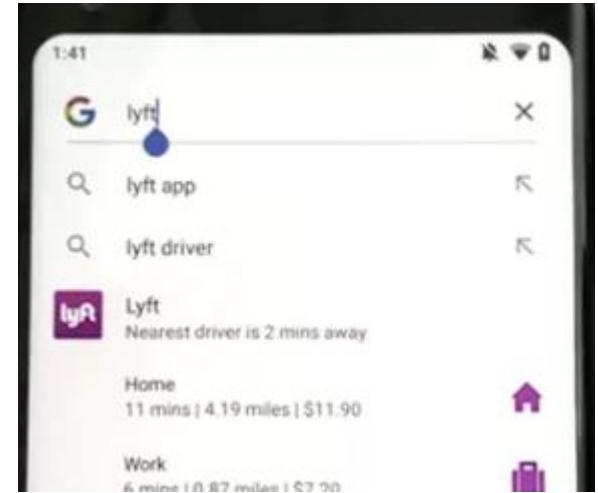
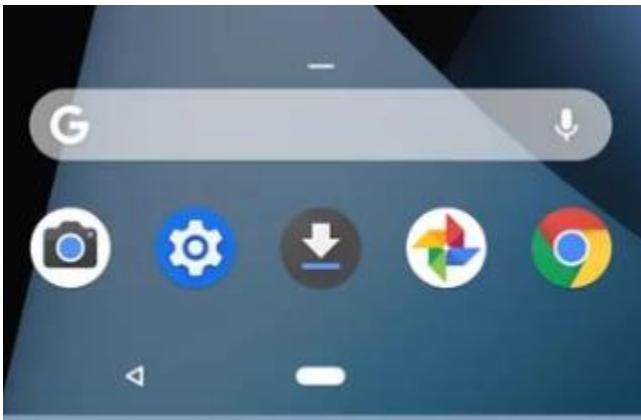
Usprawnienia w wypełnianiu formularzy



- ▶ Wsparcie dla kodeków typu LDAC
- ▶ Asystent do połączeń WIFI

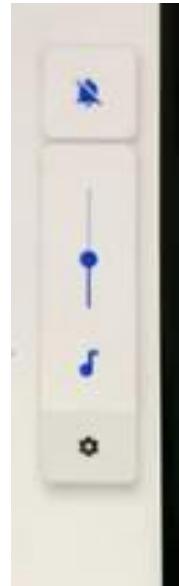
Pie – co nowego

- ▶ Adaptacyjna jasność i bateria
- ▶ Obsługa za pomocą gestów
- ▶ Tryb Slices

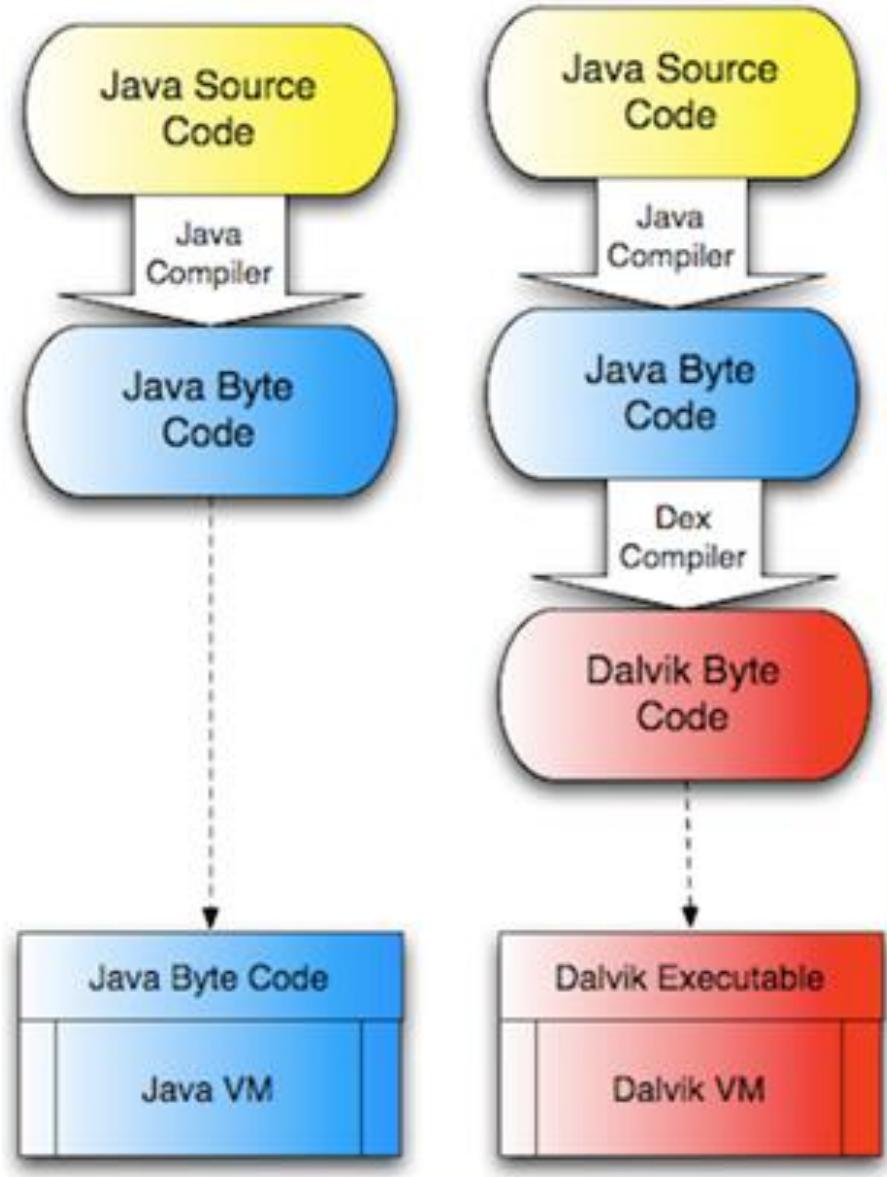
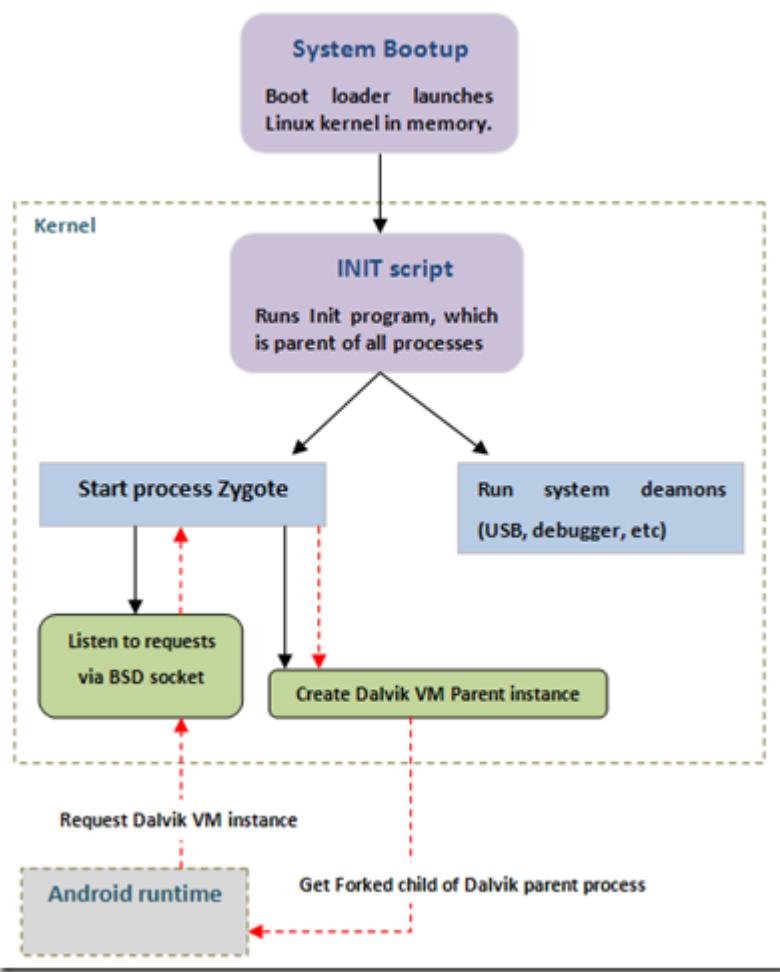


Ulepszone zaznaczanie tekstu

- ▶ Wsparcie dla HDR i ekranów z wcięciem
- ▶ Ograniczenia w dostępie do mikrofonu, kamery i sensorów



Działanie Dalvik VM



Czemu kod Javy nie jest bezpośrednio kompilowany do Dalvik Byte Code?

Application Framework

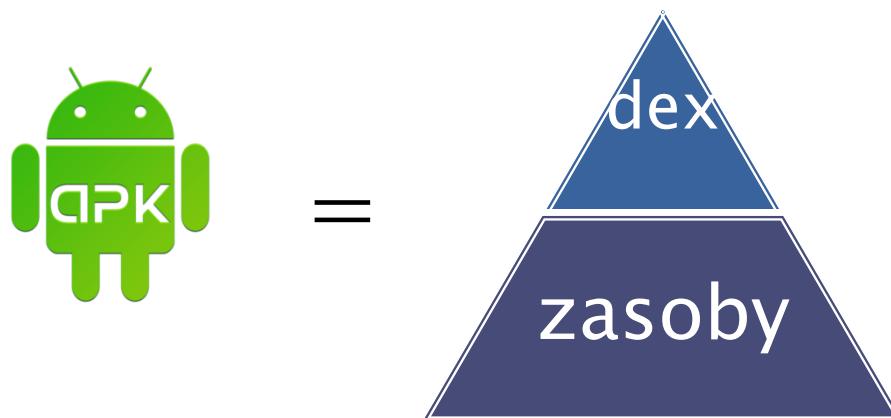
Główne komponenty

- ▶ **Activity Manager** – zarządza cyklem życia (i nawigacją) aplikacji
- ▶ **Package Manager** – zarządzanie zainstalowanymi aplikacjami
- ▶ **Content Providers** – zarządzanie udostępnianiem danych między aplikacjami
- ▶ **View System** – zarządzanie warstwą GUI oraz generowanymi zdarzeniami



Aplikacje

- ▶ Plik wykonywalny Dalvik + Zasoby = APK
- ▶ Aplikacje muszą być podpisane
 - Istnieje też debug key
- ▶ Dostępne są różne sklepy: Google Play, Amazon AppStore, GetJar, AppBrain, F-Droid itp.



APPLICATIONS

Home

Contacts

Phone

Browser

...

Nowy projekt

New Android Application

New Android Application

Creates a new Android Application

Application Name: Hello World

Project Name: HelloWorld

Package Name: pl.tomaszx.helloworld

Minimum Required SDK: API8: Android 2.2 (Froyo)

Target SDK: API19: Android 4.4 (KitKat)

Compile With: API19: Android 4.4 (KitKat)

Theme: Holo Light with Dark Action Bar

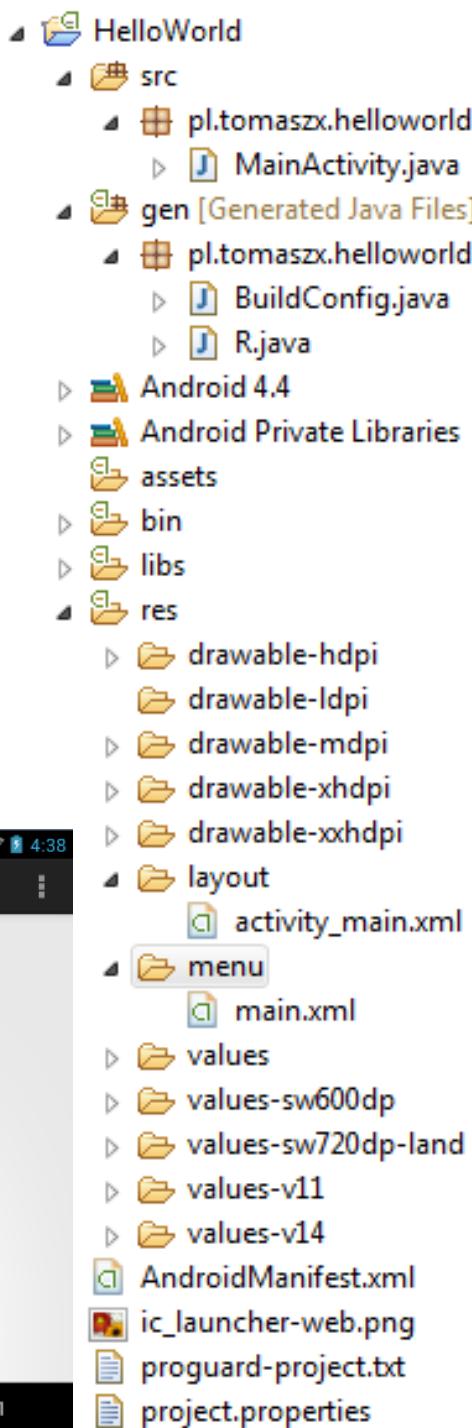
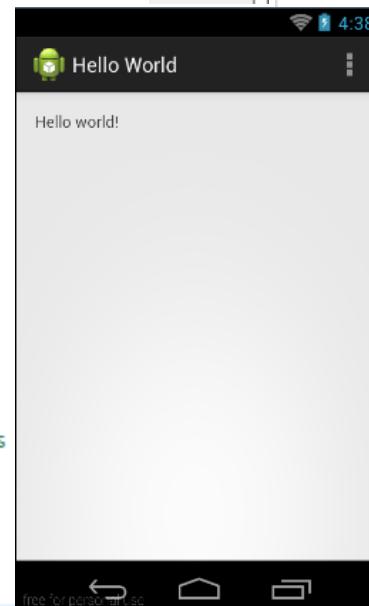
```
package pl.tomaszx.helloworld;

import android.os.Bundle;

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) {
        // Inflate the menu; this adds items to the action bar if it is
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```



Porównanie androidowych marketów

	Google Play	Amazon App Store	SlideMe	GetJar	Mobango
Koszt wgrania aplikacji	25 \$	100 \$	Za darmo	Za darmo	Za darmo
Procent prowizji	30 %	30 %	9 %	0 % przy reklamach	1 \$ per pobranie
Dzienna liczba pobrań	10 milionów lub więcej	Brak danych	Brak danych	3 miliony	1 milion lub więcej
Zalety	<ul style="list-style-type: none"> Dużo potencjalnych klientów Łatwa instalacja aplikacji Instalowany domyślnie Brak recenzji 	<ul style="list-style-type: none"> Zintegrowany z wyszukiwarką sklepu 	<ul style="list-style-type: none"> Szybki proces recenzji Dość duża baza użytkowników 	<ul style="list-style-type: none"> Darmowa promocja Użytkownicy często korzystają szukając promocji 	<ul style="list-style-type: none"> Brak kosztów dla developera (gdy nikt nie ściągnął aplikacji)
Wady	<ul style="list-style-type: none"> Duża konkurencja 	<ul style="list-style-type: none"> Długotrwała recenzja aplikacji 	<ul style="list-style-type: none"> Kiepskie możliwości kontaktu z klientem 	<ul style="list-style-type: none"> Konieczna recenzja aplikacji Tylko darmowe aplikacje Dostęp przez przeglądarkę 	<ul style="list-style-type: none"> Nieprzyjazny UI/wyszukiwanie
Dane na 2012 rok					

Skróty klawiszowe emulatora

Emulated Device Key	Keyboard Key
Home	HOME
Menu (left softkey)	F2 or Page-up button
Star (right softkey)	Shift-F2 or Page Down
Back	ESC
Call/dial button	F3
Hangup/end call button	F4
Search	F5
Power button	F7
Audio volume up button	KEYPAD_PLUS, Ctrl-F5
Audio volume down button	KEYPAD_MINUS, Ctrl-F6
Camera button	Ctrl-KEYPAD_5, Ctrl-F3
Switch to previous layout orientation (for example, portrait, landscape)	KEYPAD_7, Ctrl-F11
Switch to next layout orientation (for example, portrait, landscape)	KEYPAD_9, Ctrl-F12
Toggle cell networking on/off	F8
Toggle code profiling	F9 (only with <code>-trace</code> startup option)
Toggle fullscreen mode	Alt-Enter
Toggle trackball mode	F6
Enter trackball mode temporarily (while key is pressed)	Delete
DPad left/up/right/down	KEYPAD_4/8/6/2
DPad center click	KEYPAD_5
Onion alpha increase/decrease	KEYPAD_MULTIPLY(*) / KEYPAD_DIVIDE(/)

Transfer plików i debug

DDMS - HelloWorld/project.properties - ADT

File Edit Navigate Search Project Run Window Help

Devices X

Name

- genymotion-galaxy_s2_4_1_1_a Online
 - system_process 801
 - com.android.systemui 910
 - com.android.inputmethod.latii 980
 - com.android.phone 998
 - com.android.launcher 1013
 - com.android.smsspush 1061
 - android.process.acore 1076
 - com.android.providers.calenda 1161
 - android.process.media 1186
 - com.genymotion.clipboardpro 1196
 - com.android.email 1222
 - com.android.mmss 1255
 - com.android.voicedialer 1301
 - com.android.defcontainer 1352
 - com.android.musicfx 1368
 - com.svox.pico 1384
 - com.noshufou.android.su 1397
 - com.android.quicksearchbox 1410
 - com.android.keychain 1503
 - pl.tomaszx.helloworld 1582
 - com.android.contacts 1597
 - com.android.exchange 1648
 - com.android.gallery 1677

Threads Heap Allocation Tra... Network Stat... File Explorer Emulator Con... System Infor...

Quick Access Java DDMS

Name Size Date Time Permissions Info

Name	Size	Date	Time	Permissions	Info
acct		2014-02-25	16:33	drwxr-xr-x	
cache		2014-02-23	17:47	drwxrwx---	
config		2014-02-25	16:33	dr-x-----	
d		2014-02-25	16:33	lrwxrwxrwx	-> /sys/ker...
data		2014-02-23	17:53	drwxrwx--x	
default.prop	116	1970-01-01	00:00	-rw-r--r--	
dev		2014-02-25	16:33	drwxr-xr-x	
etc		2014-02-25	16:33	lrwxrwxrwx	-> /system...
fstab.vbox86	625	1970-01-01	00:00	-rw-r----	
init	206451	1970-01-01	00:00	-rwxr-x---	
init.goldfish.rc	2344	1970-01-01	00:00	-rwxr-x---	
init.rc	15537	1970-01-01	00:00	-rwxr-x---	
init.trace.rc	1637	1970-01-01	00:00	-rwxr-x---	
init.usb.rc	3915	1970-01-01	00:00	-rwxr-x---	
init.vbox86.rc	872	1970-01-01	00:00	-rwxr-x---	
mnt		2014-02-25	16:33	drwxrwxr-x	
USB		2014-02-25	16:33	d-----	
asec		2014-02-25	16:33	drwxr-xr-x	
sdcard		2014-02-25	16:33	drwxr-xr-x	
Alarms		1970-01-01	00:00	drwxrwxrwx	
DCIM		2014-02-23	17:51	drwxrwxrwx	
Download		2014-02-25	17:59	drwxrwxrwx	
Chrysanthemum.jpg	87024	2014-02-25	17:59	-rwxrwxrwx	
Movies		2014-02-23	17:51	drwxrwxrwx	
Music		2014-02-23	17:51	drwxrwxrwx	
Notifications		2014-02-23	17:51	drwxrwxrwx	
Pictures		2014-02-23	17:51	drwxrwxrwx	

LogCat X

Saved Filters + -

All messages (no filters) (4'
pl.tomaszx.helloworld (Ses

Search for messages. Accepts Java regexes. Prefix with pid: app: tag: or text: to limit scope.

L... Time PID TID Application Tag Text

161M of 335M Android SDK Content Loader free for personal use

Genymotion for personal use - Galaxy S2 - 4.1.1 - API 16 - 480x800 (480x800)



Dostęp do powłoki systemowej

adb shell

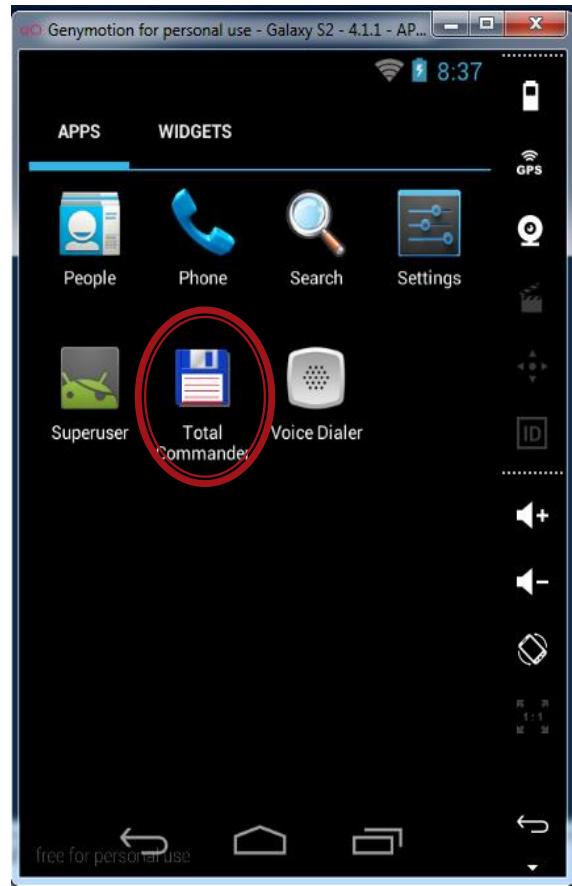
```
PS D:\adt-bundle-windows-x86_64-20131030\sdk\platform-tools> .\adb.exe shell
root@android:/ # ls -al
ls -al
drwxr-xr-x root      root          2014-02-25 20:19 acct
drwxrwx--- system    cache         2014-02-23 17:47 cache
dr-x----- root      root          2014-02-25 20:19 config
lrwxrwxrwx root      root          2014-02-25 20:19 d -> /sys/kernel/debug
drwxrwx--- system    system        2014-02-23 17:53 data
-rw-r--r-- root      root          116 1970-01-01 00:00 default.prop
drwxr-xr-x root      root          2014-02-25 20:19 dev
lrwxrwxrwx root      root          2014-02-25 20:19 etc -> /system/etc
-rw-r----- root      root          625 1970-01-01 00:00 fstab.vbox86
-rwxr-x--- root      root          206451 1970-01-01 00:00 init
-rwxr-x--- root      root          2344 1970-01-01 00:00 init.goldfish.rc
-rwxr-x--- root      root          15537 1970-01-01 00:00 init.rc
-rwxr-x--- root      root          1637 1970-01-01 00:00 init.trace.rc
-rwxr-x--- root      root          3915 1970-01-01 00:00 init.usb.rc
-rwxr-x--- root      root          872 1970-01-01 00:00 init.vbox86.rc
drwxrwxr-x root      system        2014-02-25 20:19 mnt
dr-xr-xr-x root      root          2014-02-25 20:19 proc
drwx----- root      root          2012-10-01 14:14 root
drwxr-x--- root      root          1970-01-01 00:00 sbin
lrwxrwxrwx root      root          2014-02-25 20:19 sdcard -> /mnt/sdcard
dr-xr-xr-x root      root          2014-02-25 20:19 sys
drwxr-xr-x root      root          1970-01-01 00:00 system
-rw-r--r-- root      root          272 1970-01-01 00:00 ueventd.goldfish.rc
-rw-r--r-- root      root          3879 1970-01-01 00:00 ueventd.rc
-rw-r--r-- root      root          30 1970-01-01 00:00 ueventd.vbox86.rc
lrwxrwxrwx root      root          2014-02-25 20:19 vendor -> /system/vendor
root@android:/ #
```

Możliwości ADB

```
PS D:\adt-bundle-windows-x86_64-20131030\ sdk\platform-tools> .\adb.exe -h
Android Debug Bridge version 1.0.31

-a                                - directs adb to listen on all interfaces for a connection
-d                                - directs command to the only connected USB device
-e                                - returns an error if more than one USB device is present.
-s <specific device>             - directs command to the only running emulator.
                                    - returns an error if more than one emulator is running.
-p <product name or path>        - directs command to the device or emulator with the given
                                    serial number or qualifier. Overrides ANDROID_SERIAL
                                    environment variable.
                                    - simple product name like 'sooner', or
                                    a relative/absolute path to a product
                                    out directory like 'out/target/product/sooner'.
                                    If -p is not specified, the ANDROID_PRODUCT_OUT
                                    environment variable is used, which must
                                    be an absolute path.
-H                                - Name of adb server host (default: localhost)
-P                                - Port of adb server (default: 5037)
devices [-l]                         - list all connected devices
                                    ('-l' will also list device qualifiers)
connect <host>[:<port>]           - connect to a device via TCP/IP
                                    Port 5555 is used by default if no port number is specified.
disconnect [<host>[:<port>]]       - disconnect from a TCP/IP device.
                                    Port 5555 is used by default if no port number is specified.
                                    Using this command with no additional arguments
                                    will disconnect from all connected TCP/IP devices.

device commands:
adb push <local> <remote>          - copy file/dir to device
adb pull <remote> [<local>]          - copy file/dir from device
adb sync [<directory>]              - copy host->device only if changed
                                    (-l means list but don't copy)
                                    (see 'adb help all')
adb shell                           - run remote shell interactively
adb shell <command>                 - run remote shell command
adb emu <command>                  - run emulator console command
adb logcat [<filter-spec>]          - View device log
```



```
PS D:\adt-bundle-windows-x86_64-20131030\ sdk\platform-tools> .\adb.exe devices
List of devices attached
192.168.56.101:5555    device
```

```
PS D:\adt-bundle-windows-x86_64-20131030\ sdk\platform-tools> .\adb.exe pull data/app/ApiDemos.apk c:/ApiDemos.apk
5056 KB/s (3050203 bytes in 0.589s)
PS D:\adt-bundle-windows-x86_64-20131030\ sdk\platform-tools> .\adb.exe uninstall data/app/ApiDemos.apk
Failure
PS D:\adt-bundle-windows-x86_64-20131030\ sdk\platform-tools> .\adb.exe install C:\tcandroid204.apk
5988 KB/s (1202118 bytes in 0.196s)
    pkg: /data/local/tmp/tcandroid204.apk
Success
```

Komendy powłoki systemowej

- ▶ Przydatne komendy powłoki systemowej:
 - **ls** listing plików i katalogów
 - **mkdir** stwórz nowy katalog
 - **rmdir** usuń katalog
 - **rm -r** usuń katalog(i) z plikami
 - **rm** usuń pliki
 - **mv** przenoszenie/zmiany nazwy plików
 - **cat** podgląd zawartości plików
 - **cd** zmiana aktualnej ścieżki
 - **pwd** aktualna ścieżka
 - **df** wolne miejsce na dysku
 - **chmod** zmiana uprawnień (do pliku/katalogu)
 - **date** wyświetlenie daty
 - **exit** zakończenie sesji

- ▶ Kiedyś nie było komendy kopiowania (cp), ale można ją zastąpić polecienniem cat:

```
#cat data/app/ApiDemos.apk > cache/apiDemos.apk
```

Wysyłanie wiadomości SMS / wykonywanie połączeń - Emulator

- ▶ telnet localhost **5554**
- ▶ sms send <numer> <treść>
- ▶ gsm call <numer dzwoniącego>

Android console command help:

```
help|hi?          print a list of commands
event             simulate hardware events
geo               Geo-location commands
gsm               GSM related commands
cdma              CDMA related commands
kill              kill the emulator instance
network           manage network settings
power             power related commands
quit|exit         quit control session
redir             manage port redirections
sms               SMS related commands
avd               control virtual device execution
window            manage emulator window
qemu              QEMU-specific commands
sensor            manage emulator sensors

try 'help <command>' for command-specific help
OK
OK    sms send 12345678 sms testowy
OK
```



SMS/Połączenia z GUI

The screenshot shows the Android Studio Emulator Control interface. On the left, the Devices tab displays a list of connected emulators, with "emulator-5554" selected. In the center, the Telephony Actions section is highlighted with a red circle, showing an incoming call from number 21112332. The user can choose between "Voice" or "SMS" and enter a message. Below this is the Location Controls section, also circled in red, which allows setting coordinates via Decimal or Sexagesimal input and sending them. On the right, the emulator screen shows an incoming call from "5554:GalaxyS" with the number 21112332 and the text "INCOMING CALL". A large red circle highlights the "Emulator Control" tab at the top of the interface.

Devices

Name	Status	Device
emulator-5554	Online	GalaxyS [...]
system_process	289	8600
com.android.systemui	402	8601
com.android.inputmethod.latin	430	8611
com.android.phone	443	8613
com.android.settings	457	8615
com.android.music	500	8619
android.process.acore	519	8621
com.android.launcher	537	8617
android.process.media	543	8623
com.android.quicksearchbox	580	8625
com.android.contacts	613	8627
com.android.mms	636	8629
com.android.deskclock	671	8630
com.android.exchange	698	8631
com.android.providers.calendar	716	8632
com.android.calendar	732	8633
com.android.location.fused	753	8634

Threads Heap Allocation Tracker Network Statistics File Explorer Emulator Control System Information

Telephony Status

Voice: home Speed: Full
Data: home Latency: None

Telephony Actions

Incoming number: 21112332

Voice
 SMS

Message:

Call Hang Up

Location Controls

Manual GPX KML

Decimal
 Sexagesimal

Longitude: -122,084095
Latitude: 37,422006

Send

5554:GalaxyS

3G 5:44

21112332

INCOMING CALL

A large red circle highlights the "Emulator Control" tab at the top of the interface.

Cykl życia aplikacji

- ▶ Każda aplikacja uruchamiana jest jako należąca do innego użytkownika.
- ▶ Tylko użytkownik o określonym ID (właściciel) ma bezpośredni dostęp do wszystkich plików danej aplikacji.
- ▶ Każda aplikacja jest izolowana od pozostałych - uruchamiana w osobnej instancji wirtualnej maszyny.
- ▶ Każda aplikacja to osobny proces systemowy. Proces jest uruchamiany, gdy jakikolwiek komponent aplikacji musi zostać uruchomiony i zakończony gdy nie jest potrzebny bądź należy zwolnić zasoby (w wyniku ich zapełnienia)

Elementy składowe aplikacji

- ▶ Aplikacja na Androida składa się z jednego lub więcej elementu składowego.
- ▶ Takim elementem może być:
 1. **Activity**
 2. **Service**
 3. **Broadcast receiver**
 4. **Content provider**

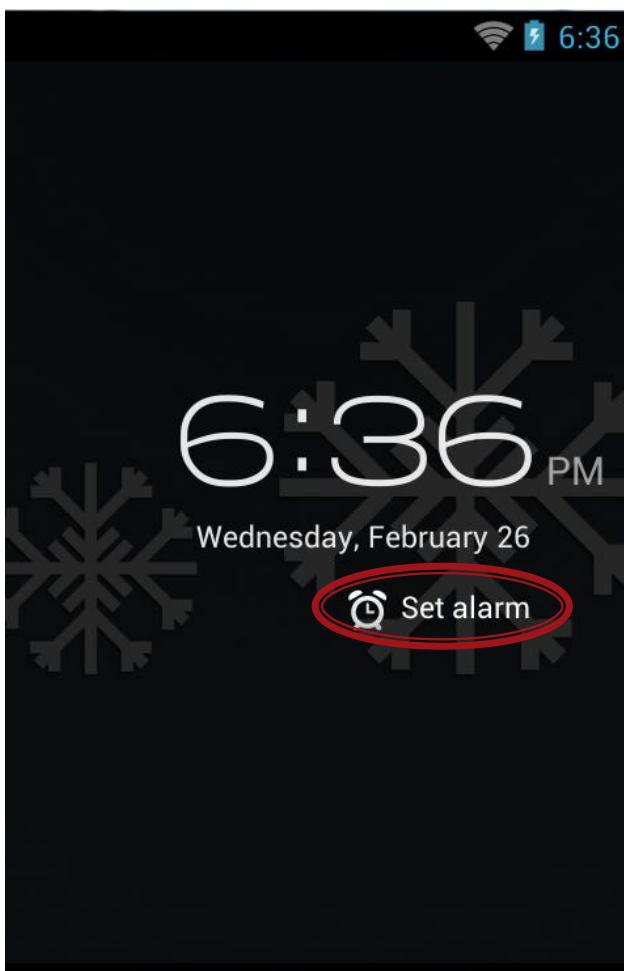


Activity (aktywność)

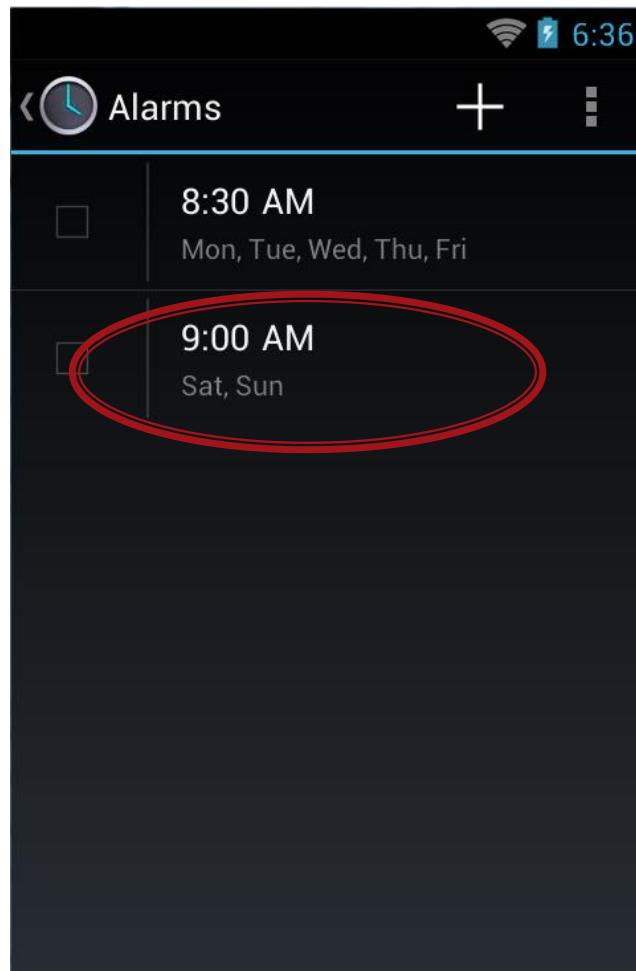
- ▶ Zwykle aplikacja składa się z jednej lub więcej aktywności.
- ▶ Tylko jedna aktywność (zwana główną) jest wybrana do wyświetlania przy pierwszym uruchomieniu aplikacji.
- ▶ Aktywność może przekazać sterowanie (i dane) do innej aktywności wykorzystując protokół komunikacyjny zwany intencją (intent).
- ▶ Aktywność zazwyczaj utożsamiana jest z pojedynczym ekranem GUI.

Przykład trzech aktywności

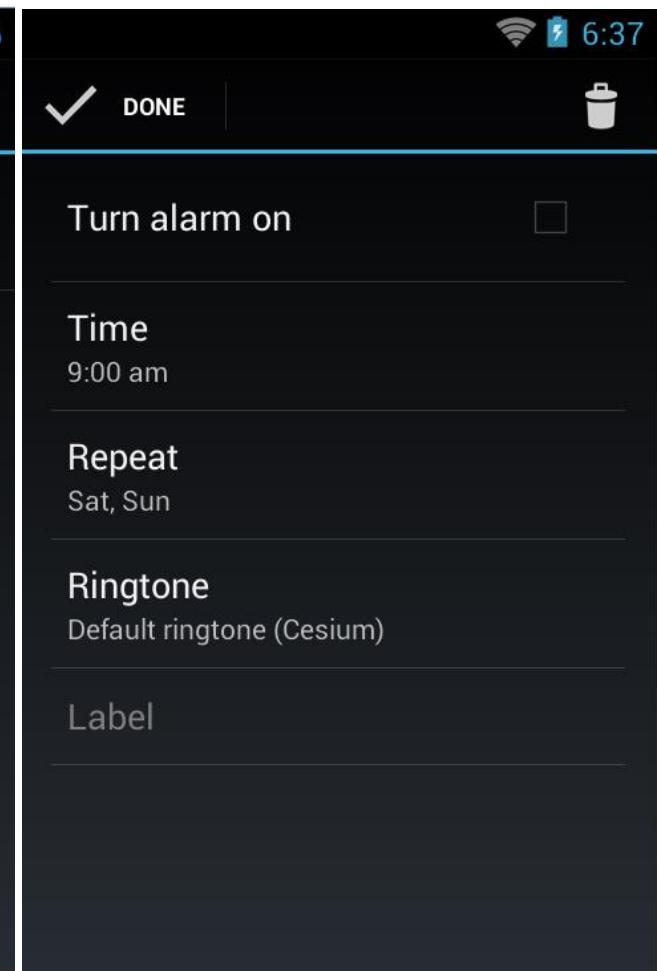
Aktywność 1



Aktywność 2



Aktywność 3



Usługa (Service)

- ▶ Usługa to specjalny typ aktywności, która nie posiada wizualnej reprezentacji
- ▶ Usługi zwykle uruchamiane są w tle na nieograniczony przedział czasu
- ▶ Aplikacje mogą uruchamiać własne usługi lub korzystać z już aktywnych
- ▶ Przykład:
Usługa GPS działa w tle i co jakiś czas wysyła dane lokalizacyjne do aplikacji nimi zainteresowanymi



Broadcast receiver

- ▶ Broadcast receiver to „słuchacz” oczekujący na globalnie (w obrębie całego systemu) przesyłane komunikaty.
- ▶ Nie posiada graficznego interfejsu.
- ▶ Zwykle broadcast receiver jest rejestrowany w systemie za pomocą filtra. Gdy komunikat zostanie dopasowany do danego filtra, BR jest aktywowany.
- ▶ Broadcast receiver może odpowiedzieć na komunikat wywołując określoną aktywność lub wyświetlając powiadomienie (notification).

Dostawca treści (content provider)

- ▶ Jego zadaniem jest udostępnianie zbiorów danych aplikacjom.
- ▶ Typowe zbiory danych (globalne): kontakty, zdjęcia, wiadomości, pliki audio itp.
- ▶ Globalne zbiory danych często przechowywane są w bazie SQLite.
- ▶ Dostawca treści jest warstwą abstrakcyjną – dostarcza spójne metody do pobierania, dodawania, modyfikacji i usuwania danych, bez względu na specyfikę zbioru danych.

Activity Stack

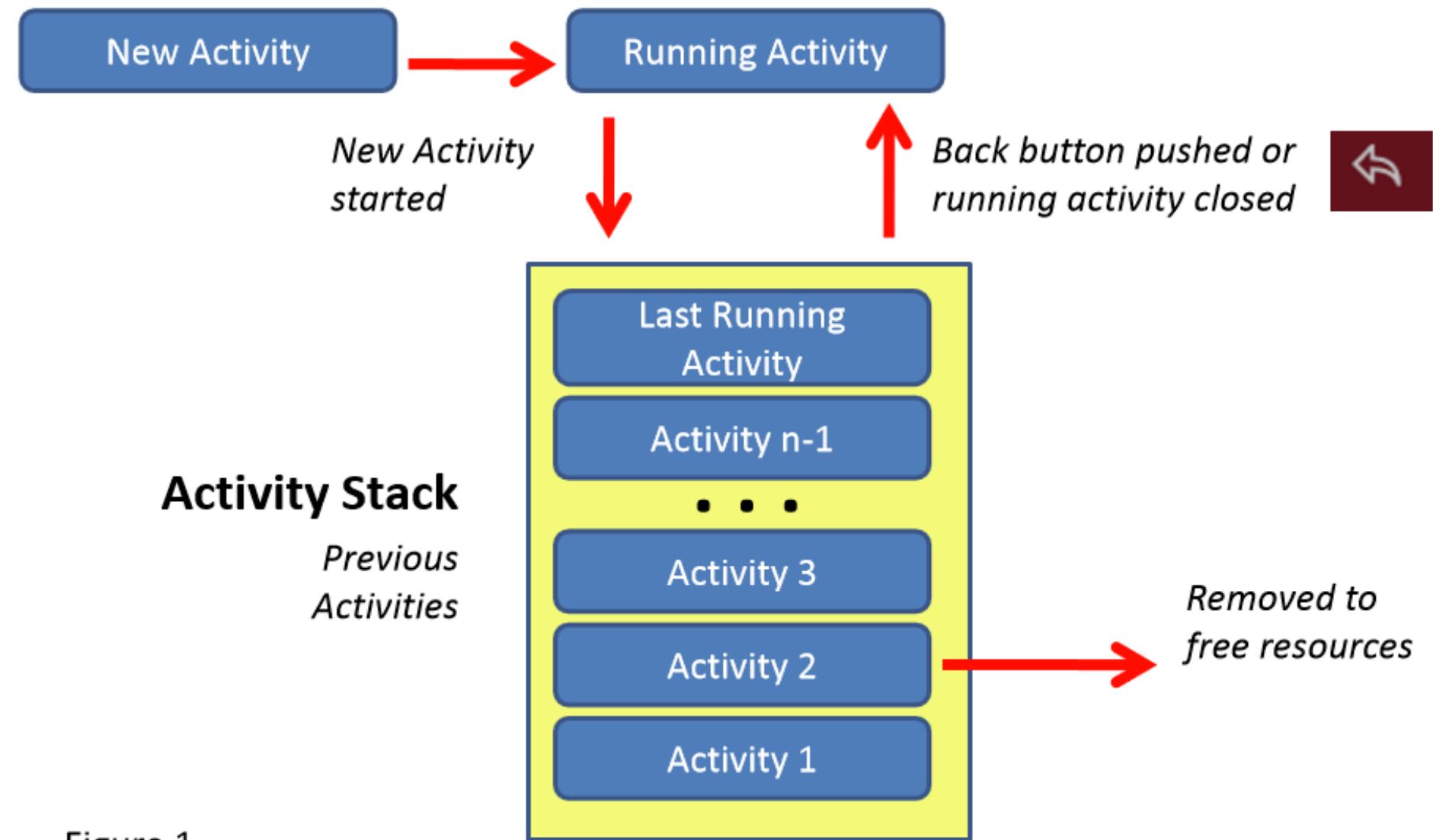
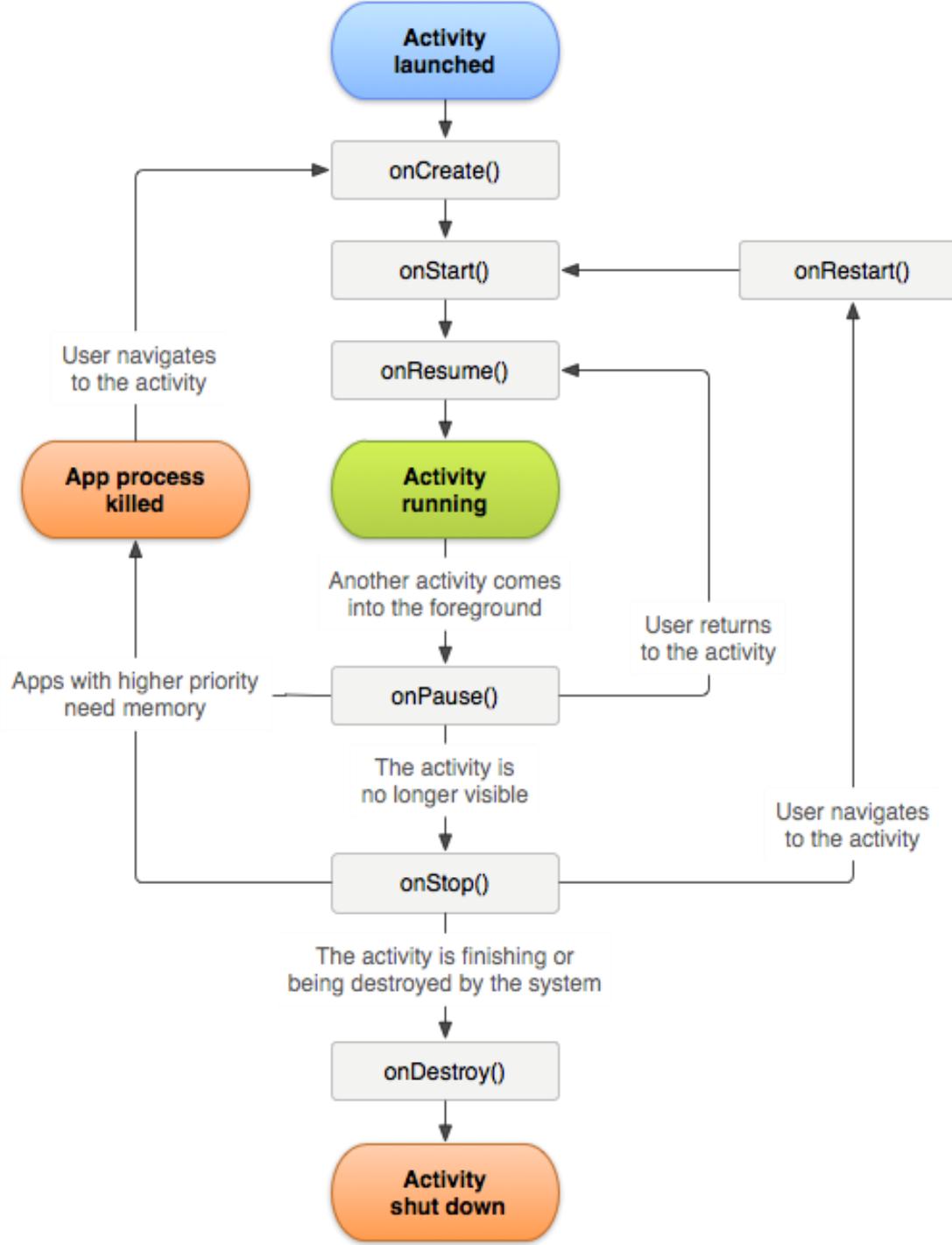


Figure 1

Cykl życia aktywności



Implementacja callbacks

```
public class PrzykladAktywnosci extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // Aktywność jest tworzona  
    }  
    @Override  
    protected void onStart() {  
        super.onStart();  
        // Aktywność będzie widoczna  
    }  
    @Override  
    protected void onResume() {  
        super.onResume();  
        // Aktywność jest widoczna (na pierwszym planie)  
    }  
    @Override  
    protected void onPause() {  
        super.onPause();  
        // Inna aktywność posiada focus  
    }  
    @Override  
    protected void onStop() {  
        super.onStop();  
        // Aktywność nie jest już widoczna  
    }  
    @Override  
    protected void onDestroy() {  
        super.onDestroy();  
        // Aktywność zostanie usunięta  
    }  
}
```

Zakończenie cyklu życia

- ▶ Metody **onPause()**, **onStop()**, **onDestroy()** mają status *killable*, czyli po zakończeniu dowolnej z nich, pozostałe nie muszą zostać wywołane (jeśli system wymusi zakończenie aplikacji).
- ▶ **onPause()** to jedyna metoda, która na pewno będzie wywołana przed zakończeniem aplikacji.
- ▶ Metoda **onPause()** powinna być wykorzystywana do zapisania stanu aplikacji.

Badanie cyklu życia aktywności - przykład

```
package pl.tomaszx.helloworld;

import android.os.Bundle;

public class MainActivity extends Activity {
    private Context context;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btnExit = (Button) findViewById(R.id.button1);

        btnExit.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });

        context = getApplicationContext();
        Toast.makeText(context, "onCreate", Toast.LENGTH_SHORT).show();
    }
}
```

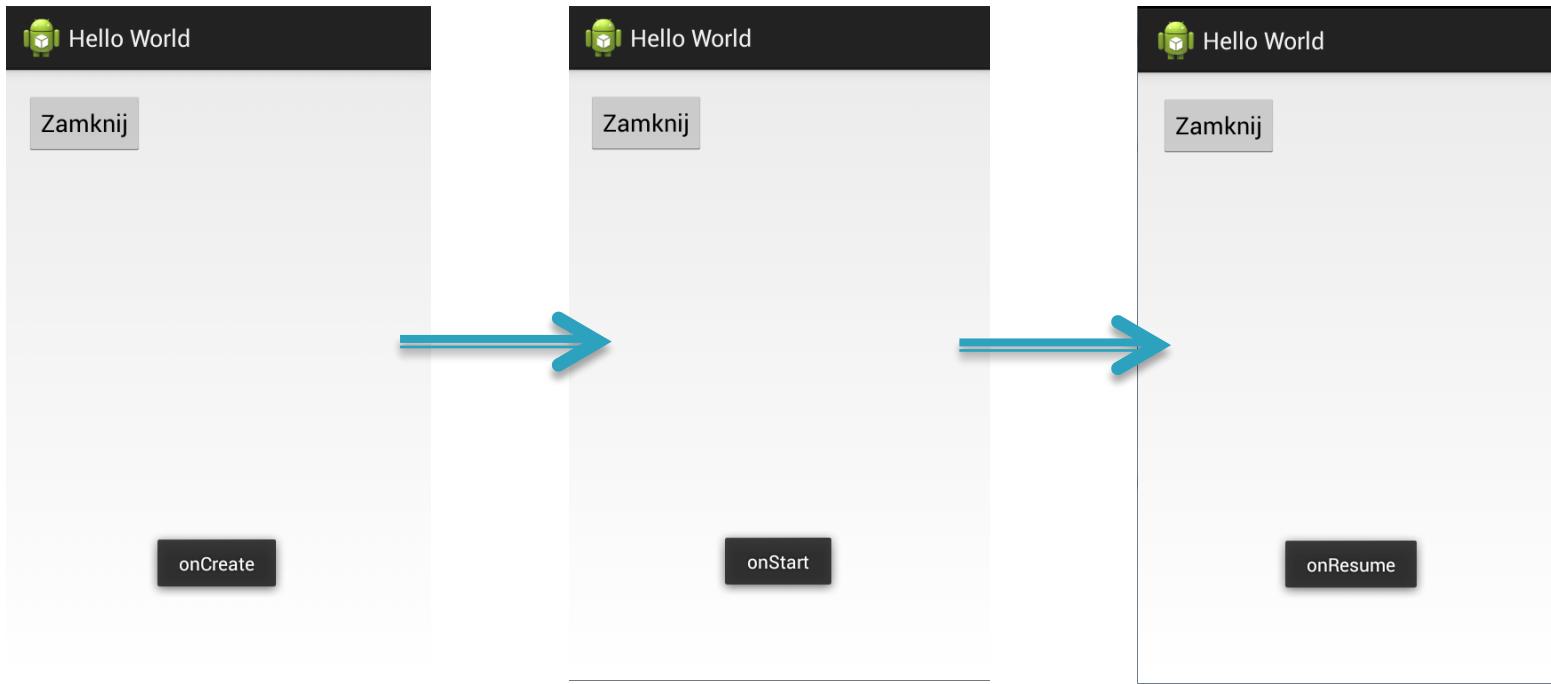
Badanie cyklu życia aktywności – przykład c.d.

```
⊕ @Override  
protected void onDestroy() {  
    super.onDestroy();  
    Toast.makeText(context, "onDestroy", Toast.LENGTH_SHORT).show();  
}  
  
⊕ @Override  
protected void onPause() {  
    super.onPause();  
    Toast.makeText(context, "onPause", Toast.LENGTH_SHORT).show();  
}  
  
⊕ @Override  
protected void onRestart() {  
    super.onRestart();  
    Toast.makeText(context, "onRestart", Toast.LENGTH_SHORT).show();  
}  
  
⊕ @Override  
protected void onResume() {  
    super.onResume();  
    Toast.makeText(context, "onResume", Toast.LENGTH_SHORT).show();  
}
```

Badanie cyklu życia aktywności – przykład c.d.

```
@Override  
protected void onStart() {  
    super.onStart();  
    Toast.makeText(context, "onStart", Toast.LENGTH_SHORT).show();  
}
```

```
@Override  
protected void onStop() {  
    super.onStop();  
    Toast.makeText(context, "onStop", Toast.LENGTH_SHORT).show();  
}
```



Android Studio Demo

SQLite1 - [C:\Users\Tomek\Downloads\SQLite1] - [app] - ...\\app\\src\\main\\AndroidManifest.xml - Android Studio 2.2

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

SQLlite1 app src main AndroidManifest.xml

Project Structure Captures Build Variants Favorites

Gradle Model

Text Merged Manifest

TODO Android Monitor Terminal Messages

Event Log Gradle Console

Gradle build finished in 6s 281ms (8 minutes ago)

1:1 CRLF+ UTF-8 Context: <no context>

The screenshot shows the Android Studio interface with the following details:

- Project Tree:** Shows the project structure under the "app" module, including "manifests", "java", "res", and "Gradle Scripts".
- Editor:** Displays the content of the "AndroidManifest.xml" file. The XML code defines the application's manifest, specifying activities, their labels, themes, and intent filters.
- Toolbars and Menus:** Standard Android Studio menus like File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help are visible at the top.
- Bottom Bar:** Includes tabs for TODO, Android Monitor, Terminal, Messages, Event Log, and Gradle Console, along with status information about the build.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.pietrzyk.sqlite1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="SQLite1"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="SQLite1"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".RefuelNewActivity"
            android:label="">
            //brak tytułu na toolbarze
        </activity>
        <activity android:name=".RefuelListActivity" />
        <activity
            android:name=".RefuelUpdateActivity"
            android:label="UpdateRefuelActivity"
            android:theme="@style/AppTheme" />
    </application>
</manifest>
```

Pytania? Problemy?

Dziękuję za uwagę!

Programowanie urządzeń mobilnych

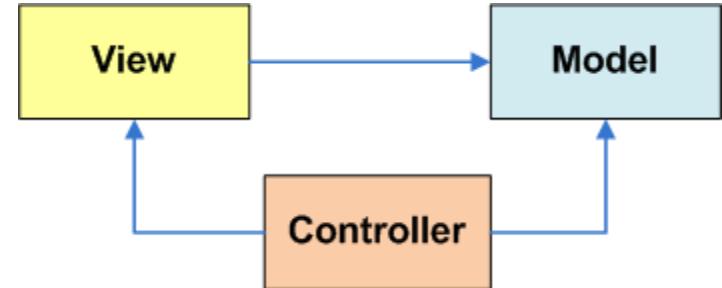
Graficzny interfejs użytkownika

Tłumaczenie i adaptacja materiałów: dr Tomasz Xięski.
Na podstawie prezentacji udostępnionych przez Victor Matos, Cleveland State University.

Portions of this page are reproduced from work created and [shared by Google](#) and used according to terms

Wzorzec MVC

Wzorzec Model-Widok-Kontroler (**MVC**) jest istotnym wzorcem projektowym, którego głównym zadaniem jest odseparowanie (1) interfejsu użytkownika, (2) biznesowej oraz (3) operacyjnej logiki.



Jak to wygląda z punktu widzenia programisty Android?

- **Model.** Składa się z kodu w języku JAVA i odwołań API, które zarządzają zachowaniem danych aplikacji.
- **Widok.** Zestaw ekranów, które użytkownik widzi i wchodzi w interakcję.
- **Kontroler.** Implementowany m. in. poprzez system operacyjny, odpowiedzialny za interpretację zdarzeń użytkownika i systemowych. Zdarzenia mogą pochodzić z różnorakich źródeł: trackball, klawiatura, ekran dotykowy, moduł GPS, usługi działające w tle. Nakazuje modelowi i/lub widokowi (zwykle przez wywołania zwrotne i nasłuchiwaczy) by dokonały odpowiednich zmian.

[Burbeck92] Burbeck, Steve. "Application Programming in Smalltalk-80: How to use Model-View-Controller (MVC)." *University of Illinois in Urbana-Champaign (UIUC) Smalltalk Archive*. Available at: <http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html>.

Wzorzec MVC c. d.

Graficzne interfejsy użytkownika zwykle implementowane są jako pliki XML (aczkolwiek mogą one być tworzone również w sposób dynamiczny poprzez kod języka Java).

Zachowanie kontrolera:

Bezpośrednia interakcja

Gdy użytkownik dotknie określonego miejsca na ekranie, kontroler dokonuje interpretacji tego zdarzenia i określa, jaki dokładnie fragment ekranu oraz gest miał miejsce. Na podstawie tej informacji, przekazywane jest do modelu wywołanie określonej funkcji (callback) lub konieczność zmiany stanu aplikacji.

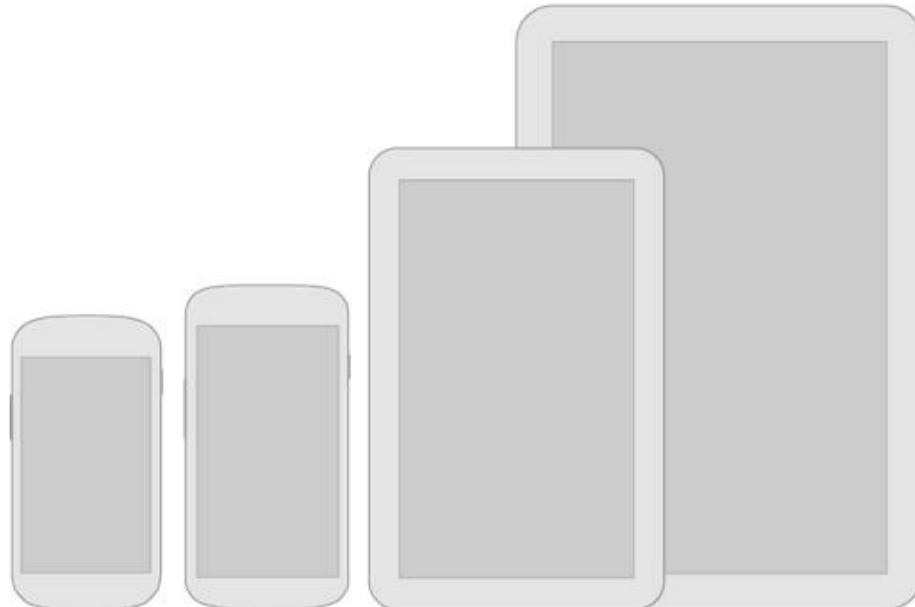
Niejawna interakcja

Usługa działająca w tle, może bez ingerowania użytkownika powiadomić kontroler odnośnie zmian stanu (np. osiągnięto miejsce docelowe w nawigacji), co w konsekwencji może spowodować zmianę widoku.

Wzorce projektowe GUI

Devices and Displays

Android powers more than a billion phones, tablets, and other devices in a wide variety of screen sizes and form factors. By taking advantage of Android's flexible layout system, you can create apps that gracefully scale from large tablets to smaller phones.



Be flexible

Stretch and compress your layouts to accommodate various heights and widths.

Optimize layouts

On larger devices, take advantage of extra screen real estate. Create compound views that combine multiple views to reveal more content and ease navigation.

Assets for all

Provide resources for different screen densities ([DPI](#)) to ensure that your app looks great on any device.

1x

1.5x

2x

3x

4x



<https://developer.android.com/design/patterns/index.html>

Klasa View

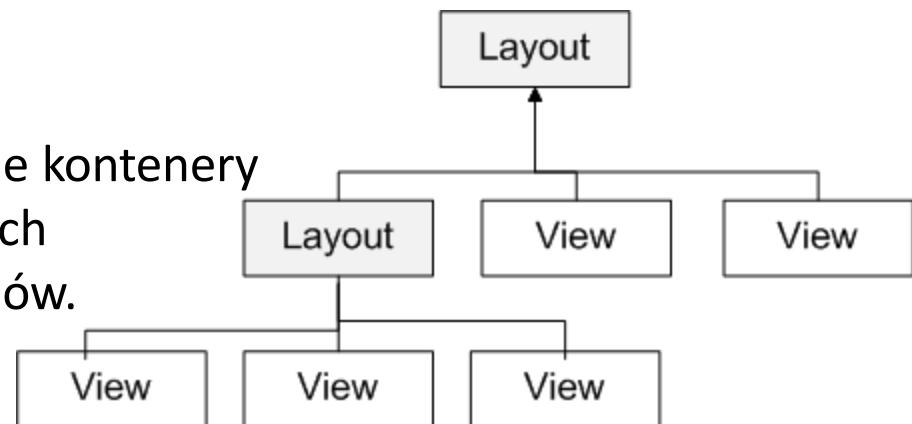
- Klasa **View** reprezentuje podstawowy komponent Androida za pomocą którego można tworzyć graficzne interfejsy użytkownika. Stanowi kontener dla wszystkich elementów, które można wyświetlić.



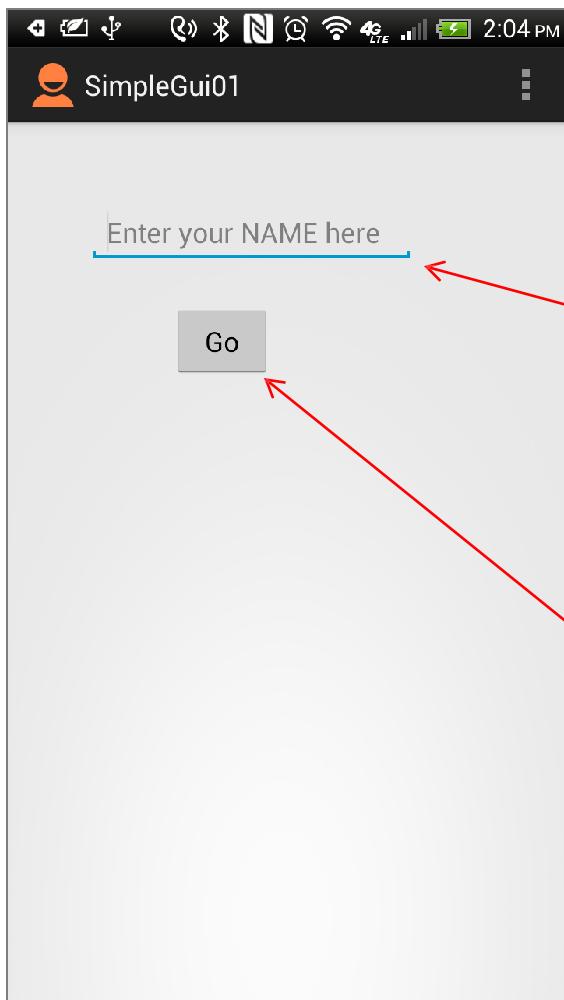
- **Widok** zajmuje prostokątną przestrzeń na ekranie i jest odpowiedzialny za *rysowanie* jak również *przetwarzanie zdarzeń*.

- **Widżety (ang. widgets)** są potomkami klasy View. Używane są do stworzenia interaktywnych komponentów graficznych takich jak przyciski, etykiety, pola tekstowe itp.

- **Układy (ang. layouts)** to niewidzialne kontenery umożliwiające pozycjonowanie innych widoków oraz zagnieżdżonych układów.



Interfejs GUI ↔ Układ 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/editText1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_alignParentLeft="true"  
        android:layout_alignParentTop="true"  
        android:layout_marginLeft="35dp"  
        android:layout_marginTop="35dp"  
        android:ems="10"  
        android:hint="Enter your NAME here" />  
  
    <Button  
        android:id="@+id/button1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_alignLeft="@+id/editText1"  
        android:layout_below="@+id/editText1"  
        android:layout_marginLeft="54dp"  
        android:layout_marginTop="26dp"  
        android:text="Go" />  
  
</RelativeLayout>
```

Rzeczywisty interfejs aplikacji

Wersja tekstowa: *activity_main.xml*



Wykorzystanie Widoków

- Plik **XML** z widokiem składa się z układem (**layout**) tworzącym hierarchiczną strukturę zawartych w nim elementów.
 - Wewnętrzne elementy mogą być zwykłymi widżetami bądź zagnieżdzonymi widokami zawierające skomplikowane hierarchie elementów.
 - Aktywność używa **setContentView(R.layout.xmlfilename)** by wyświetlić widok na ekranie urządzenia.

```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:orientation="horizontal" >  
  
    } } Widżety i zagnieżdżone układy  
    ↗
```

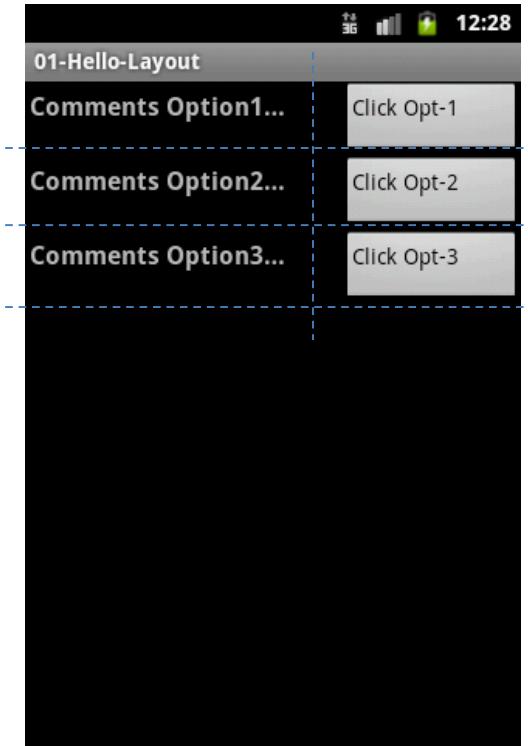
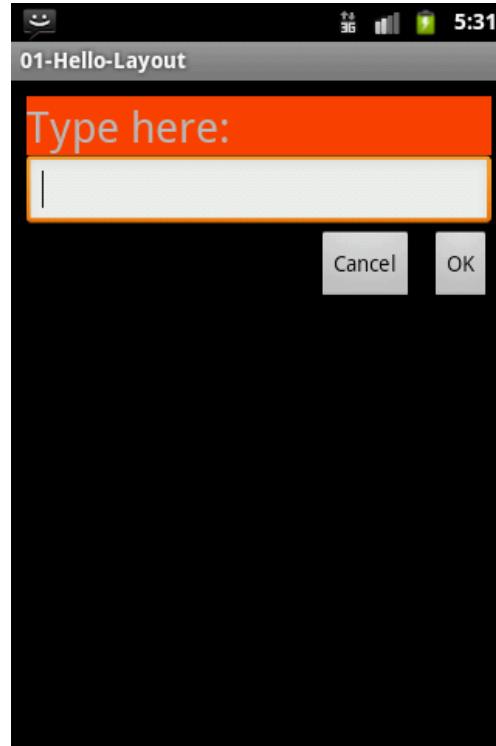
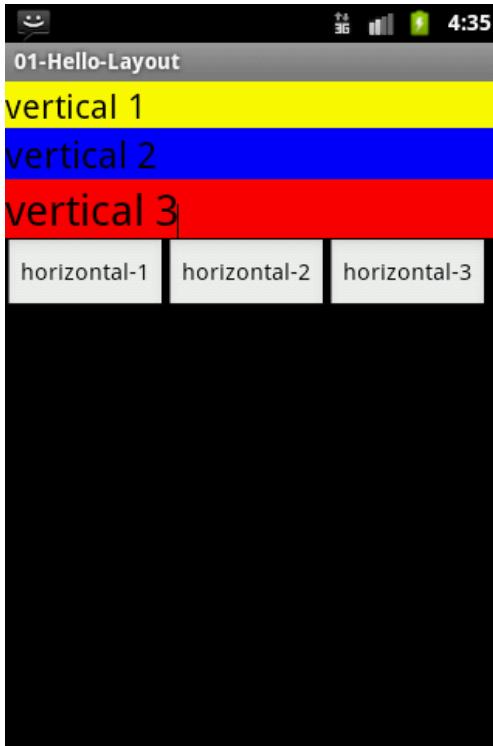
Wykorzystanie widoków

Wykorzystanie widoków i układów z reguły składa się z następującego szeregu czynności:

1. **Ustaw właściwości:** Przykładowo kolor tła, tekstu, czcionki i wielkości komponentów.
2. **Ustaw nasłuchiwacze (ang. listeners):** Przykładowo obraz może być skonfigurowany by reagował na zdarzenia kliknięcia, dłuższego przytrzymania palca itp.
3. **Ustaw focus:** By ustawić focus na określonym komponencie należy użyć metody `requestFocus()` lub znacznika XML `<requestFocus/>`
4. **Ustaw widoczność:** Można pokazywać lub ukrywać elementy wykorzystując metodę `setVisibility(...)`.

Przykładowe komponenty GUI

Układy



Linear Layout

LinearLayout
rozmieszcza widoki
horyzontalnie lub
wertykalnie.

Relative Layout

RelativeLayout jest grupą
elementów, która umożliwia
rozmieszczenie widoków w
sposób względny.

Reference: <http://developer.android.com/guide/topics/ui/layout-objects.html>

Table Layout

TableLayout jest grupą
elementów, która
rozmieszcza elementy
w wierszu bądź
kolumnie wirtualnej
tabeli.

Przykładowe komponenty GUI

Widżety

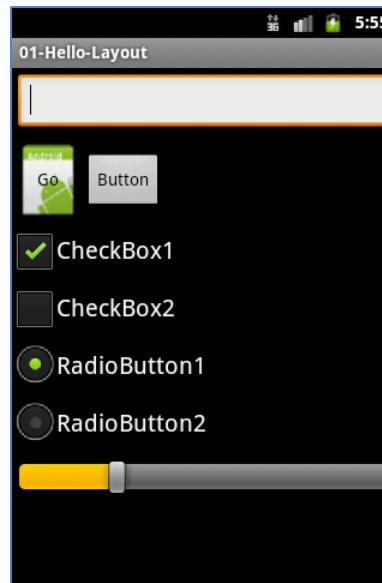


TimePicker

AnalogClock

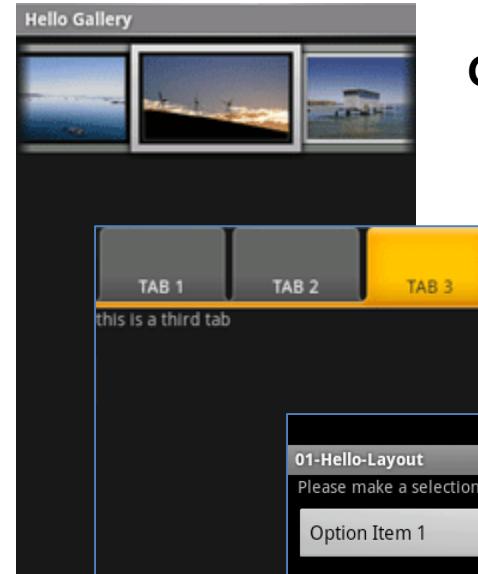
DatePicker

DatePicker jest komponentem umożliwiającym wybór miesiąca, dnia i roku.



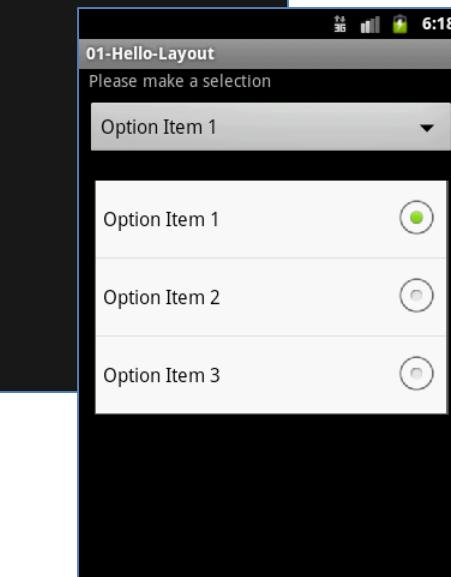
Kontrolki formularza

Grupa komponentów do wpisania danych: *przyciski z grafiką, pola jednokrotnego i wielokrotnego wyboru itp.*



GalleryView

TabWidget



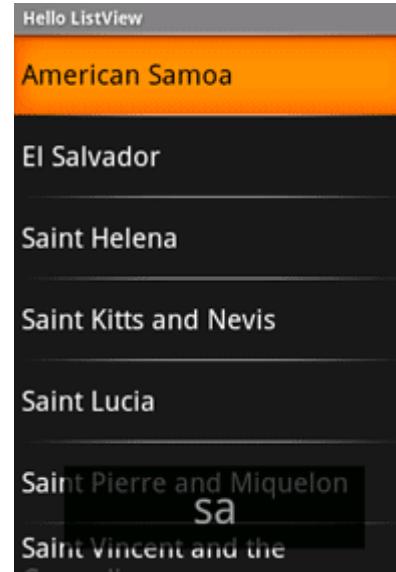
Spinner

Przykładowe komponenty GUI



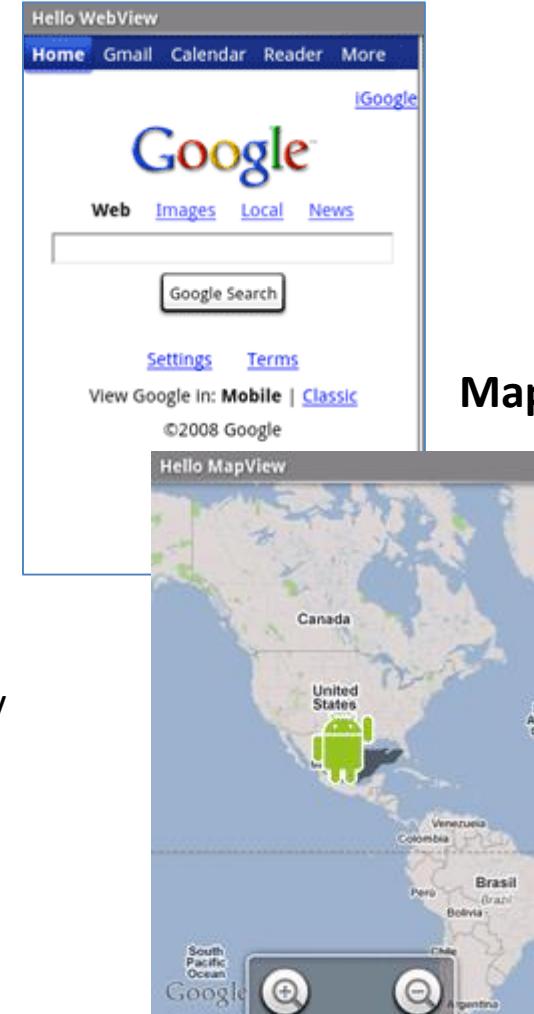
AutoCompleteTextView

Jest specyficką wersją pola tekstowego (*EditText*), który pokazuje sugestie użytkownikowi podczas pisania. Sugestie pochodzą z kolekcji typu tablica ciągów.



ListView

ListView jest widokiem który pokazuje dane w formie pionowej, przewijalnej listy. Dane pochodząc z obiektu typu *ListAdapter*.

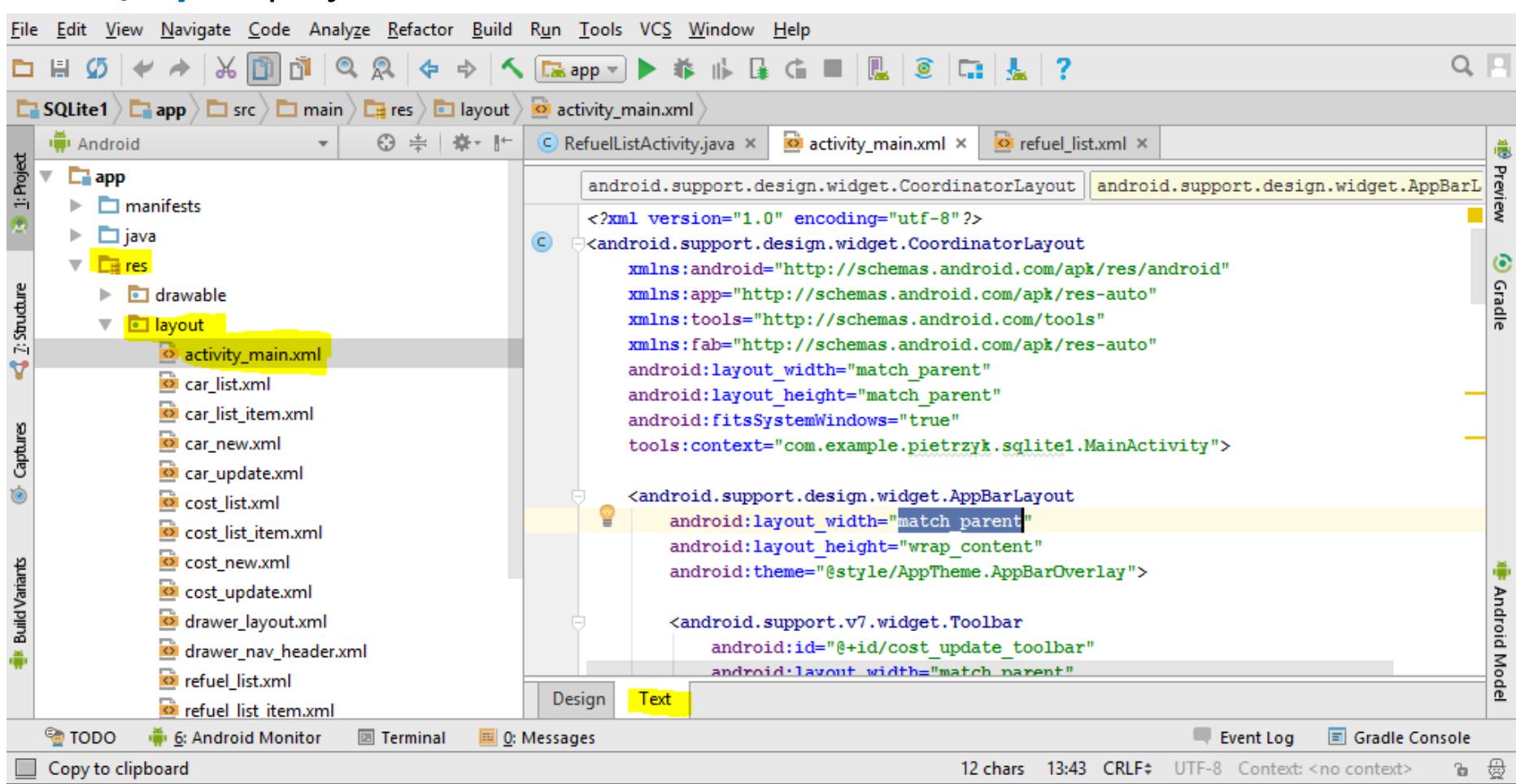


WebView

MapView

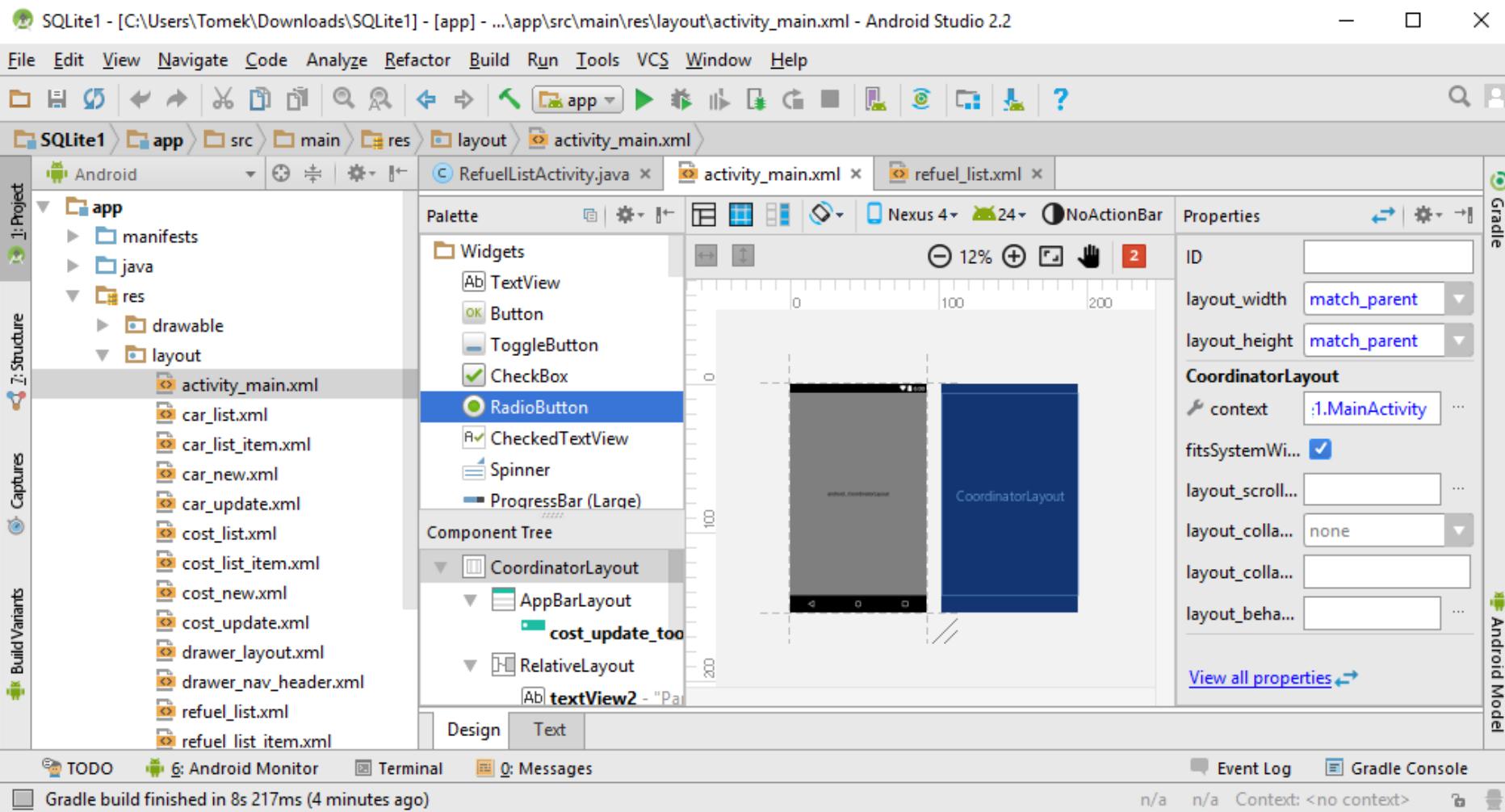
Układy XML w Android Studio

W kontekście platformy Android, pliki XML wchodzą w skład **zasobów (ang. resources)**, więc pliki z interfejsem graficznym znajdują się w katalogu **res/layout** projektu.



Układy XML w Android Studio

Android Studio zawiera również edytor typu WYSIWIG, który umożliwia tworzenie interfejsów metodą przeciągij-i-upuść.



Jak tworzyć GUI w Androidzie

- Układy w Androidzie są kontenerami GUI, posiadające ścisłe zdefiniowaną strukturę oraz właściwości dotyczące rozłożenia elementów.
- **Układy mogą być zagnieżdżane**, dlatego komórka, wiersza albo kolumna może być początkiem dla innego widoku bądź układu.
- W Android Studio dostępne są następujące typy układów (co ewoluje z każdą wersją Androida):

Layouts

 ConstraintLayout

 GridLayout

 FrameLayout

 LinearLayout (horizontal)

 LinearLayout (vertical)

 RelativeLayout

 TableLayout

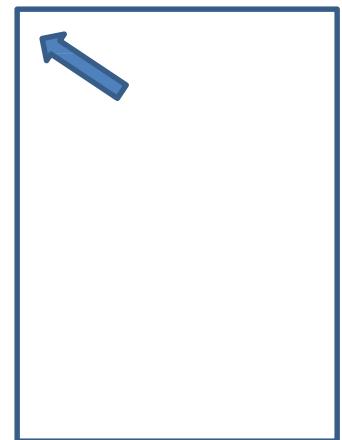
 TableRow

 <fragment>

Podstawowe układy

FrameLayout

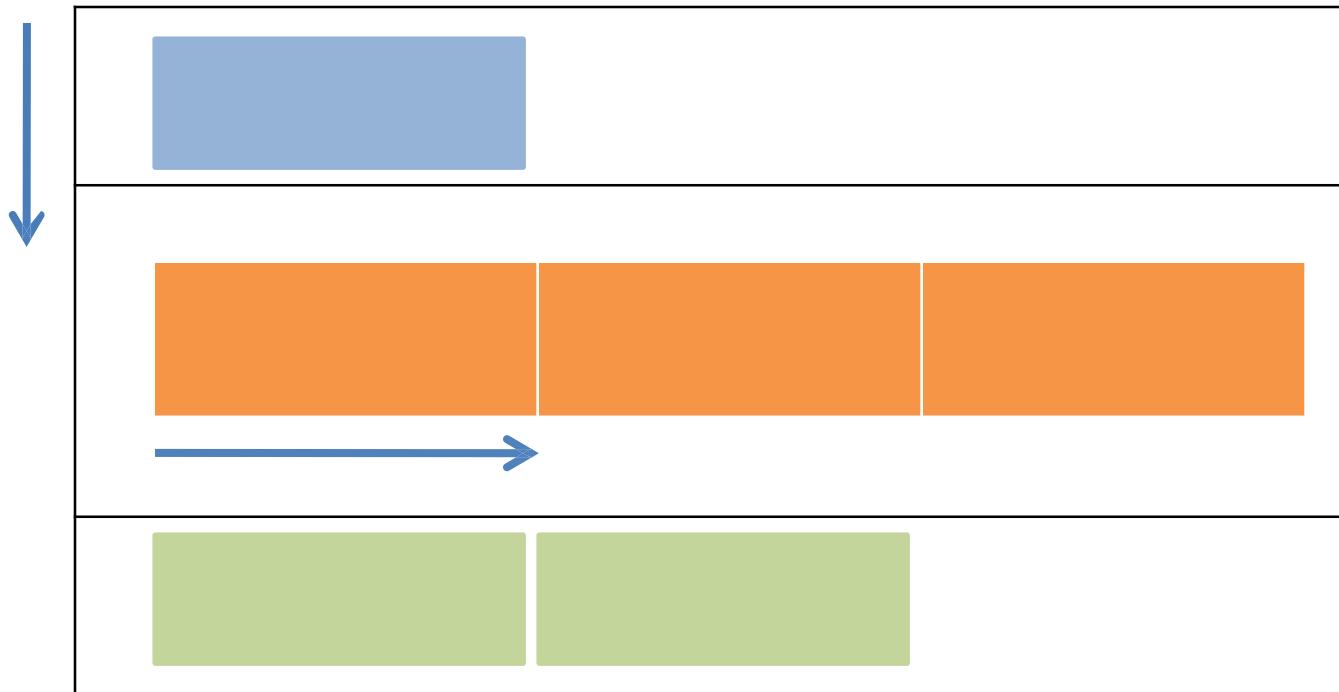
- FrameLayout jest najprostszym typem układu.
- Najczęściej wykorzystywany jest jako najbardziej zewnętrzny układ.
- Umożliwia zdefiniowanie jak duża część ekranu (wysokość, szerokość) ma zostać wykorzystana.
- Wszystkie jego elementy potomne pozycjonowane są względem górnego lewego końca ekranu.



Linear Layout

1. Linear Layout

- **LinearLayout** wspiera strategię wypełnienia według której elementy są ułożone w sposób **horyzontalny** lub **wertykalny**.
- Jeżeli orientacja układu ustawiona jest na pionową nowe wiersze (widoki) układane są jeden na drugim.
- Orientacja horyzontalna wykorzystuje rozmieszczenie jeden obok drugiego.



Linear Layout

1. LinearLayout: Ustawianie właściwości

Konfiguracja **LinearLayout** sprowadza się do ustawienia następujących właściwości:

- orientation *(vertical, horizontal)*
- fill model *(match_parent, wrap_contents)*
- weight *(0, 1, 2, ...n)*
- gravity *(top, bottom, center,...)*
- padding *(dp – dev. independent pixels)*
- margin *(dp – dev. independent pixels)*

LinearLayout - Orientation

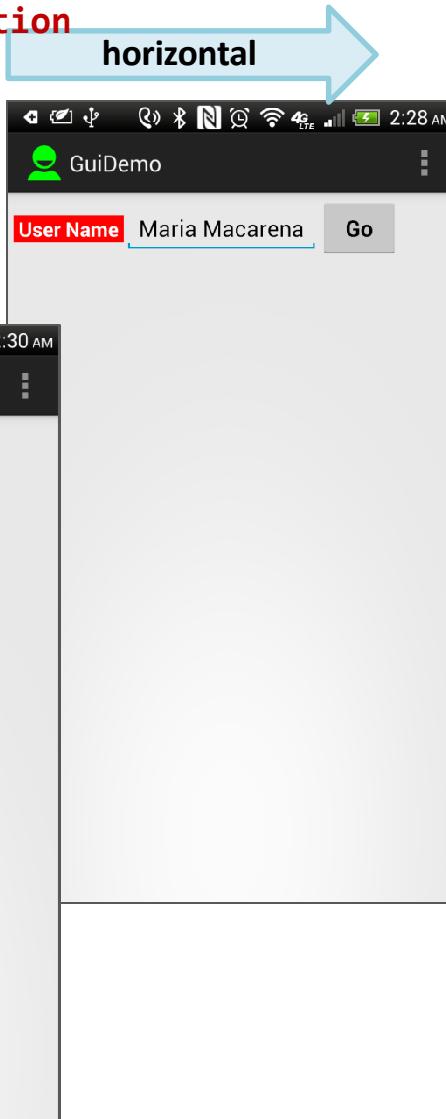
1.1 Atrybut Orientation

Właściwość **android:orientation**

przyjmuje wartości:

horizontal lub **vertical**.

Można też wywołać
`setOrientation()` z
poziomu kodu,



```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myLinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:padding="4dp" >

    <TextView
        android:id="@+id/labelUserName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#ffff0000"
        android:text=" User Name "
        android:textColor="#ffffffff"
        android:textSize="16sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/ediName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Maria Macarena"
        android:textSize="18sp" />

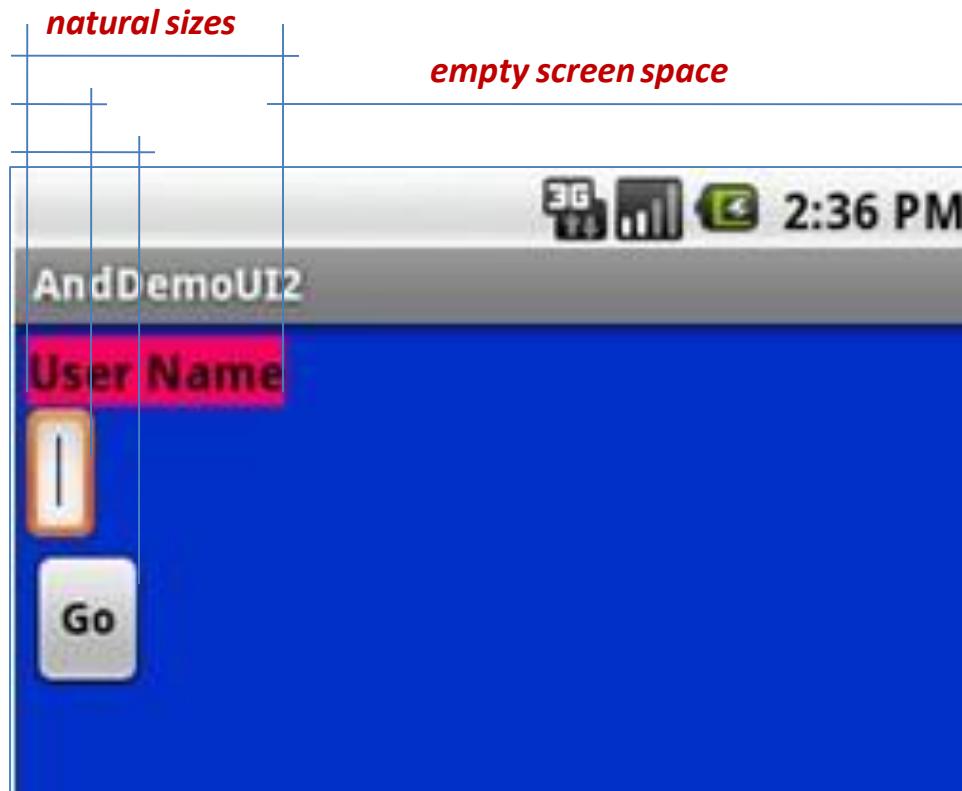
    <Button
        android:id="@+id/btnGo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go"
        android:textStyle="bold" />

</LinearLayout>
```

LinearLayout – Fill Model

1.2 Fill Model

- Widżety posiadają "naturalny rozmiar" określony na podstawie ich zawartości.
- W pewnych okolicznościach widżet powinien posiadać określony rozmiar (szerokość, wysokość) nawet jeżeli nie wprowadzono żadnego tekstu (tak jak na poniższym rysunku).



LinearLayout – Fill Model

1.2 Fill Model

Wszystkie widżety wewnątrz LinearLayout **muszą** posiadać nadane atrybuty ‘width’ i ‘height’.

android:layout_width
android:layout_height

Wartości używane do określenia szerokości czy wysokości to:

1. Określony rozmiar jak **125dp** (device independent pixels, a.k.a. **dip**)
2. **wrap_content** wskazuje, że widżet zachowuje swój naturalny rozmiar.
3. **match_parent** (kiedyś ‘**fill_parent**’) wskazuje, że widżet powinien być tak samo wielki jak jego rodzic

LinearLayout – Fill Model

1.2 Fill Model



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myLinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff0033cc"
    android:orientation="vertical"
    android:padding="4dp" >

    <TextView
        android:layout_width="_parent"
        android:layout_height="wrap_content"
        android:background="#ffff0066"
        android:text="User Name"
        android:textColor="#ff000000"
        android:textSize="16sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/ediName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp" />

    <Button
        android:id="@+id/btnGo"
        android:layout_width="125dp"
        android:layout_height="wrap_content"
        android:text="Go"
        android:textStyle="bold" />

</LinearLayout>
```

Row-wise

Use all the row

Specific size: 125dp

LinearLayout – Weight

1.2 Weight

Określa ile miejsca w LinearLayout powinien mieć zaalokowany widok. Należy użyć **0** jeżeli widok nie ma się rozciągać. Im większa waga, tym więcej miejsca posiada dany widżet na swoim poziomie hierarchii.

Przykład

Specyfikacja XML okna jest podobna jak w poprzednim przykładzie.

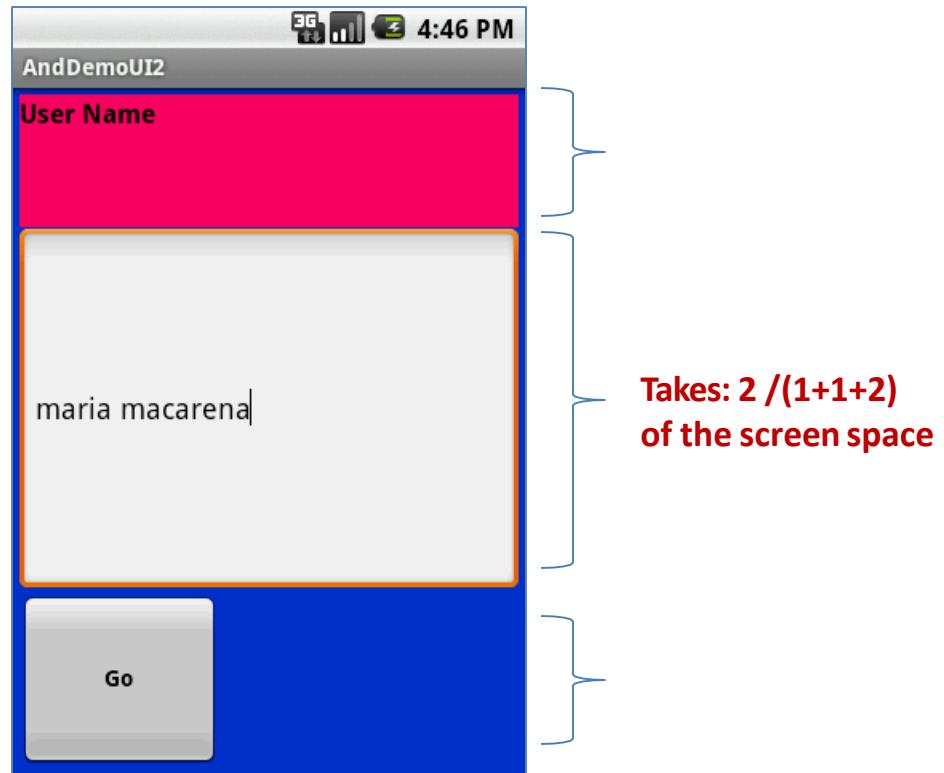
Kontroki TextView oraz Button mają dodatkowo właściwość:

`android:layout_weight="1"`

Gdzie komponent EditText ma

`android:layout_weight="2"`

Domyślna wartość to 0



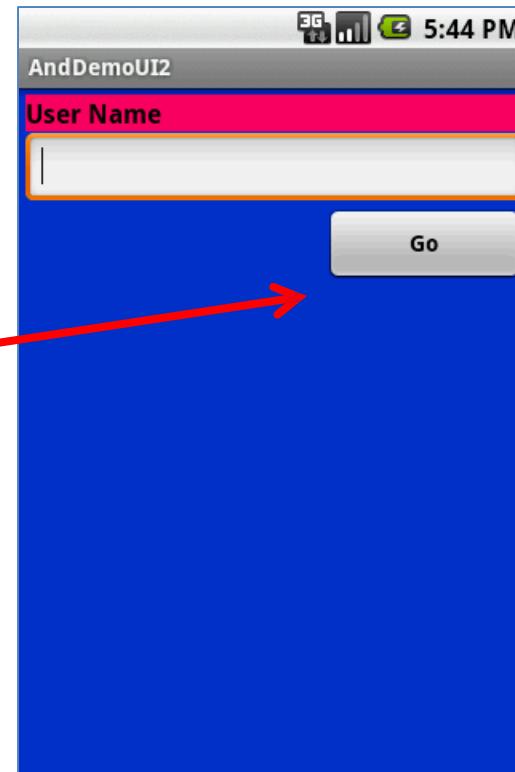
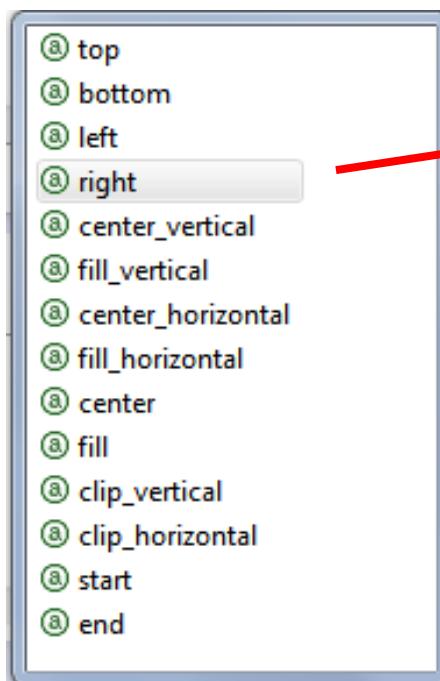
LinearLayout – Gravity

1.3 Layout_Gravity

- Używana by wskazać, jak wyrównane są elementy.
- Zwykle widżety pozycjonowane są do *lewej, górnej strony*.
- Wykorzystując w pliku XML właściwość

`android:layout_gravity="..."`

można ustawić inne wyrównanie:
left, center, right, top, bottom, itp.



Button has
right
layout_gravity

The LinearLayout – Gravity

1.3

UWAGA: gravity vs. layout_gravity



Różnica względem:

android:gravity

Wskazuje jak zachowuje się obiekt wewnątrz kontenera. Tutaj tekst jest wycentrowany

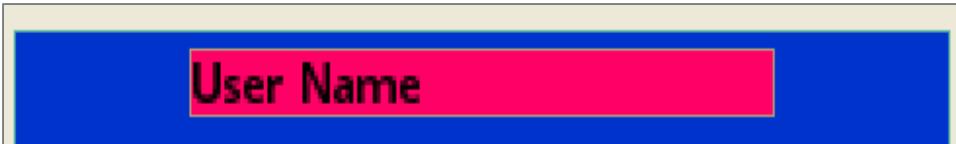
android:gravity="center"



android:layout_gravity

Pozycjonuje widżet względem rodzica:

android:layout_gravity="center"



LinearLayout – Padding

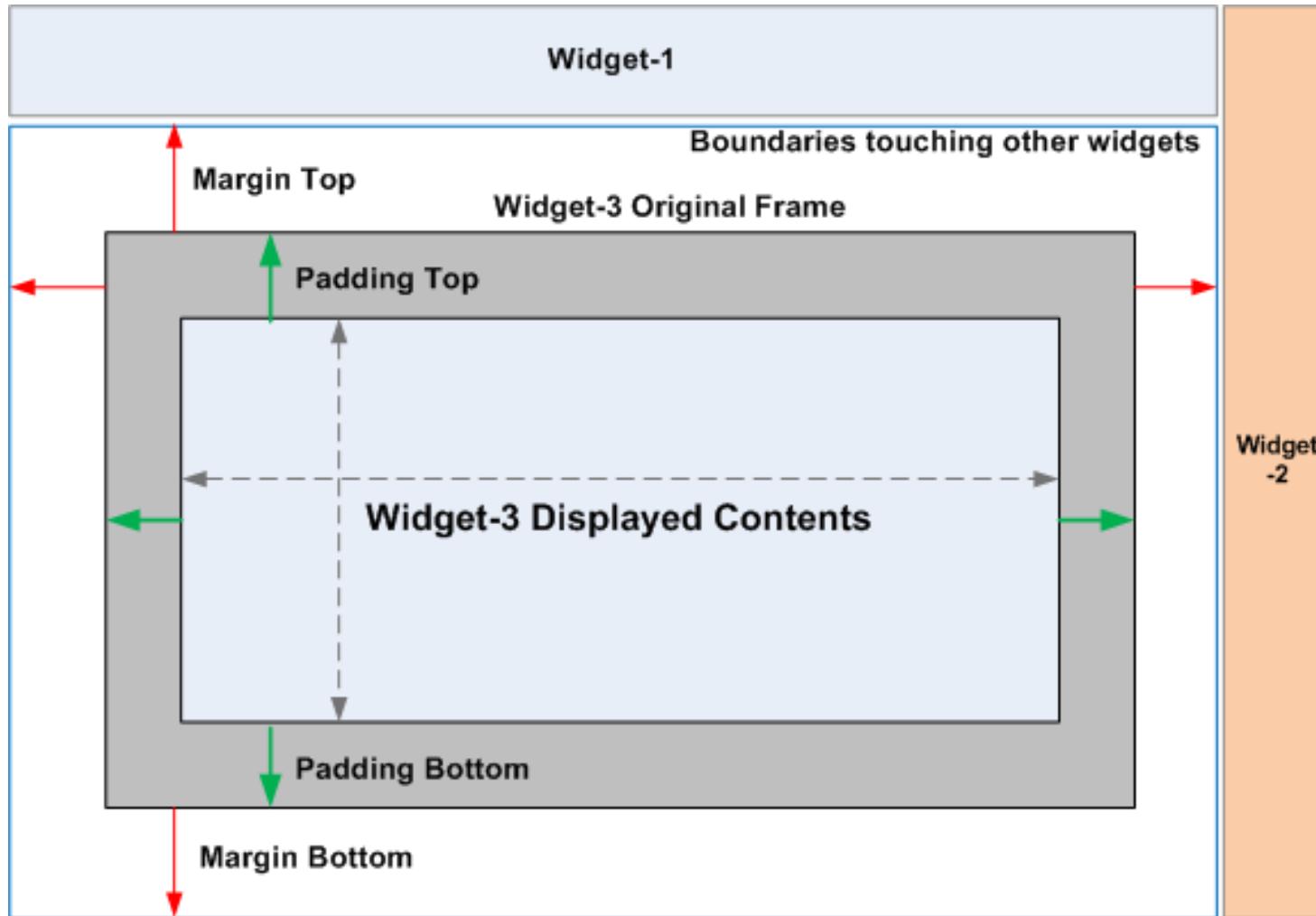
1.4 Padding

- Atrybut **padding** określa wewnętrzny margines widżetu (w **dp**).
- Wewnętrzny margines dodaje dodatkowe miejsce między obramowaniem widżetu a jego faktyczną zawartością.
- Używaj
 - **android:padding** (właściwości)
 - lub metody **setPadding()** z poziomu kodu

LinearLayout – Padding

1.3 Padding i Margin

Różnice między wewnętrznym i zewnętrznym marginesem

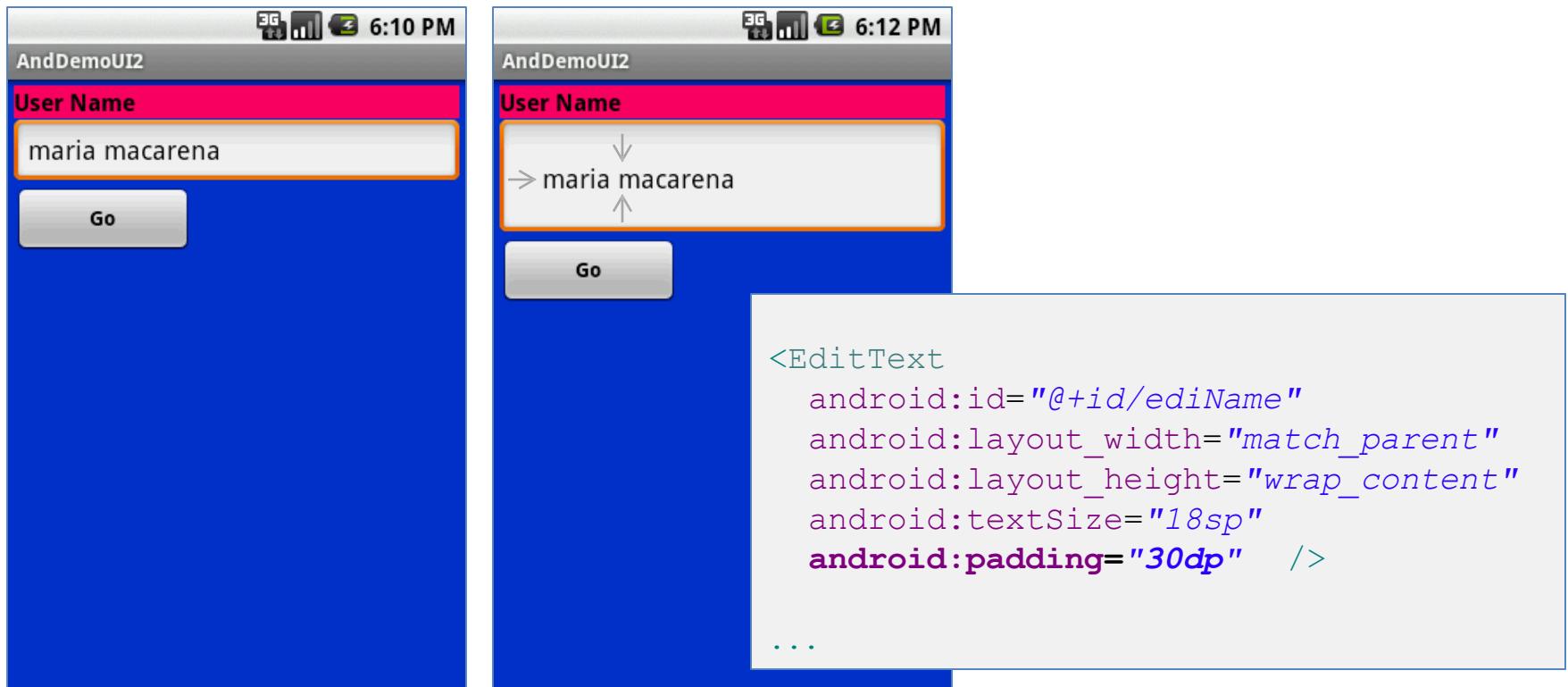


LinearLayout – Padding

1.3 Wewnętrzny margines używając padding

Przykład:

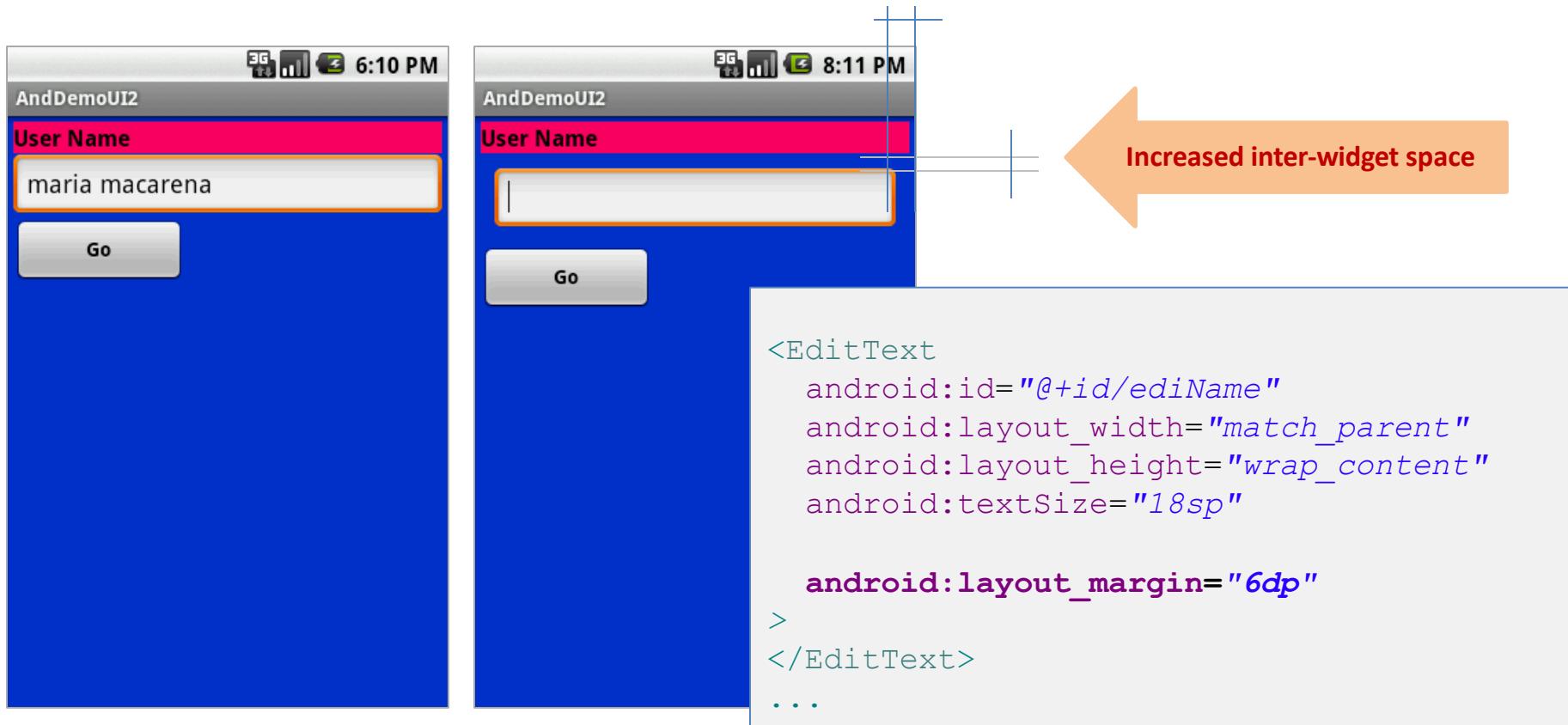
Komponentowi EditText nadano margines o wielkości 30dp



LinearLayout – Margin

1.4 Zewnętrzny margines

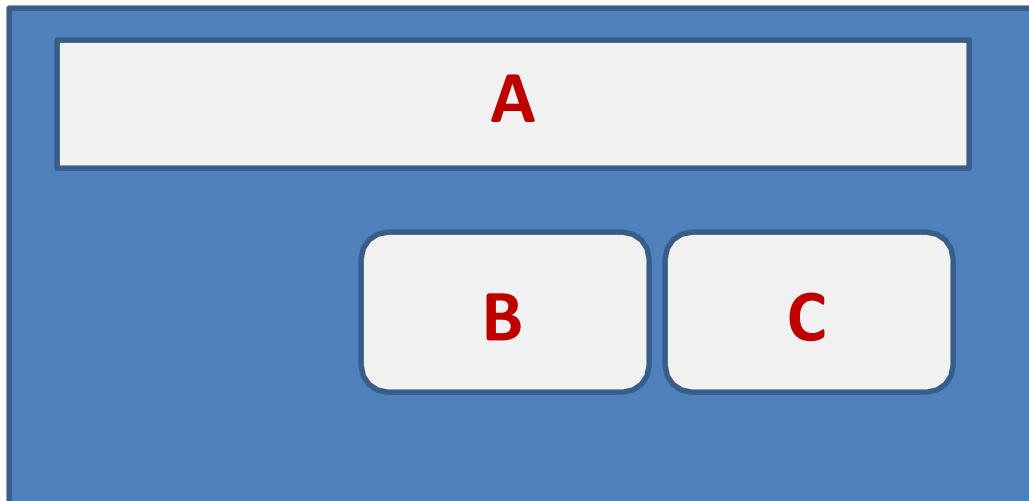
- Widżety domyślnie pozycjonowane są tuż obok siebie.
- Atrybut **android:layout_margin** umożliwia zwiększenie tego miejsca



Relative Layout

2. Relative Layout

Rozmieszczenie widżetów w **RelativeLayout** uwzględnia relację sąsiedztwa do innych *widżetów w kontenerze* oraz w przodku danej hierarchii.



Przykład:

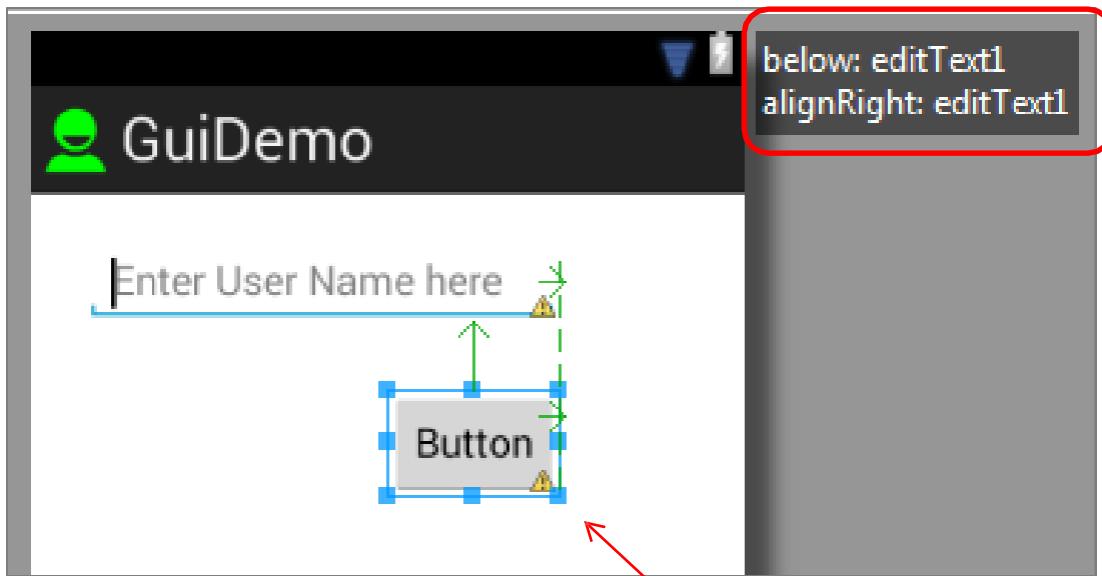
A jest w górnej części rodzica

C jest pod A, po jego prawej stronie

B jest pod A, na lewo od C

Relative Layout

2. Przykład: Relative Layout

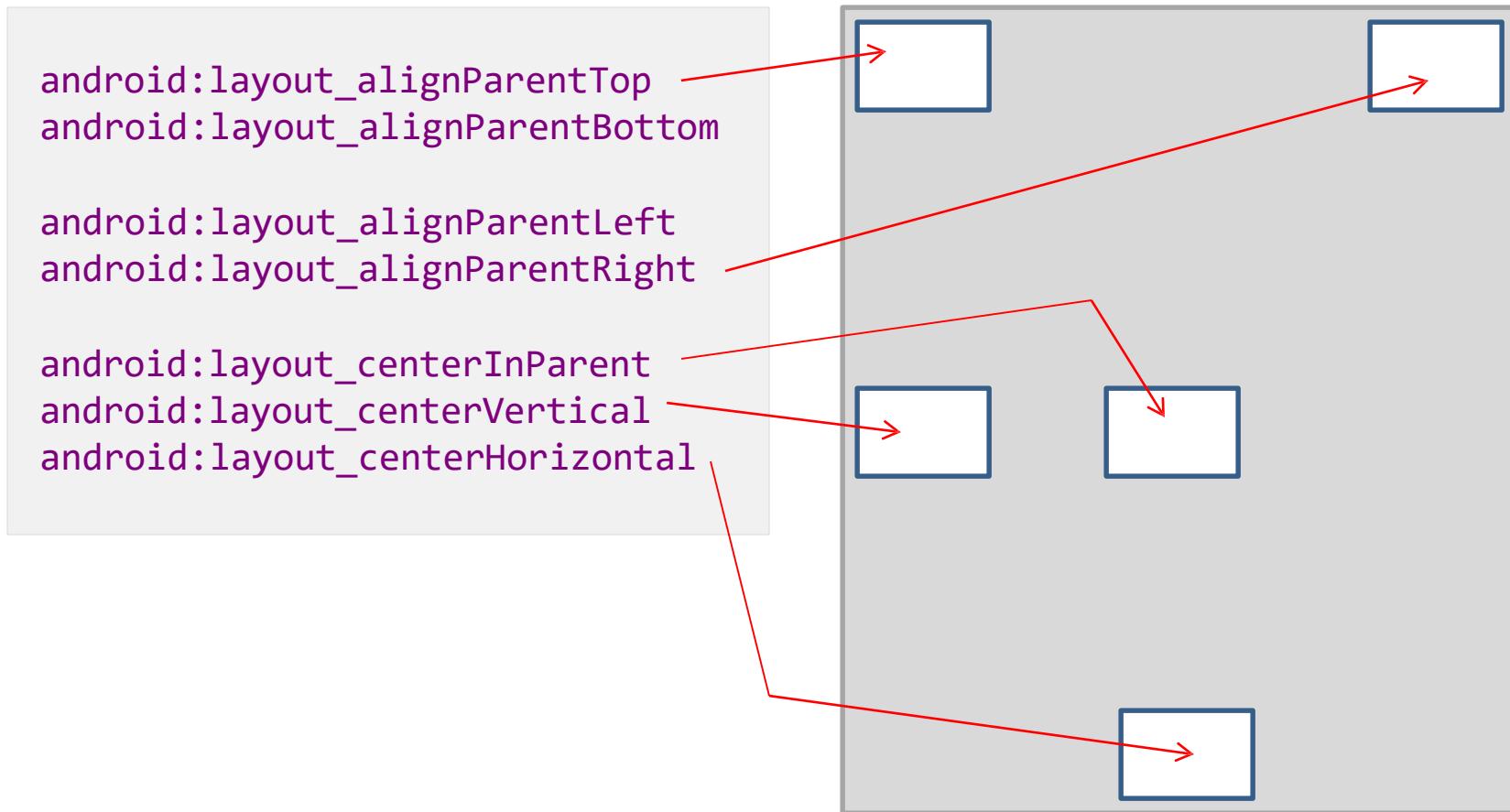


Lokalizacja przycisku jest wyrażona w odniesieniu do (relatywnej) pozycji pola tekstowego (EditText).

Relative Layout

2. Odniesienie do kontenera

Poniżej znajduje się lista atrybutów XML typu **boolean** (=“true/false”) określających pozycję widżetów względem jego **rodzica (kontenera)**.



Relative Layout

2. Odniesienie do innych widżetów

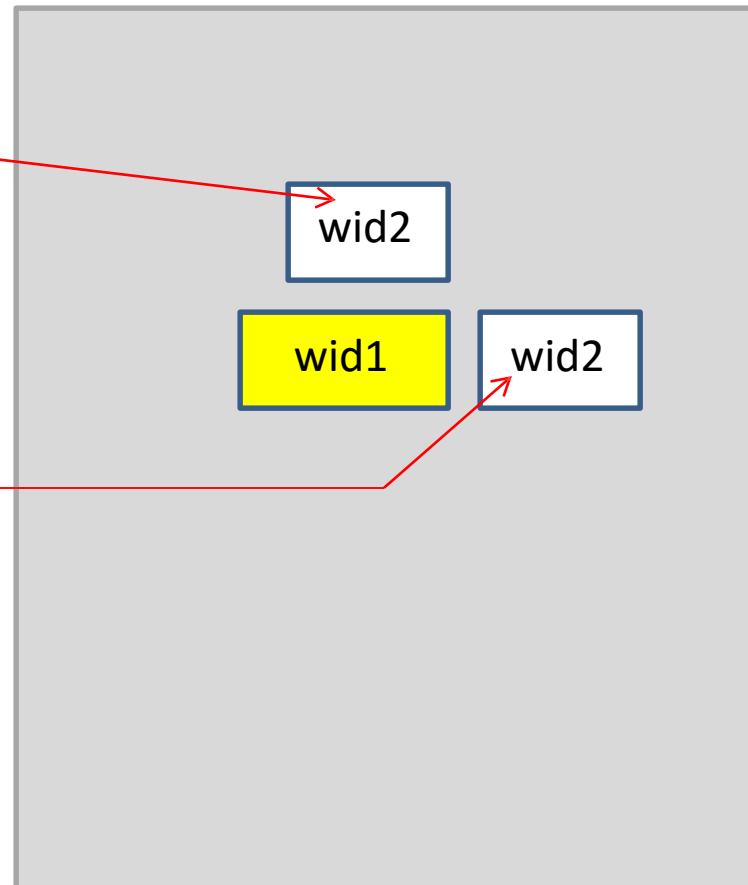
Następujące atrybuty określają położenie widżetu **w odniesieniu do innych widżetów**:

`android:layout_above="@+id/wid1"`

`android:layout_below`

`android:layout_toLeftOf`

`android:layout_toRightOf`



W tym przykładzie “wid2” jest w relacji sąsiedztwa z wid1 (“@+id/wid1”)

Relative Layout

2. Odniesienie do innych widżetów c.d.

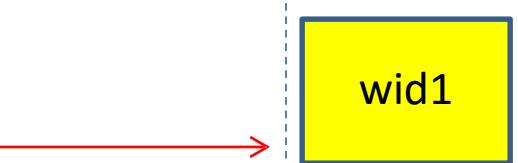
`android:layout_alignTop="@+id/wid1"`



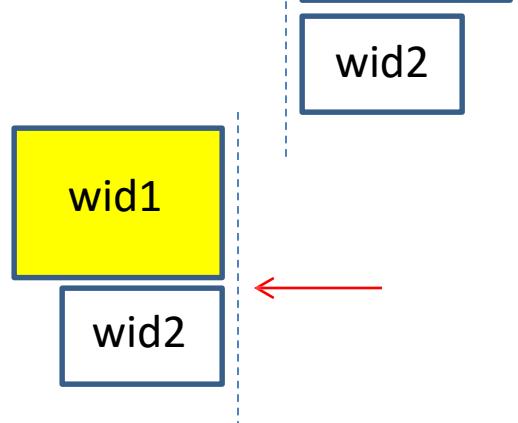
`android:layout_alignBottom = "@+id/wid1"`



`android:layout_alignLeft="@+id/wid1"`



`android:layout_alignRight="@+id/wid1"`



Relative Layout

2. Odniesienie do innych widżetów c.d.

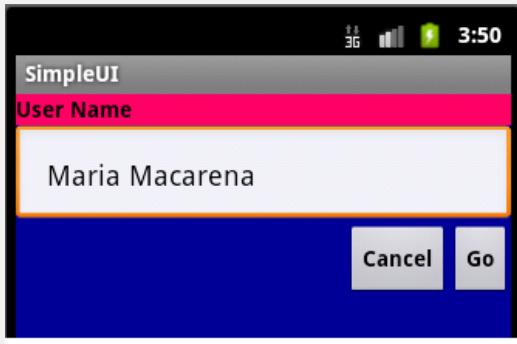
Używając pozycjonowania relatywnego należy zadbać o:

1. Nadanie identyfikatorów (atrybut **android:id**) dla wszystkich elementów do których będziemy się odnosili.
2. Elementy XML są nazywane poprzez zapis: `@+id/...`. Przykładowo pole tekstowe może być nazwane: `android:id="@+id/txtUserName"`
3. Można odwoływać się jedynie do uprzednio zdefiniowanych widżetów. Przykładowo, nowy komponent który znajduje się pod polem tekstowym `txtUserName` można wypożycjonować jako:
`android:Layout_below="@+id/txtUserName"`

Relative Layout

2. Przykład

```
<RelativeLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
        android:id="@+id/myRelativeLayout"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:background="#ff000099" >  
  
<TextView  
        android:id="@+id/LblUserName"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:layout_alignParentLeft="true"  
        android:layout_alignParentTop="true"  
        android:background="#fffff0066"  
        android:text="User Name"  
        android:textColor="#ff000000"  
        android:textStyle="bold" >  
</TextView>
```



```
<EditText  
        android:id="@+id/txtUserName"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:layout_alignParentLeft="true"  
        android:layout_below="@+id/LblUserName"  
        android:padding="20dp" >  
</EditText>  
  
<Button  
        android:id="@+id	btnGo"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
  
        android:layout_alignRight="@+id/txtUserName"  
        android:layout_below="@+id/txtUserName"  
        android:text="Go"  
        android:textStyle="bold" >  
</Button>  
  
<Button  
        android:id="@+id btnCancel"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_below="@+id/txtUserName"  
        android:layout_toLeftOf="@+id(btnGo)"  
        android:text="Cancel"  
        android:textStyle="bold" >  
</Button>  
  
</RelativeLayout>
```

Table Layout

3. Table Layout

1. Układ **TableLayout** wykorzystuje siatkę do pozycjonowania widżetów.
2. Podobnie jak w macierzy, komórki siatki identyfikowane są po numerach wierszy i kolumn.
3. Kolumny są responsywne – mogą się zwężać bądź rozciągać by dostosować się do ich zawartości.
4. Klasa **TableRow** wykorzystywana jest do stworzenia nowego wiersza w którym widżety mogą zostać umieszczone.
5. Liczba kolumn TableRow jest wyznaczana przez liczbę widżetów umieszczonych w danym wierszu.

Table Layout

3. Table Layout – Ustawienie liczby kolumn

Liczba kolumn w ramach wiersza wyznacza jest przez system Android.

Przykład: Jeżeli TableLayout ma trzy wiersze, jeden z dwoma widżetami, jeden z trzema i jeden z czterema, zostaną stworzone przynajmniej 4 kolumny.

0		1	
0		1	2
0	1	2	3

Table Layout

3. Table Layout – Rozciąganie kolumn

- Pojedynczy widżet wewnątrz TableLayout może zajmować więcej niż jedną kolumnę.
- Właściwość **android:layout_span** wskazuje na jaką liczbę kolumn widżet może się rozciągnąć.

```
<TableRow>
    <TextView android:text="URL:" />
    <EditText android:id="@+id/txtData"
        android:layout_span="3" />
</TableRow>
```

Table Layout

3. Table Layout – Rozciąganie kolumn

Widżety w wierszu tabeli ustawiane są od lewej do prawej, rozpoczynając od pierwszej wolnej kolumny. Każda kolumna w układzie rozciąga się zgodnie z początkową zawartością danego widżetu.

Przykład: Pokazana tabela ma 4 kolumny (*indeksy: 0,1,2,3*). Nazwa ("ISBN") tworzy pierwszą kolumnę (*indeks 0*). Pole tekstowe wykorzystuje atrybut `layout_span` by zostać rozciągniętym na 3 kolumny.

The diagram illustrates a 2x4 Table Layout. The first row contains four cells: 'Label (ISBN)' (col 0), 'EditText' (col 1), 'EditText-span' (col 2), and 'EditText-span' (col 3). A yellow box highlights the attribute 'android:layout_span="3"' above the second and third columns, with a blue double-headed arrow spanning these two columns. The second row contains four cells: 'Column 0' (col 0), 'Column 1' (col 1), 'Column 2' (col 2), and 'Column 3' (col 3). Below 'Column 2' and 'Column 3' is a 'Button' labeled 'Cancel' in the first row and 'OK' in the second row. A yellow box highlights the attribute 'android:layout_column="2"' below 'Column 2'.

Label (ISBN)	EditText	EditText android:layout_span="3"	EditText android:layout_span="3"
Column 0	Column 1	Column 2 Button Cancel	Column 3 Button OK

android:layout_column="2"

Android - Graphical User Interfaces

Table Layout – przykład 2

<TableLayout>

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:id="@+id/myTableLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="6dp" >
```

<TableRow>

```
    <TextView
        android:background="#FF33B5E5"
        android:text="Item" />
    <TextView
        android:layout_marginLeft="5dp"
        android:background="#FF33B5E5"
        android:text="Calories" />
    <TextView
        android:layout_marginLeft="5dp"
        android:background="#FF33B5E5"
        android:text="Price $" />
</TableRow>
```

<View>

```
    android:layout_height="1dp"
    android:background="#FF33B5E5" />
```

<TableRow>

```
    <TextView
        android:text="Big Mac" />
    <TextView
        android:gravity="center"
        android:text="530" />
    <TextView
        android:gravity="center"
        android:text="3.99" />
    <Button
        android:id="@+id/btnBuyBigMac"
        android:gravity="center"
        android:text="Buy" />
</TableRow>
```

<View>

```
    android:layout_height="1dp"
    android:background="#FF33B5E5" />
```

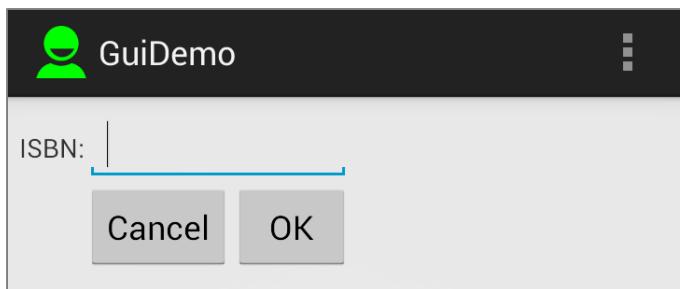
```
<!-- other TableRows omitted --!>
```

</TableLayout>



Table Layout

3. Table Layout - przykład



Spróbuj zmienić
layout_span na 1, 2, 3

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myTableLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="6dp"
    android:orientation="vertical" >
    <TableRow>
        <TextView android:text="ISBN:" />
        <EditText
            android:id="@+id/ediISBN"
            android:layout_span="3" />
    </TableRow>

    <TableRow>
        <Button
            android:id="@+id/cancel"
            android:layout_column="2"
            android:text="Cancel" />
        <Button
            android:id="@+id/ok"
            android:text="OK" />
    </TableRow>
</TableLayout>
```

Zajmij 3
kolumny

Pomiń
kolumny 0, 1

Table Layout

3. Rozciąganie całej tabeli

- Domyślnie, kolumna jest szeroka jak “naturalny” rozmiar najszerzszego widżetu zawartego w tej kolumnie (przykładowo kolumna z przyciskiem “Go” jest węższa niż kolumna z przyciskiem “Cancel”).
- Tabela niekoniecznie musi zajmować całą dostępną przestrzeń w poziomie.
- Jeżeli tabela ma (w poziomie) być zrównana z kontenerem można skorzystać z opcji:

```
android:stretchColumns="column(s)"
```

Jej wartość to indeks kolumny (bądź lista indeksów kolumn) która ma zostać rozciągnięta by zająć całą dostępną przestrzeń w ramach wiersza.

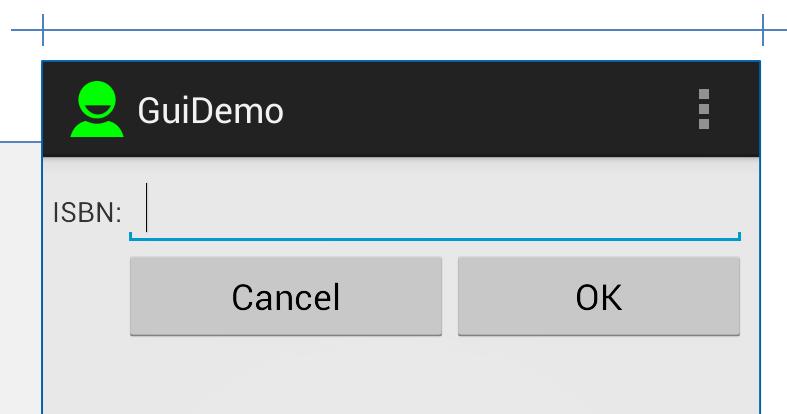
Table Layout

3. Przykład: Rozciąganie kolumn

W tym przykładzie kolumny 2 i 3 zajmują całą (wolną) dostępną przestrzeń w poziomie.

```
...
<TableLayout
    android:id="@+id/myTableLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:stretchColumns="2,3"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    ...

```

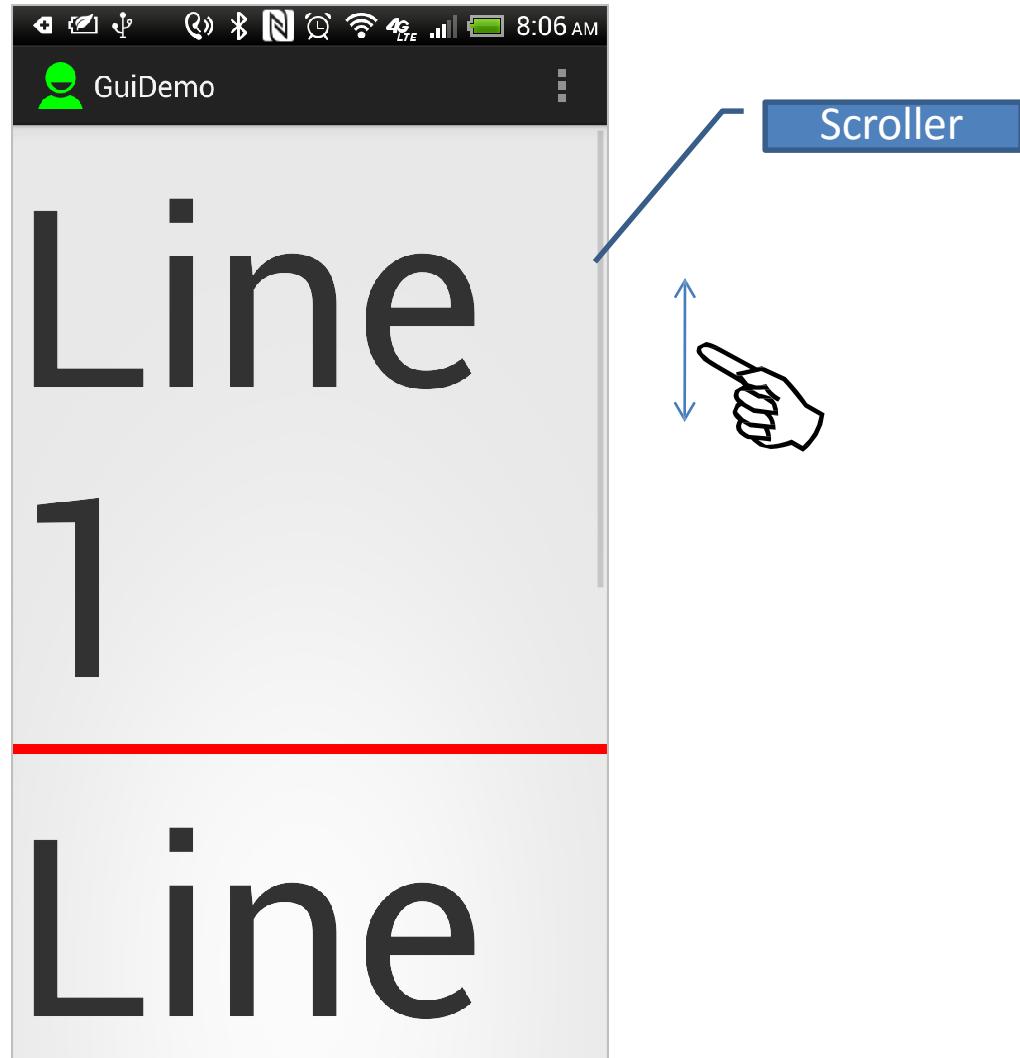


Do zrobienia: spróbuj rozciągania z innymi kolumnami.

ScrollView

4. ScrollView Layout

- Komponent **ScrollView** jest przydatny w sytuacjach, gdy do wyświetlenia jest więcej danych niż można pomieścić na danym ekranie.
- ScrollView zapewnia dostęp do danych z użyciem pasków przewijania.
- Jedynie porcja danych jest wyświetlana naraz. Pozostała część jest ukryta.



ScrollView

4. Przykład: ScrollView Layout

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

<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myScrollView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:id="@+id/myLinearLayoutVertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Line1"
            android:textSize="150dp" />

        <View
            android:layout_width="match_parent"
            android:layout_height="6dp"
            android:background="#ffff0000" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Line2"
            android:textSize="150dp" />

        <View
            android:layout_width="match_parent"
            android:layout_height="6dp"
            android:background="#ffff0000" />

        <TextView
            android:id="@+id/textView3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Line3"
            android:textSize="150dp" />

    </LinearLayout>
</ScrollView>
```

Przykład HorizontalScrollView Layout

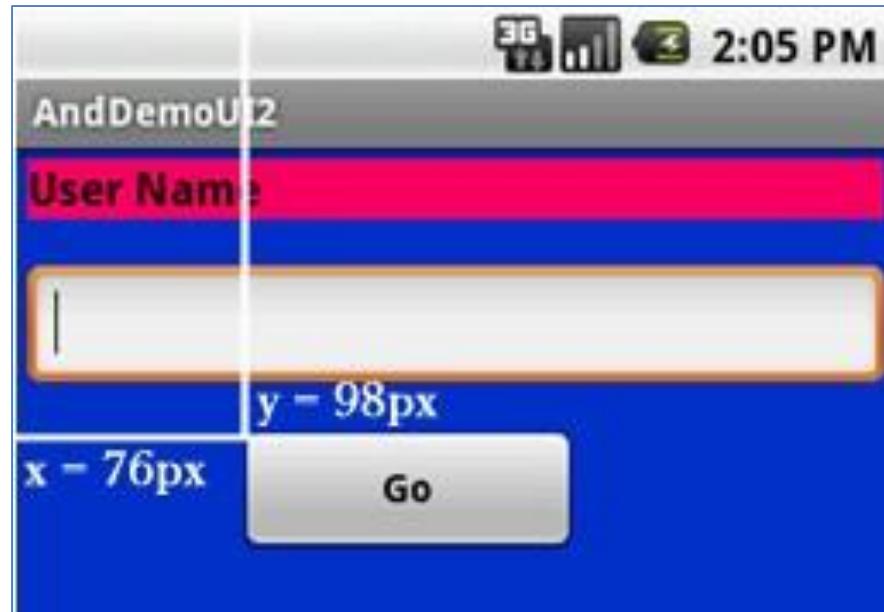
```
<HorizontalScrollView  
    xmlns:android="http://schemas.android.com/apk/res/android"  
        android:id="@+id/myHorizontalScrollView1"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content" >  
  
<LinearLayout  
    android:id="@+id/myLinearLayoutVertical"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="horizontal" >  
  
<TextView  
    android:id="@+id/textView1"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Item1"  
    android:textSize="75sp" />  
  
<View  
    android:layout_width="6dp"  
    android:layout_height="match_parent"  
    android:background="#ffff0000" />  
  
<TextView  
    android:id="@+id/textView2"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Item2"  
    android:textSize="75sp" />  
  
<View  
    android:layout_width="6dp"  
    android:layout_height="match_parent"  
    android:background="#ffff0000" />  
  
<TextView  
    android:id="@+id/textView3"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Item3"  
    android:textSize="75sp" />  
</LinearLayout>  
  
</HorizontalScrollView>
```



Absolute Layout

5. Absolute Layout

- Układ pozwala podać dokładną pozycje (koordynaty x/y) jego potomków.
- Pozycjonowanie absolutne *jest mniej elastyczne i trudniejsze w utrzymaniu* niż w przypadku pozostałych układów opartych o relację między komponentami.



Absolute Layout

5. Przykład Absolute Layout

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/myLinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff0033cc"
    android:padding="4dp"
    xmlns:android="http://schemas.android.com/apk/res/android"
>

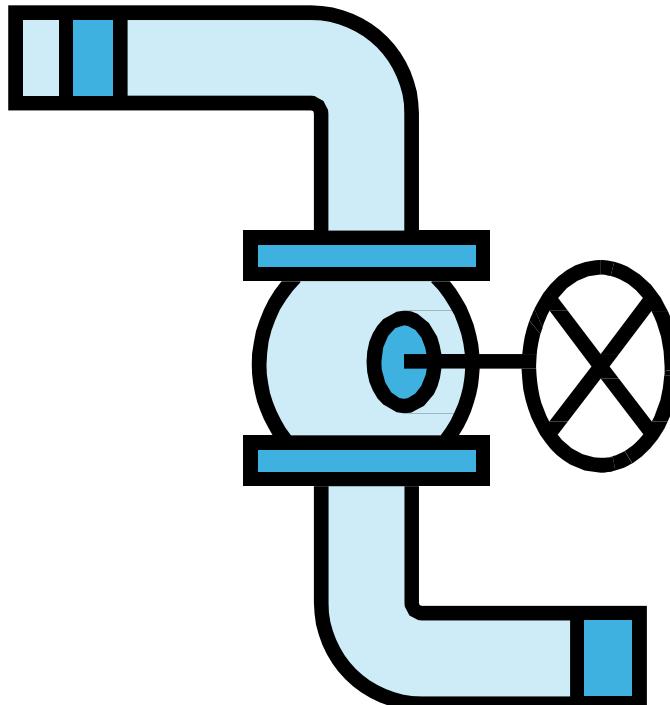
<TextView
    android:id="@+id/tvUserNam"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#ffff0066"
    android:text="User Name"
    android:textSize="16sp"
    android:textStyle="bold"
    android:textColor="#ff000000"
    android:layout_x="0dp"
    android:layout_y="10dp"
>
</TextView>
<EditText
    android:id="@+id/etName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:layout_x="0dp"
    android:layout_y="38dp"
>
</EditText>
<Button
    android:layout_width="120dp"
    android:text="Go"
    android:layout_height="wrap_content"
    android:textStyle="bold"
    android:id="@+id/btnGo"
    android:layout_x="100dp"
    android:layout_y="170dp"  />
</AbsoluteLayout>
```



Powiązanie układu z kodem Java

Należy „połączyć” elementy XML i odpowiadające im obiekty w ramach instancji klasy aktywności napisanej w Javie.

XLM Layout
<xml....
...
...
</xml>



JAVA code
public class
{
...
...
}

Powiązanie układu z kodem Java

Założymy, że GUI zostało stworzone w *res/layout/main.xml*. Ten układ może zostać powiązany z daną aktywnością poprzez instrukcję:

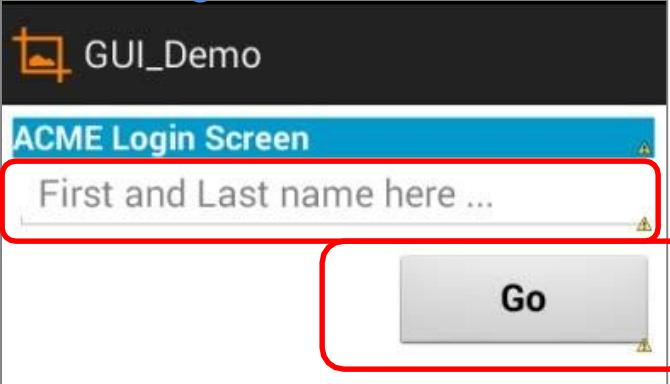
```
setContentView(R.layout.main);
```

Poszczególne widżety, jak np. *myButton* mogą zostać zaadresowane z pomocą polecenia `findViewById(...)` tzn:

```
Button btn= (Button) findViewById(R.id.myButton);
```

gdzie **R** jest klasą automatycznie wygenerowaną by możliwe było śledzenie zasobów dostępnych dla danej aplikacji. W szczególności **R.id...** stanowi kolekcję widżetów zdefiniowanych w ramach interfejsu/ów XML.

Powiązanie układu z kodem Java



```
<!-- XML LAYOUT -->
<LinearLayout
    android:id="@+id/myLinearLayout"
    ... >

    <TextView
        android:text="ACME Login Screen"
        ... />

    <EditText
        android:id="@+id/edtUserName"
        ... />

    <Button
        android:id="@+id/btnGo"
        ... />

</LinearLayout>
```

Java code

```
package csu.matos.gui_demo;
import android...;

public class MainActivity extends Activity {

    EditText edtUserName;
    Button btnGo;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        edtUserName = (EditText) findViewById(R.id.edtUserName);
        btnGo = (Button) findViewById(R.id.btnGo);
        ...
    }
    ...
}
```

Powiązanie układu z kodem Java

Dodawanie nasłuchiwaczy do widżetów

W tym momencie przycisk btn może zostać wykorzystany np. poprzez zdefiniowanie dla niego reakcji na kliknięcie:

```
btn.setOnClickListener(new OnClickListener() {  
    @Override  
    → public void onClick(View v) {  
        updateTime();  
    }  
} );  
  
private void updateTime() {  
    btn.setText(new Date().toString());  
}
```

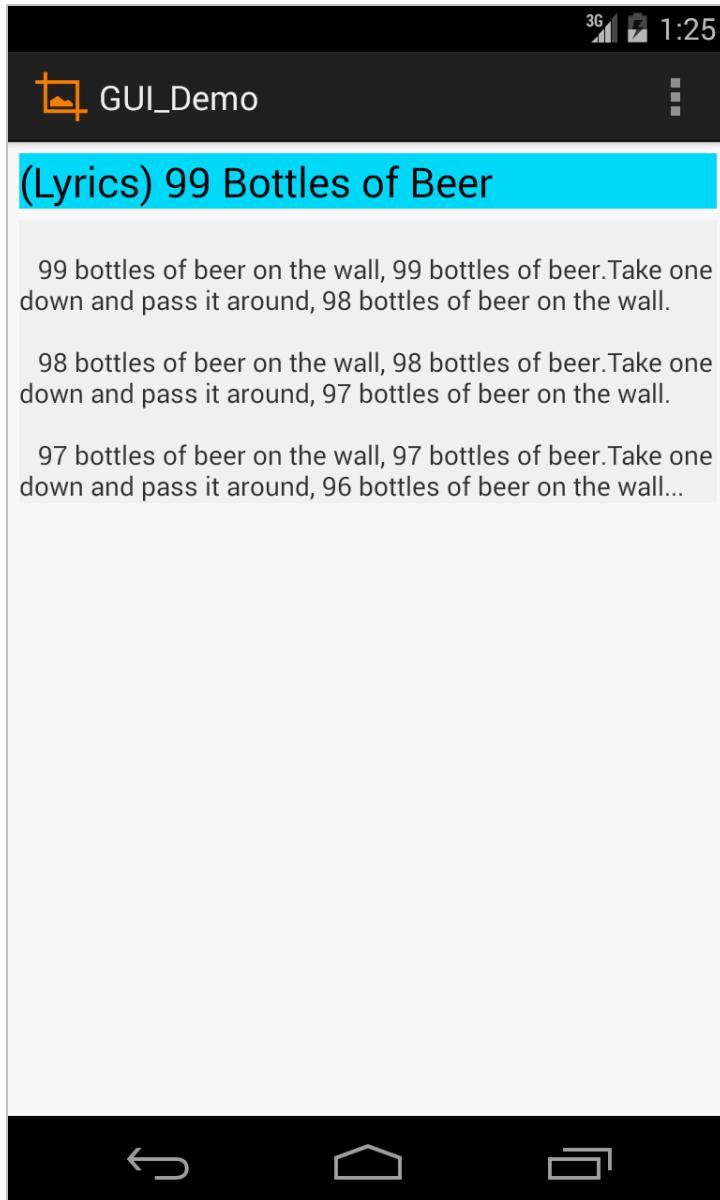
Czym jest kontekst?

Dodatek

Dla Androida **Context** definiuje tzw. **środowisko** na którym aplikacja może tworzyć i używać zasobów.

- Kiedy widżet jest tworzony, jest mu przyporządkowany określony kontekst. Poprzez afiliację do tego środowiska, ma on wówczas dostęp do innych składowych hierarchii do której został przyporządkowany.
- Dla prostej aplikacji składającej się wyłącznie z jednej aktywności np. `MainActivity` metoda **getApplicationContext()** oraz referencja **MainActivity.this** zwracają ten sam rezultat.
- Aplikacja ma jednak zwykle **wiele aktywności**. W takiej sytuacji dostępny jest jeden globalny kontekst aplikacji oraz po jednym kontekście dla każdej z aktywności, każdy umożliwiający dostęp do zasobów zdefiniowanych w ramach tego kontekstu.

Etykiety



- Etykieta w kontekście Androida nazywana jest **TextView**.
- Zwykle wykorzystywana jest do prezentacji danych tekstowych bądź opisywania komponentów aplikacji.
- Etykiety nie umożliwiają edycji tekstu przez użytkownika.
- Tekst może zawierać pewne znaki specjalne jak np. \n (nowa linia)
- Można również wykorzystać formatowanie HTML:
`Html.fromHtml("bold string")`

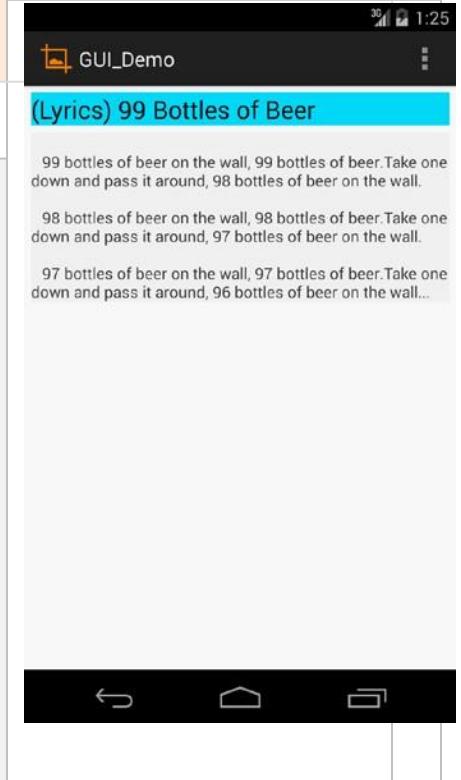
Etykiety - przykład

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="6dp" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/holo_blue_bright"
        android:text="(Lyrics) 99 Bottles of Beer"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="6dp"
        android:background="@color/gray_light"
        android:text="\n\t99 bottles of beer on the wall, 99 bottles of beer. Take one down and
        pass it around, 98 bottles of beer on the wall.\n\n\t98 bottles of beer on the wall, 98 bottles
        of beer. Take one down and pass it around, 97 bottles of beer on the wall. \n\n\t97 bottles of
        beer on the wall, 97 bottles of beer. Take one down and pass it around, 96 bottles of beer on
        the wall..." 
        android:textSize="14sp" />

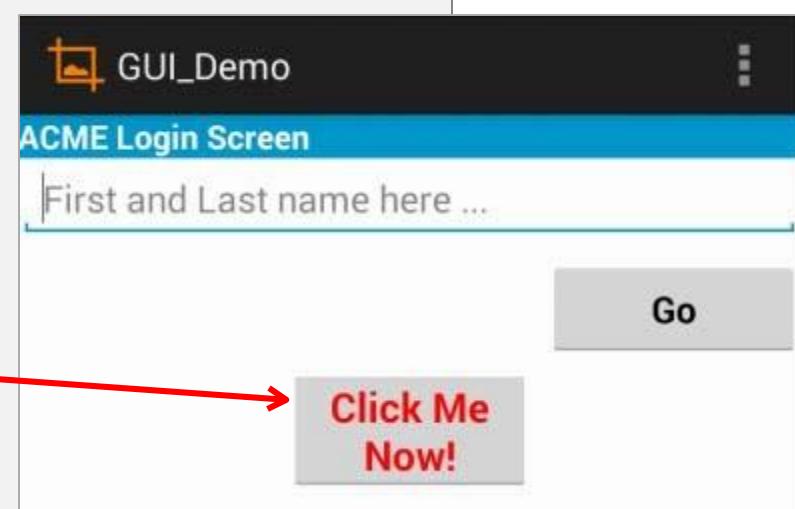
</LinearLayout>
```



Przyciski

- Widżet **Button** jest graficzną reprezentacją standardowego przycisku.
- **Button** jest podklassą **TextView**. Dlatego formatowanie przycisku przebiega podobnie jak formatowanie **TextView**.
- Można zmienić domyślny wygląd przycisku poprzez realizację własnej specyfikacji *drawable.xml* (*podanej jako „tło”*). Można wówczas wskazać kształt, kolor, obramowanie, wierzchołki, gradient oraz wygląd poszczególnych stanów (naciśnięty, posiada focus).

```
<Button  
    android:id="@+id/btnClickMeNow"  
    android:layout_width="120dp"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:layout_marginTop="5dp"  
    android:gravity="center"  
    android:padding="5dp"  
    android:text="Click Me Now!"  
    android:textColor="#ffff0000"  
    android:textSize="20sp"  
    android:textStyle="bold" />
```

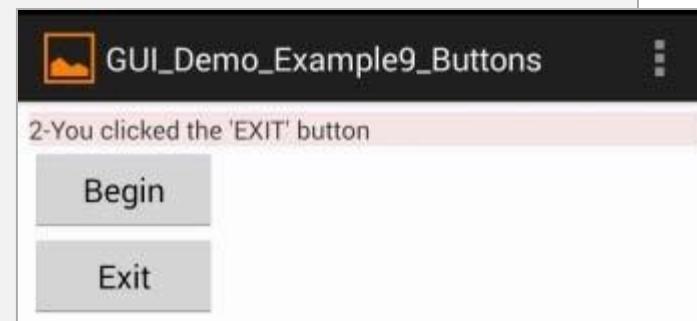


Podłączanie wielu przycisków

```
public class MainActivity extends Activity implements OnClickListener {  
    TextView txtMsg;  
    Button btnBegin;  
    Button btnExit;  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main );  
  
        txtMsg = (TextView) findViewById(R.id.txtMsg);  
        btnBegin = (Button) findViewById(R.id.btnBegin);  
        btnExit = (Button) findViewById(R.id.btnExit);  
  
        btnBegin.setOnClickListener(this);  
        btnExit.setOnClickListener(this);  
    } //onCreate
```

```
@Override  
public void onClick(View v) {  
    if (v.getId() == btnBegin.getId()) {  
        txtMsg.setText("1-You clicked the 'BEGIN' button");  
    }  
    if (v.getId() == btnExit.getId()) {  
        txtMsg.setText("2-You clicked the 'EXIT' button");  
    }  
} //onClick
```

Proszę zwrócić uwagę na implementację interfejsu **OnClickListener**, jako alternatywną wersję obsługi wielu przycisków. Metoda **onClick** sprawdza, który z przycisków jest źródłem zdarzenia i wykonuje zadaną akcję.



Podłączanie wielu przycisków

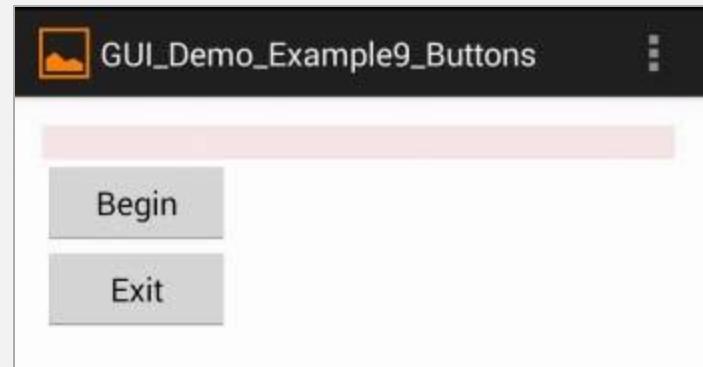
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="6dp" >

    <TextView
        android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#88eed0d0" />

    <Button
        android:id="@+id/btnBegin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="5"
        android:text="Begin" />

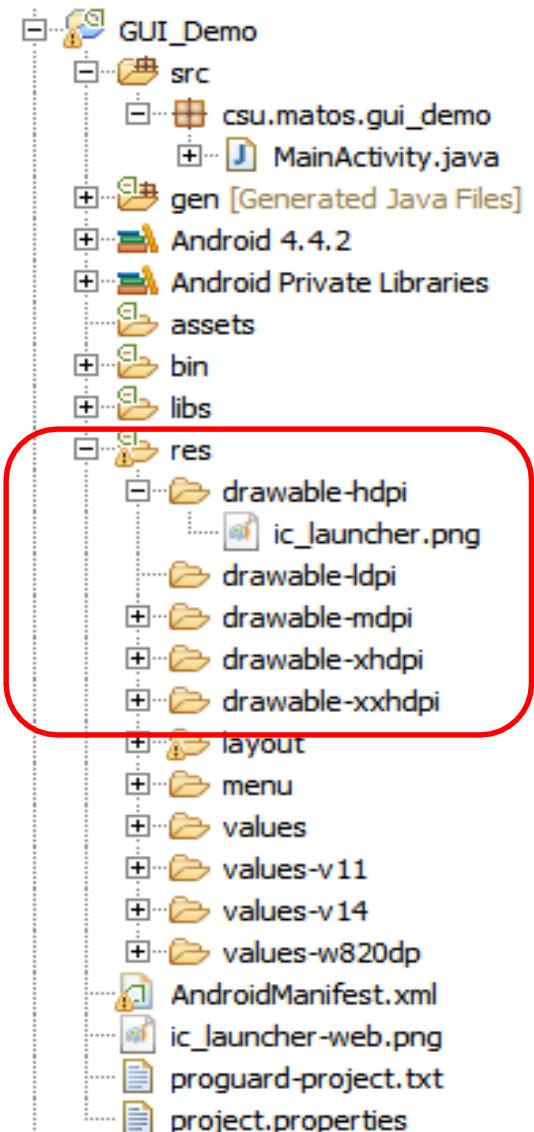
    <Button
        android:id="@+id/btnExit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="5"
        android:text="Exit" />

</LinearLayout>
```



ImageView i ImageButton

- **ImageView** oraz **ImageButton** pozwalają na umieszczenie grafik (gif, jpg, png, etc).
- Stanowią analogię dla komponentów odpowiednio *TextView* oraz *Button*
- Każdy taki widżet posiada atrybut **android:src** lub **android:background** (w pliku XML) pozwalający określić używany obrazek.
- Obrazki przechowywane są w folderze **res/drawable** (opcjonalnie można dodać różne wersje tego samego obrazka w katalogach z przedrostkami *medium*, *high*, *x-high*, *xx-high* i *xxx-high*). Szczegóły dostępne są na:
<http://developer.android.com/design/style/iconography.html>



ImageView i ImageButton

```
<LinearLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:padding="6dp"  
    android:orientation="vertical" >
```

```
    <ImageButton
```

```
        android:id="@+id/imgButton1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:src="@drawable/ic_launcher" >
```

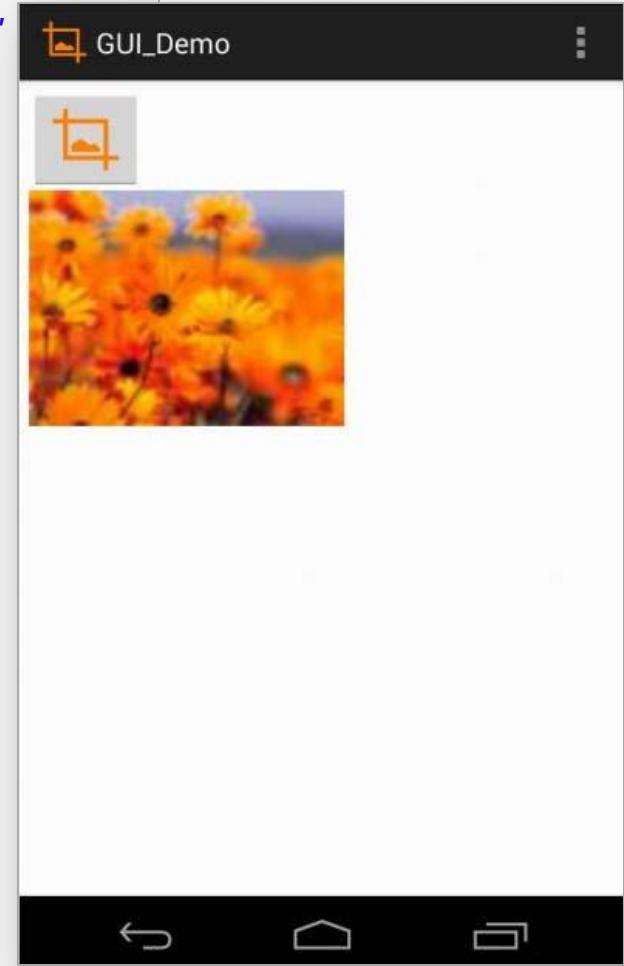
```
    </ImageButton>
```

```
    <ImageView
```

```
        android:id="@+id/imgView1"  
        android:layout_width="200dp"  
        android:layout_height="150dp"  
        android:scaleType="fitXY"  
        android:src="@drawable/flowers1" >
```

```
    </ImageView>
```

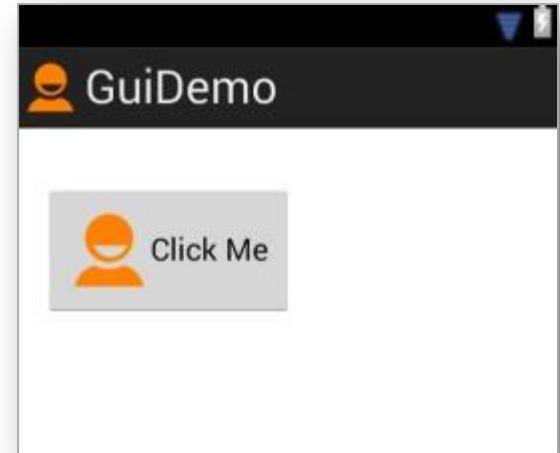
```
</LinearLayout>
```



ImageView i ImageButton

Zwykły przycisk typu **Button** też może wyświetlać grafikę, ale **ImageButton** ma więcej opcji

```
<LinearLayout  
    . . .  
  
<Button  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:drawableLeft="@drawable/ic_launcher"  
        android:gravity="left/center_vertical"  
        android:padding="15dp"  
        android:text="Click me" />  
  
</LinearLayout>
```



Jak Android wykorzystuje ikony?

Ikony – małe obrazki reprezentujące aplikację bądź jej część. Mogą pojawiać się w wielu miejscach aplikacji takich jak:

- Ekran domowy
- Launcher
- Menu główne
- Action Bar
- Pasek stanu
- Zakładkach
- Dialogach
- Listach

Więcej informacji dostępnych jest pod:

<http://developer.android.com/design/style/iconography.html>

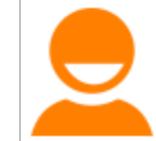
WSKAZÓWKA: Wiele witryn oferuje darmową konwersję między różnymi formatami graficznymi:

<http://www.prodraw.net/favicon/index.php>

<http://converticon.com/>



mdpi (761 bytes)
1x = 48 x 48 pixels
BaseLine



hdpi (1.15KB)
1.5x = 72 x 72 px



x-hdpi (1.52KB)
2x = 96 x 96 px



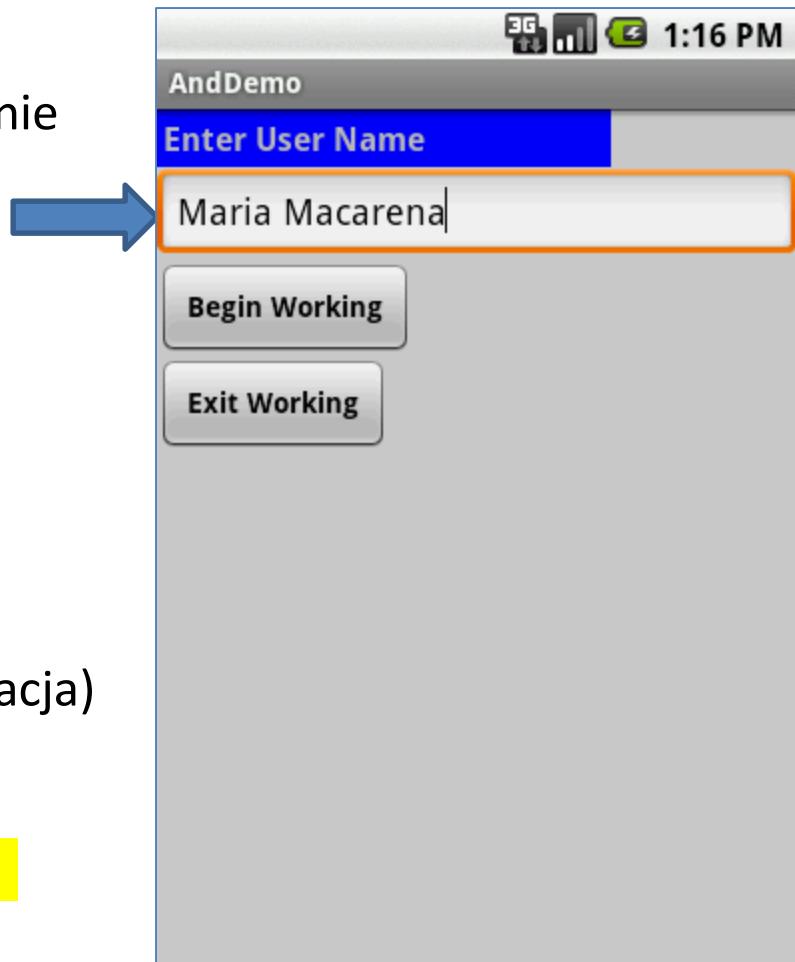
xx-hdpi (2.47KB)
3x = 144 x 144 px

Pola tekstowe

- Widżet EditText jest rozszerzeniem TextView umożliwiającym wprowadzenie tekstu przez użytkownika.
- Prócz zwykłego tekstu można używać podstawowych znaczników HTML by uzyskać pogrubienie, podkreślenie, kursywę. Umożliwia to metoda:
Html.fromHtml(html_text)
- Dostęp do tekstu (pobieranie, modyfikacja) możliwy jest dzięki metodom:

```
txtBox.setText("tekst")
```

```
txtBox.getText().toString()
```



Pola tekstowe

Format wprowadzania danych

Komponent EditText może być ustawiony by przyjmował tylko określone typu danych: numery (ze znakiem i częścią ułamkową bądź bez), numery telefonów, daty, czas, odnośniki itp.

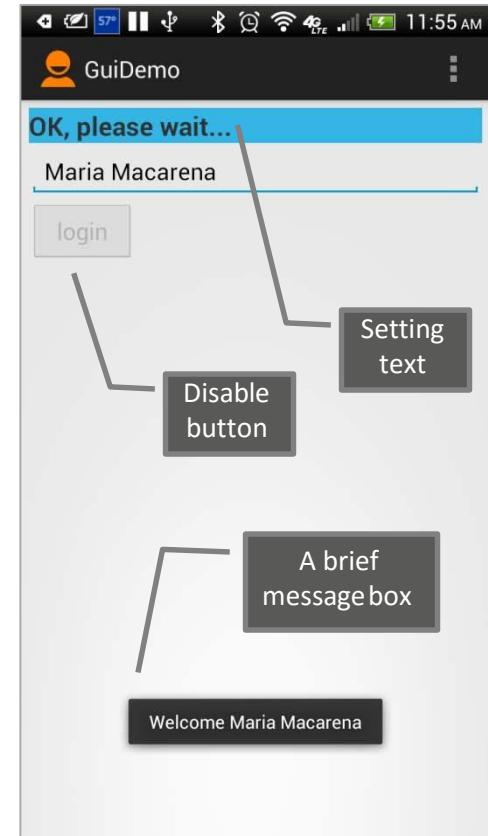
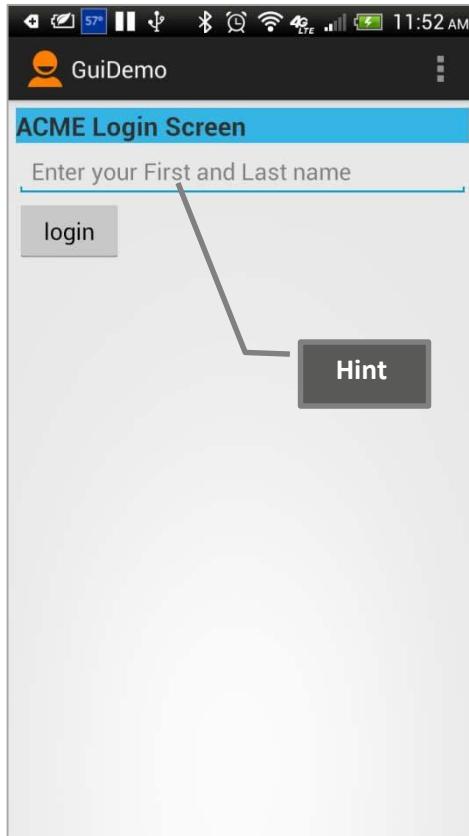
Ustawienie EditText by akceptował tylko wybrany typ danych umożliwia klauzula XML:

android:inputType="choices"

gdzie **choices** zawiera dowolne wartości pokazane na slajdzie. Wartości można łączyć, przykładowo zapis **textCapWords | textAutoCorrect** akceptuje słowa zaczynające się z wielkiej litery i umożliwia automatyczną kontrolę pisowni.

- Ⓐ "none"
- Ⓐ "text"
- Ⓐ "textCapCharacters"
- Ⓐ "textCapWords"
- Ⓐ "textCapSentences"
- Ⓐ "textAutoCorrect"
- Ⓐ "textAutoComplete"
- Ⓐ "textMultiLine"
- Ⓐ "textImeMultiLine"
- Ⓐ "textNoSuggestions"
- Ⓐ "textUri"
- Ⓐ "textEmailAddress"
- Ⓐ "textEmailSubject"
- Ⓐ "textShortMessage"
- Ⓐ "textLongMessage"
- Ⓐ "textPersonName"
- Ⓐ "textPostalAddress"
- Ⓐ "textPassword"
- Ⓐ "textVisiblePassword"
- Ⓐ "textWebEditText"
- Ⓐ "textFilter"
- Ⓐ "textPhonetic"
- Ⓐ "number"
- Ⓐ "numberSigned"
- Ⓐ "numberDecimal"
- Ⓐ "phone"
- Ⓐ "datetime"
- Ⓐ "date"
- Ⓐ "time"

Ekran logowania - przykład



Zdjęcia z HTC-One

Ekran logowania - przykład

Układ 1 z 2

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="6dp" >

    <TextView
        android:id="@+id/txtLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_blue_light"
        android:text="@string/ACME_Login_Screen"
        android:textSize="20sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/edtUserName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="2dp"
        android:hint="@string/Enter_your_First_and_Last_name"
        android:inputType="textCapWords/textAutoCorrect"
        android:textSize="18sp" >
        <requestFocus />
    </EditText>
```

Ekran logowania - przykład

Układ 2 z 2

```
<Button  
    android:id="@+id/btnLogin"  
    android:layout_width="82dp"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="2dp"  
    android:text="@string/login" />  
</LinearLayout>
```

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<!-- this is the res/values/strings.xml file -->  
<resources>  
  
    <string name="app_name">GuiDemo</string>  
    <string name="action_settings">Settings</string>  
    <string name="Login">Login</string>  
    <string name="ACME_Login_Screen">ACME Login Screen</string>  
    <string name="Enter_your_First_and_Last_name">Enter your First and Last name</string>  
  
</resources>
```

Ekran logowania - przykład

```
public class MainActivity extends ActionBarActivity {  
  
    // class variables representing UI controls to be controlled from the Java program  
    TextView txtLogin;  
    EditText edtUserName;  
    Button btnLogin;  
  
    // variables used with the Toast message class  
    private Context context;  
    private int duration = Toast.LENGTH_SHORT;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        // show the login screen  
        setContentView(R.layout.activity_main);  
        context = getApplicationContext();  
  
        // binding the UI's controls defined in "main.xml" to Java code  
        txtLogin = (TextView) findViewById(R.id.txtLogin);  
        edtUserName = (EditText) findViewById(R.id.edtUserName);  
        btnLogin = (Button) findViewById(R.id.btnLogin);
```

Ekran logowania - przykład

```
// LISTENER: allowing the button widget to react to user interaction
btnLogin.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {

        String userName = edtUserName.getText().toString();

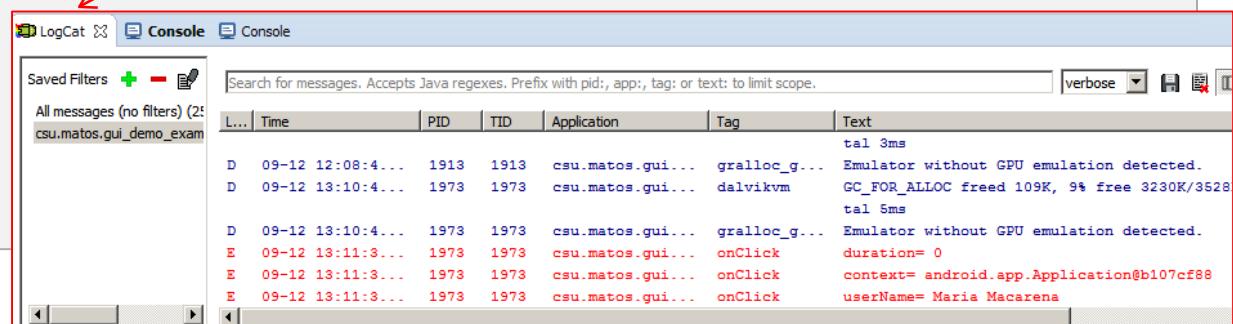
        Log.e("onClick ", "duration= " + duration);
        Log.e("onClick ", "context= " + context.toString());
        Log.e("onClick ", "userName= " + userName);

    }

    if (userName.equals("Maria Macarena")) {
        txtLogin.setText("OK, please wait...");
        Toast.makeText(getApplicationContext(),
            "Welcome " + userName, duration).show();
        btnLogin.setEnabled(false);
    } else {
        Toast.makeText(context, userName + " is not a valid USER",
            duration).show();
    }
}); // onClick

} // onCreate
```

Log.e jako
debug – usuń
pozniej!!!



Ekran logowania - przykład

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

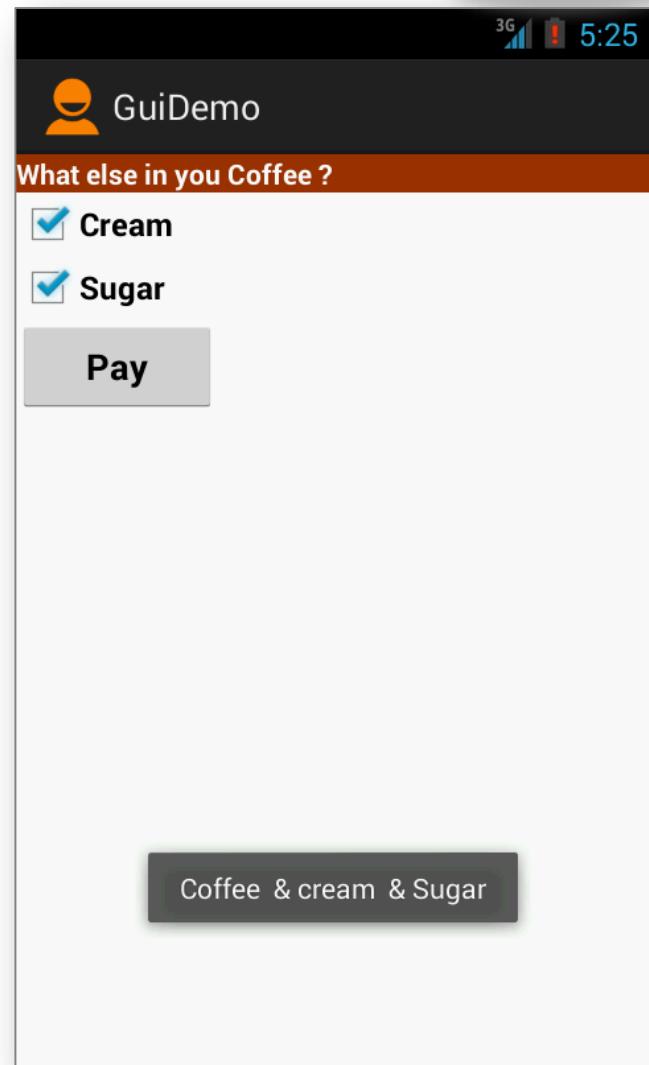
Pole typu CheckBox

Pole typu checkbox można interpretować jako **dwustanowy** przycisk który może być *zaznaczony* bądź *nie*.

Dany ekran może zawierać wiele niezależnie działających pól wyboru. W danym czasie więcej niż jeden checkbox może zostać zaznaczony.

W przykładzie wykorzystano komponenty CheckBox do możliwości wyboru dodatków do kawy (cukier, śmietanka).

Po kliknięciu przycisku Pay użytkownikowi prezentowany jest komunikat zawierający wybraną kombinację zamówienia.



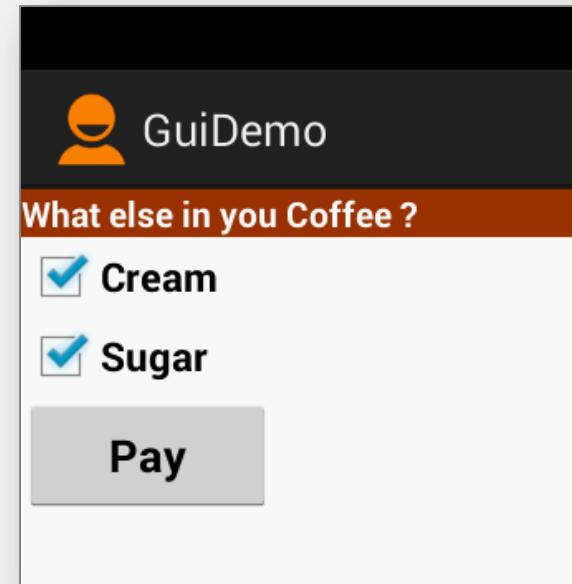


Checkbox - przykład

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="6dp"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/LabelCoffee"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff993300"
        android:text="@string/coffee addons"
        android:textColor="@android:color/white"
        android:textStyle="bold" />

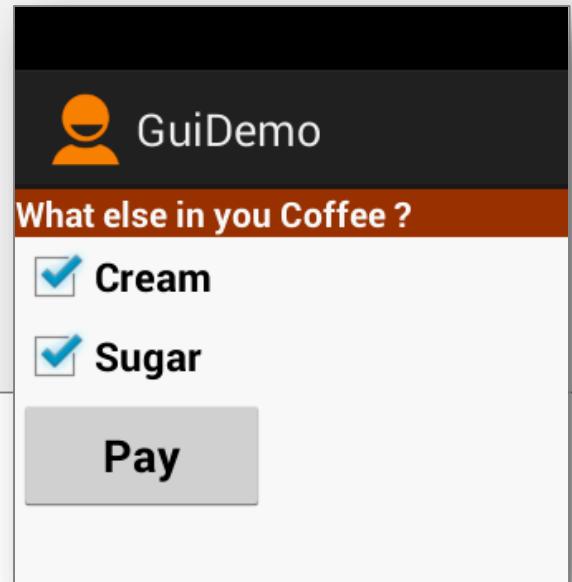
    <CheckBox
        android:id="@+id/chkCream"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cream"
        android:textStyle="bold" />
```





Checkbox - przykład

```
<CheckBox  
    android:id="@+id/chkSugar"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/sugar"  
    android:textStyle="bold" />  
  
<Button  
    android:id="@+id(btnPay"  
    android:layout_width="153dp"  
    android:layout_height="wrap_content"  
    android:text="@string/pay"  
    android:textStyle="bold" />  
  
</LinearLayout>
```





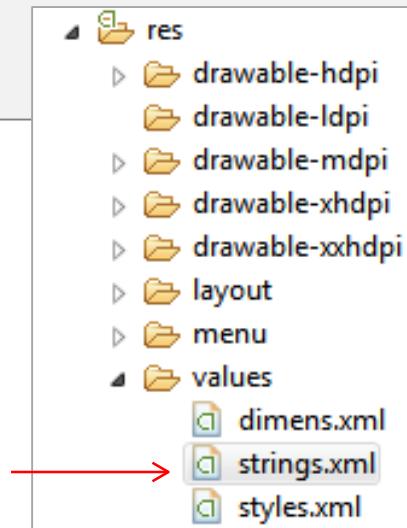
Checkbox – przykład [@string/...]

Zasoby: res/values/strings

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

    <string name="app_name">GuiDemo</string>
    <string name="action_settings">Settings</string>

    <string name="click_me">Click Me</string>
    <string name="sugar">Sugar</string>
    <string name="cream">Cream</string>
    <string name="coffee_addons">What else in your coffee?</string>
    <string name="pay">Pay</string>
</resources>
```





Checkbox - przykład

```
public class MainActivity extends Activity {  
  
    CheckBox chkCream;  
    CheckBox chkSugar;  
    Button btnPay;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        //binding XML controls with Java code  
        chkCream = (CheckBox)findViewById(R.id.chkCream);  
        chkSugar = (CheckBox)findViewById(R.id.chkSugar);  
        btnPay = (Button) findViewById(R.id.btnPay);
```



Checkbox - przykład

```
//LISTENER: wiring button-events-&-code
    btnPay.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
    String msg = "Coffee ";
    if (chkCream.isChecked()) {
        msg += " & cream ";
    }
    if (chkSugar.isChecked()){
        msg += " & Sugar";
    }
    Toast.makeText(getApplicationContext(),
                       msg, Toast.LENGTH_SHORT).show();
    //go now and compute cost...

} //onClick
});

} //onCreate

} //class
```

Pole typu RadioButton

- Pole typu **RadioButton** (jak CheckBox) jest dwu-stanowym przyciskiem, który może być *zaznaczony* bądź *nie*.
- Zwykle pola tego typu są agregowane w kontener o nazwie **RadioGroup**. Zastosowanie tego kontenera powoduje, że pola wyboru zachowują się jako **wzajemnie wykluczające się selektory**. Oznacza to, że zmiana jednego z przycisków powoduje odznaczenie pozostałych.
- Właściwości dotyczących kroju, stylu czy koloru czcionki zarządzane są podobnie jak w przypadku etykiety TextView.
- Można wywołać metodę ***isChecked()*** by sprawdzić czy poszczególny przycisk jest zaznaczony, natomiast metoda ***toggle()*** odpowiada za zmianę jego stanu.



RadioButton - przykład

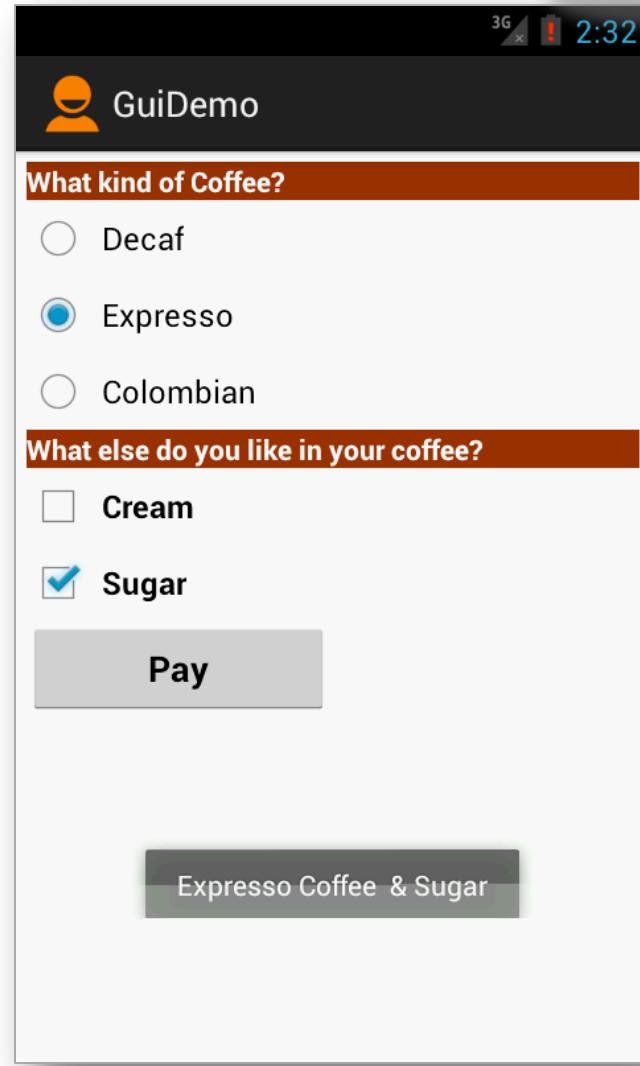


Przykład

Rozszerzono poprzednią aplikację dodając komponent **RadioGroup** by umożliwić wybór konkretnego typu kawy.

RadioGroup

Podsumowanie





RadioButton - przykład

Na podstawie poprzedniego przykładu – tylko nowe rzeczy są prezentowane

```
<TextView  
    android:id="@+id/textView1"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:background="#ff993300"  
    android:text="@string/kind_of_coffee"  
    android:textColor="#fffffff"  
    android:textStyle="bold" />
```



```
<RadioGroup  
    android:id="@+id/radioGroupCoffeeType"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" >  
  
<RadioButton  
    android:id="@+id/radDecaf"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/decaf" />  
  
<RadioButton  
    android:id="@+id/radExpresso"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/expresso" />  
  
<RadioButton  
    android:id="@+id/radColombian"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:checked="true"  
    android:text="@string/colombian" />  
</RadioGroup>
```



RadioButton - przykład

```
public class MainActivity extends Activity {  
  
    CheckBox chkCream;  
    CheckBox chkSugar;  
    Button btnPay;  
  
    RadioGroup radCoffeeType;  
    RadioButton radDecaf;  
    RadioButton radExpresso;  
    RadioButton radColombian;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        chkCream = (CheckBox) findViewById(R.id.chkCream);  
        chkSugar = (CheckBox) findViewById(R.id.chkSugar);  
        btnPay = (Button) findViewById(R.id.btnPay);  
  
        radCoffeeType = (RadioGroup) findViewById(R.id.radioGroupCoffeeType);  
        radDecaf = (RadioButton) findViewById(R.id.radDecaf);  
        radExpresso = (RadioButton) findViewById(R.id.radExpresso);  
        radColombian = (RadioButton)  
            findViewById(R.id.radColombian);
```



RadioButton - przykład

```
// LISTENER: wiring button-events-&-code
btnPay.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        String msg = "Coffee ";
        if (chkCream.isChecked())
            msg += " & cream ";
        if (chkSugar.isChecked())
            msg += " & Sugar";

        // get selected radio button ID number
        int radioId = radCoffeeType.getCheckedRadioButtonId();

        // compare selected's Id with individual RadioButtons ID
        if (radColombian.getId() == radioId)
            msg = "Colombian " + msg;
        // similarly you may use .isChecked() on each RadioButton
        if (radExpresso.isChecked())
            msg = "Expresso " + msg;
        // similarly you may use .isChecked() on each RadioButton
        if (radDecaf.isChecked())
            msg = "Decaf " + msg;

        Toast.makeText(getApplicationContext(), msg, 1).show();
        // go now and compute cost...
    }// onClick
});
}// onCreate

}// class
```



RadioButton - przykład

Wskazówka

```
radGroupradioId = (RadioGroup)findViewById(R.id.radioGroup1);  
  
int radioId = radGroupradioId.getCheckedRadioButtonId();  
  
switch (radioId) {  
    case R.id.radColombian: msg += " Colombian "; break;  
    case R.id.radExpresso: msg += " Expresso "; break;  
    case R.id.radDecaf: msg += " Decaf "; break;  
}
```

Alternatywny przykład zarządzania **RadioGroup** – nie wymaga badania stanu każdego z przycisków z osobna.

Przydatne atrybuty XML i metody Java

By kontrolować focus (XML):

 android:visibility
 android:background
 <requestFocus />

 true/false – widoczność
 color, image, drawable
 ustaw domyślny focus

Metody Java

 myButton.requestFocus()
 myTextBox.isFocused()
 myWidget.setEnabled()
 myWidget.isEnabled()

GUI

Obraz wygenerowano przy pomocy
Android Asset Studio



<http://romannurik.github.io/AndroidAssetStudio/>

Dodatek A. Wykorzystanie dyrektywy @string



Dobry programista Android **NIE** wprowadza bezpośrednio literałów znakowych w kodzie aplikacji.

Przykładowo, definiując komponent **TextView** by pokazać np. lokalizację przedsiębiorstwa, nie powinno się stosować zapisu `android:text="Cleveland"` (powoduje to też wyświetlenie ostrzeżenia **Warning [I18N] Hardcoded string "Cleveland", should use @string resource**)

Zamiast tego powinno stosować się 2-etapową procedurę:

1. Dodać literał np.. **headquarter** do **res/values/string.xml**.
`<string name="headquarter">Cleveland</string>`
2. Aby odwołać się do stworzonego literała należy wykorzystać zapis **@string/headquarter**. W przypadku etykiety wystarczy zapis `android:text="@string/headquarter"`

Dlaczego?

Dodatek B.

Android Asset Studio



LINK: <http://romannurik.github.io/AndroidAssetStudio/>

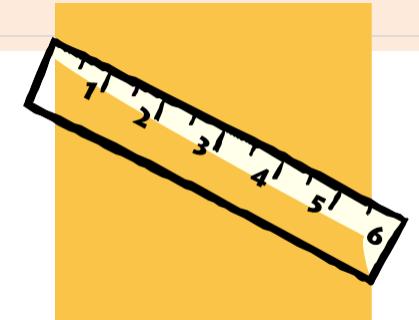
Wspomniane narzędzie oferuje możliwość tworzenia różnych rodzajów ikon oraz innych elementów graficznych

Generator ikon	Inne generatory	Pozostałe narzędzia
Launcher icons	Device frame generator	Android Action Bar Style Generator
Action bar and tab icons		
Notification icons	Simple nine-patch gen.	
Navigation drawer indicator		Android Holo Colors Generator
Generic icons		

Dodatek C. Wielkość elementów graficznych

P. Co to dpi (znane jako dp, ppi) ?

Skrót od *dots per inch*. Można go wyliczyć korzystając z następującego wzoru:



$$dpi = \sqrt{szerokośćPiksele^2 + wysokośćPiksele^2} / przekątnaCale$$

G1 (320x480)	155.92 dpi	(3.7 in)
Nexus (480x800)	252.15 dpi	
HTC One (1080x1920)	468 dpi	(4.7 in)
Samsung S4 (1080x1920)	441 dpi	(5.5 in)

Q. Jaka jest różnica między dp, dip oraz sp?

dp *Density-independent Pixels* – Abstrakcyjna metryka oparta na faktycznej gęstości upakowania pikseli ekranu. Należy wykorzystywać do każdego zasobu prócz czcionek.

sp *Scale-independent Pixels* – podobnie jak w przypadku dp, ale wykorzystywany do ustawienia wielkości **czcionki**.

Dodatek D. Rozdzielcość ekranu

Rozdzielcość ekranu – jak Android sobie z tym radzi?

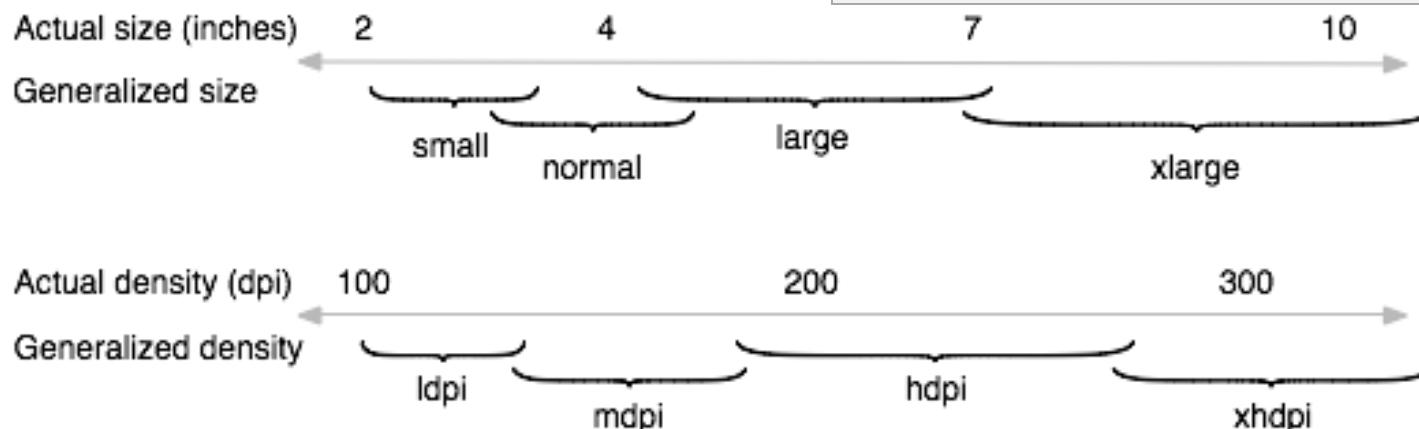
Poniżej znajdują się zapisy jak Android radzi sobie z klasyfikowaniem różnych rozdzielcości ekranu.

Zbiór czterech klas wielkości ekranu

xlarge ekran ma minimum 960dp x 720dp
large ekran ma minimum 640dp x 480dp
normal ekran ma minimum 470dp x 320dp
small ekran ma minimum 426dp x 320dp

Ogólne klasy gęstości:

ldpi ~120dpi (low)
mdpi ~160dpi (medium)
hdpi ~240dpi (high)
xhdpi ~320dpi (extra-high)
xxhdpi ~480dpi (extra-extra-high)
Xxxhdpi ~640dpi (extra-extra-extra-high)



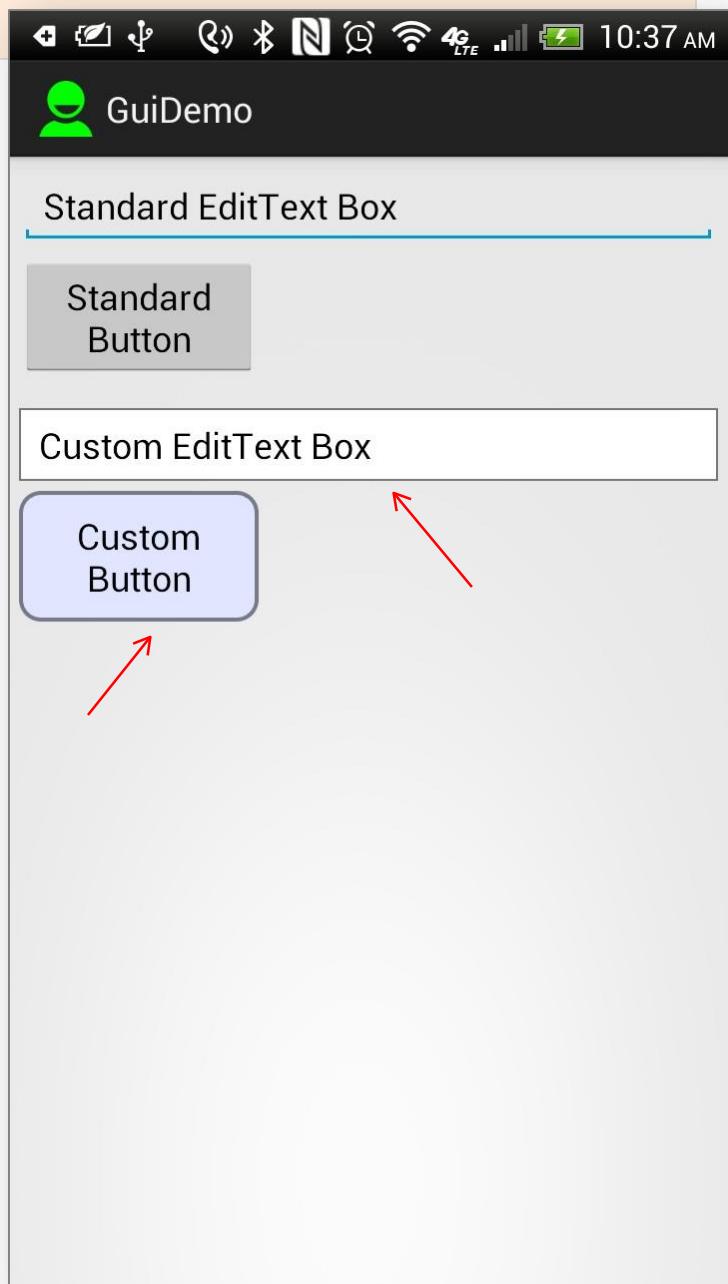
Dodatek E. Hierarchy Viewer

Narzędzie HierarchyViewer umożliwia przejrzenie faktycznej hierarchii komponentów na interfejsie graficznym.

The screenshot shows the Android Hierarchy Viewer interface. On the left, there is a preview window titled "dump_8448446474296423450.ux" displaying a mobile application's user interface. The app has a title bar "GuiDemo" and a main screen with two sections: "What kind of Coffee?" and "What else do you like in your coffee?". The "What kind of Coffee?" section contains three radio buttons: "Decaf", "Expresso", and "Colombian", with "Colombian" being selected. The "What else do you like in your coffee?" section contains two checkboxes: "Cream" and "Sugar", neither of which is checked. At the bottom is a large grey button labeled "Pay". A red dashed rectangle highlights the entire UI area in the preview window. On the right side of the interface, the "Hierarchy" tab is active, showing a tree view of the UI components. The root node is a FrameLayout containing a LinearLayout with a TextView and a RadioGroup. The RadioGroup contains three RadioButtons: Decaf, Expresso, and Colombian. Below the RadioGroup is another TextView. The "Node Detail" table provides specific information about the selected node, which is a LinearLayout with index 0, class android.widget.LinearLayout, package csu.matos.guidemo, and content-desc null. The table also lists properties such as checkable (false), checked (false), clickable (false), enabled (true), focusable (false), and focused (false). At the top of the Hierarchy tab, there is a toolbar with various icons, one of which is highlighted with a red box and a yellow arrow pointing to it from the top right corner of the slide.

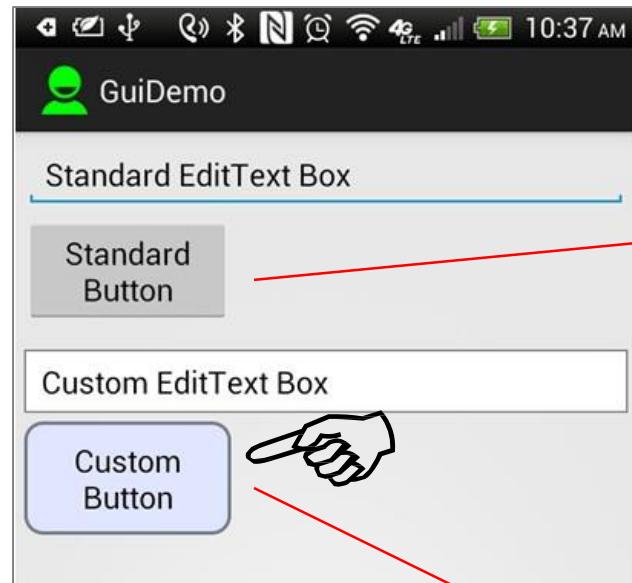
Dodatek F. Zmiana wyglądu widżetu

1. Zmiana wyglądu widżetu jest możliwa w bardzo prosty sposób. Można zmienić kształt, obramowanie, kolor, marginesy i inne właściwości.
2. Podstawowe figury to: prostokąt, elipsa, linia, pierścień.
3. Prócz zmian typowo wizualnych w domyślnym stanie, można wprowadzić zmiany do stanów jak: kliknięty, posiada focus itp.
4. Rysunek przedstawia EditText i Button prezentowane w sposób standardowy na urządzeniu z androidem 4.4. Dolna część przedstawia te komponenty po odpowiednich zmianach.

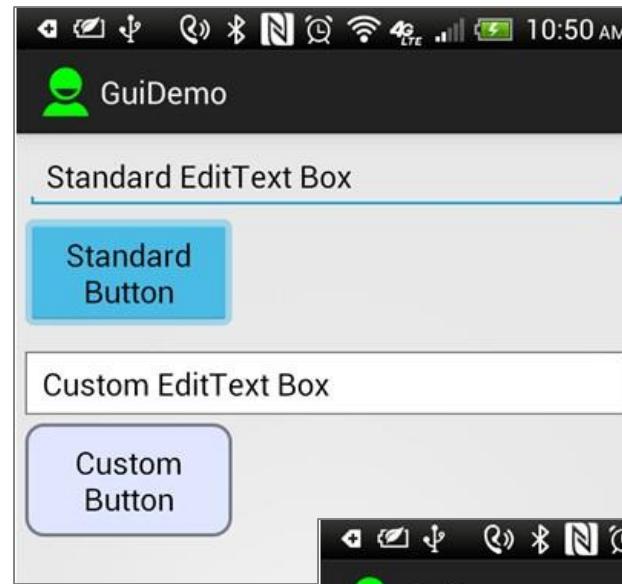


Dodatek F. Zmiana wyglądu widżetu

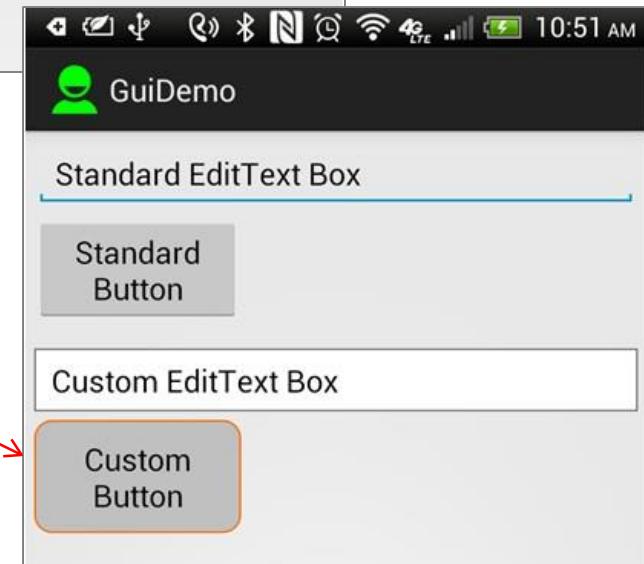
Zmiana wyglądu z podziałem na stany przycisku



Standardowe zachowanie

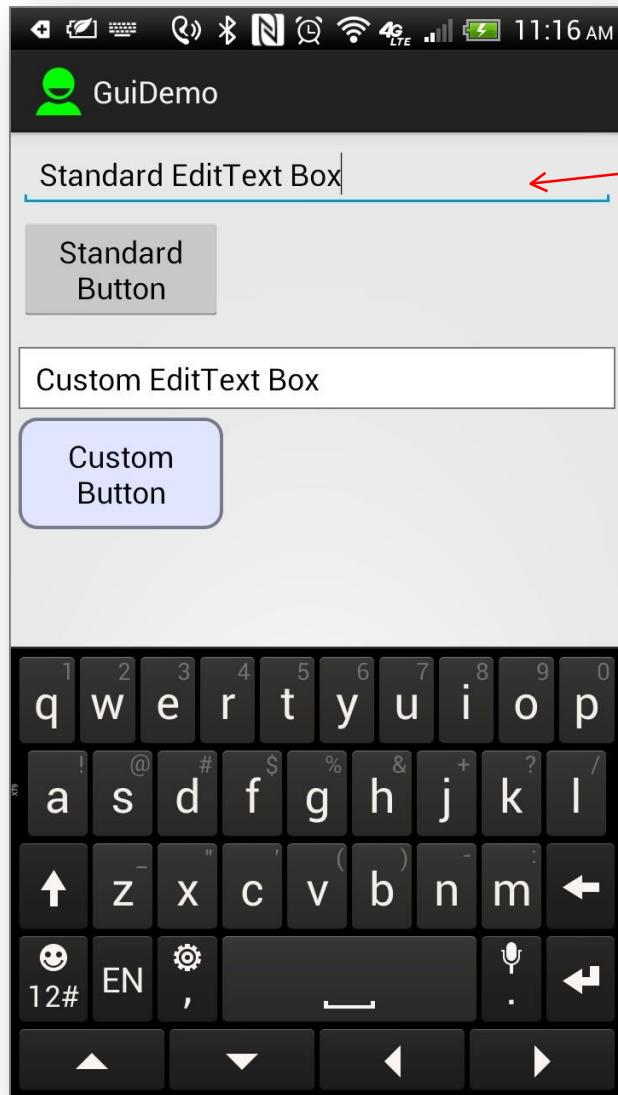


Zachowanie ustalone przez programistę



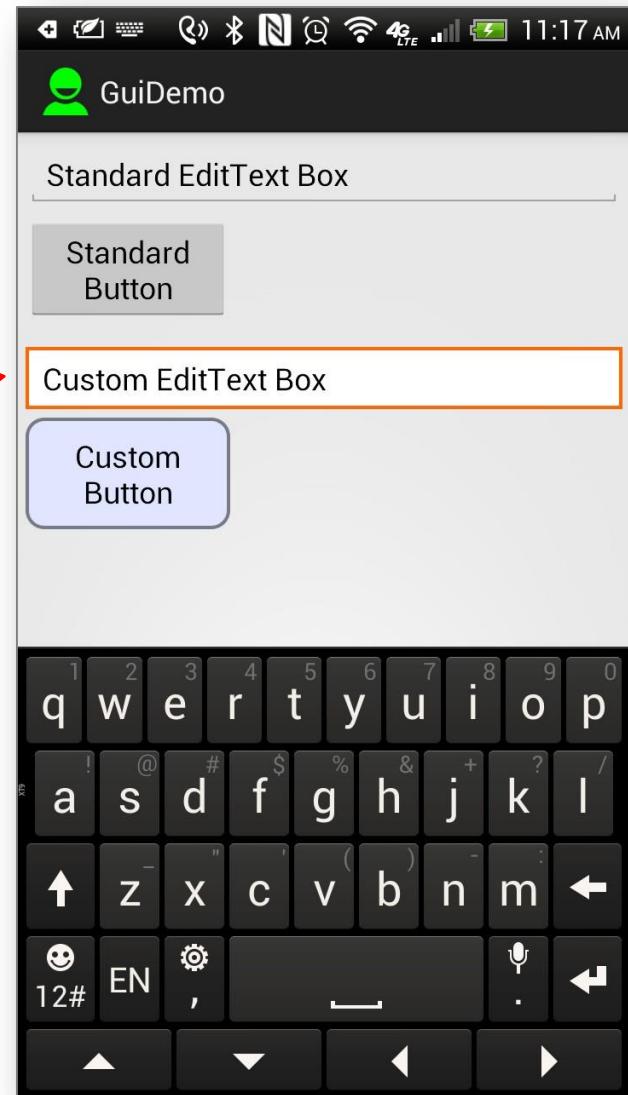
Dodatek F. Zmiana wyglądu widżetu

Zmiana wyglądu, gdy pole tekstowe posiada focus



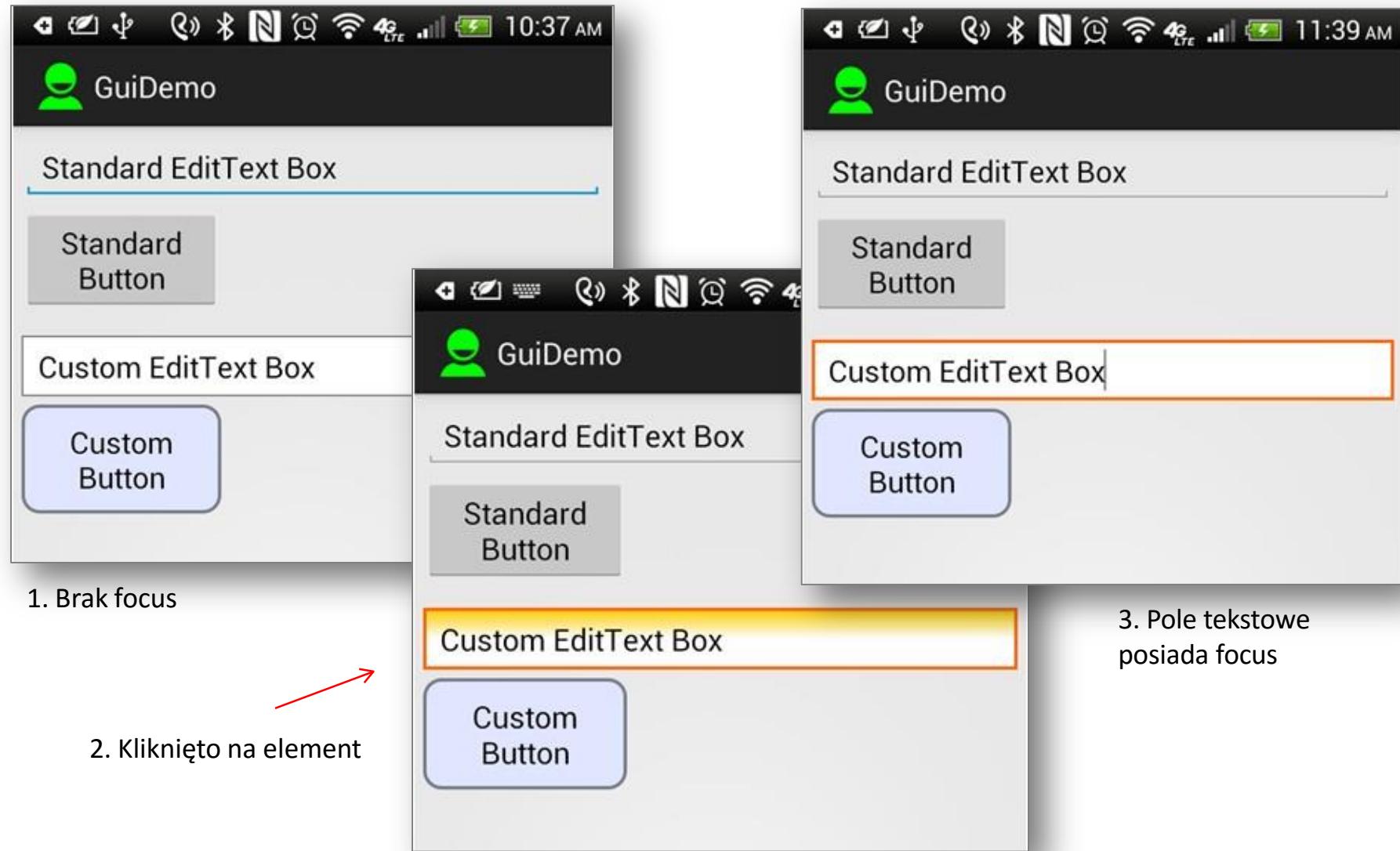
Standardowe
zachowanie

Zdefiniowane
przez
programistę



Dodatek F. Zmiana wyglądu widżetu

Zmiana wyglądu, gdy użytkownik kliknął w pole tekstowe – użycie gradientu.

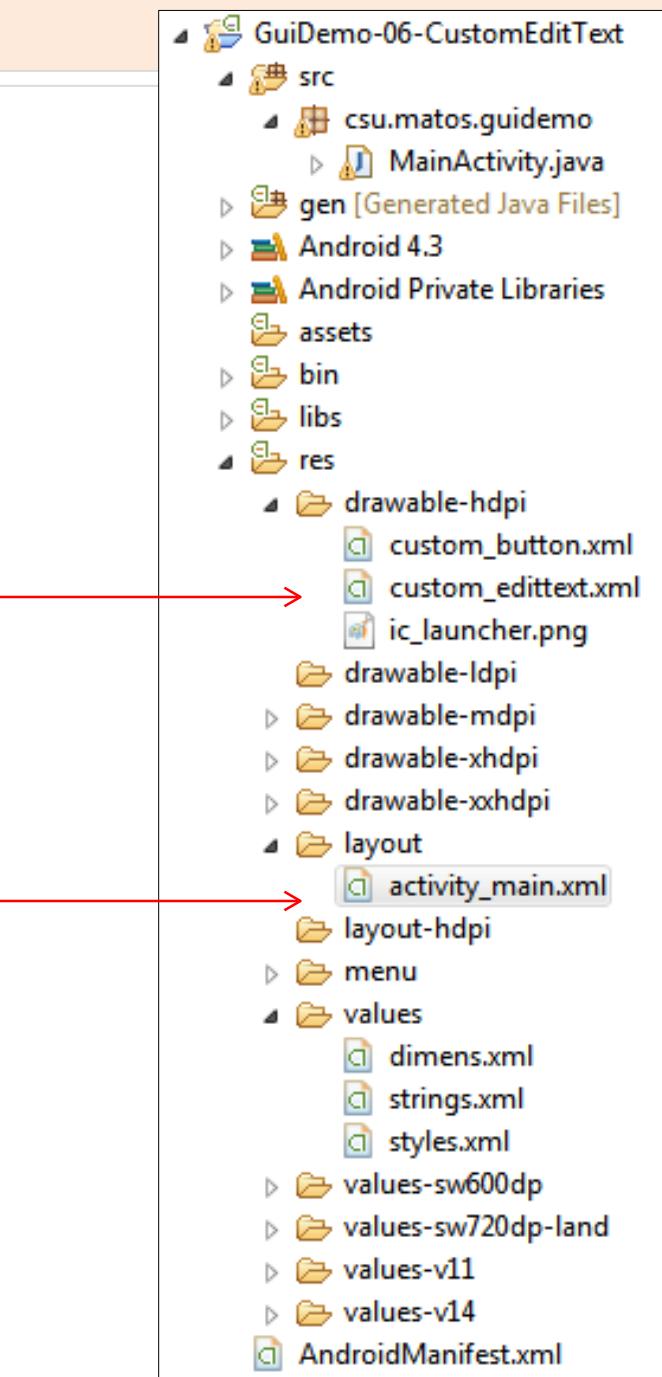


Dodatek F. Zmiana wyglądu widżetu

Organizing the application

Własne szablony wyglądu komponentów

Główny wygląd aktywności



Dodatek F. Zmiana wyglądu widżetu

Układ

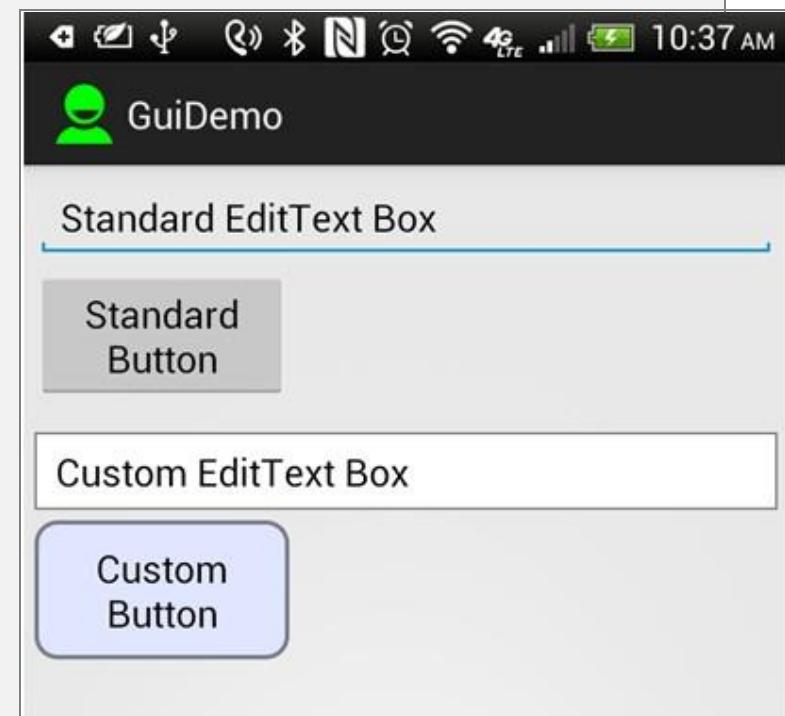
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp" >

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="5dp"
        android:ems="10"
        android:inputType="text"
        android:text="@string/standard_edittext" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/button1"
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="15dp"
        android:text="@string/standard_button" />

```

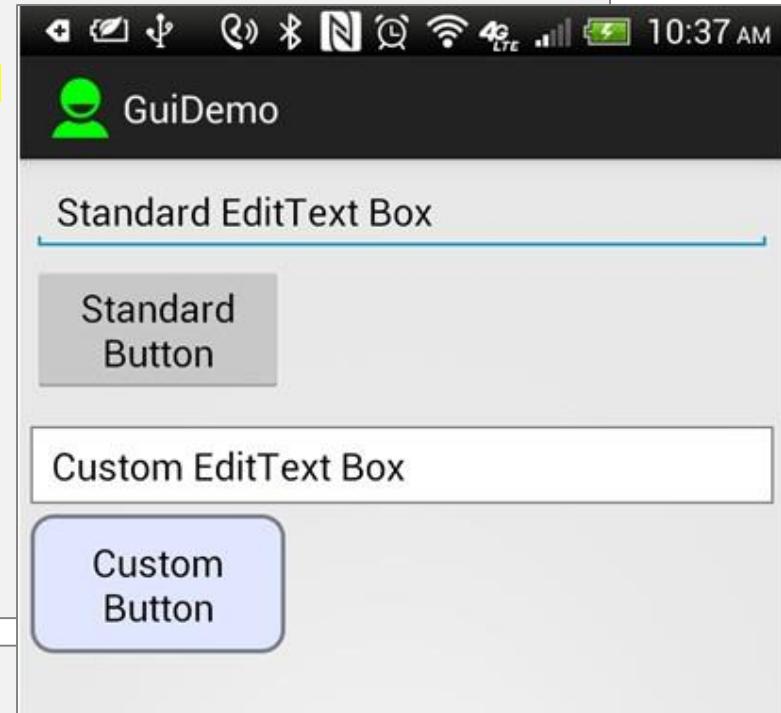


Dodatek F. Zmiana wyglądu widżetu

Układ oraz Resource: res/values/strings

```
<EditText  
    android:id="@+id/editText2"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="5dp"  
    android:background="@drawable/custom_edittext"  
    android:ems="10"  
    android:inputType="text"  
    android:text="@string/custom_edittext" />  
  
<Button  
    android:id="@+id/button2"  
    android:layout_width="120dp"  
    android:layout_height="wrap_content"  
    android:background="@drawable/custom_button"  
    android:text="@string/custom_button" />  
  
</LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string name="app_name">GuiDemo</string>  
    <string name="action_settings">Settings</string>  
    <string name="standard_button">Standard Button</string>  
    <string name="standard_edittext">Standard EditText Box</string>  
    <string name="custom_button">Custom Button</string>  
    <string name="custom_edittext">Custom EditText Box</string>  
</resources>
```



Dodatek F. Zmiana wyglądu widżetu

Zasób: res/drawable/custom_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:state_pressed="true">
        <shape android:shape="rectangle">
            <corners android:radius="10dp"/>
            <solid android:color="#ffc0c0c0" />
            <padding android:left="10dp"
                      android:top="10dp"
                      android:right="10dp"
                      android:bottom="10dp"/>
            <stroke android:width="1dp" android:color="#ffff6600"/>
        </shape>
    </item>
    <item android:state_pressed="false">
        <shape android:shape="rectangle">
            <corners android:radius="10dp"/>
            <solid android:color="#ffE0E6FF"/>
            <padding android:left="10dp"
                      android:top="10dp"
                      android:right="10dp"
                      android:bottom="10dp"/>
            <stroke android:width="2dp" android:color="#ff777B88"/>
        </shape>
    </item>
</selector>
```



Dodatek F. Zmiana wyglądu widżetu

Zasób: res/drawable/custom_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

<item android:state_pressed="true">
    <shape android:shape="rectangle">
        <gradient
            android:angle="90"
            android:centerColor="#FFfffff"
            " android:endColor="#FFffcc00"
            android:startColor="#FFfffff"
            android:type="Linear" />

        <stroke android:width="2dp"
            android:color="#FFff6600" />
        <corners android:radius="0dp" />
        <padding android:left="10dp"
            android:top="6dp"
            android:right="10dp"
            android:bottom="6dp" />
    </shape>
</item>
```

Custom EditText Box

Dodatek F. Zmiana wyglądu widżetu

Zasób: res/drawable/custom_button.xml

```
<item android:state_focused="true">
    <shape>
        <solid    android:color="#FFffffffff" />
        <stroke   android:width="2dp" android:color="#FFff6600" />
        <corners android:radius="0dp" />
        <padding  android:left="10dp"
                    android:top="6dp"
                    android:right="10dp"
                    android:bottom="6dp" />
    </shape>
</item>

<item>
    <!-- state: "normal" not-pressed & not-focused -->
    <shape>
        <stroke   android:width="1dp" android:color="#ff777777" />
        <solid    android:color="#ffffffff" />
        <corners android:radius="0dp" />
        <padding  android:left="10dp"
                    android:top="6dp"
                    android:right="10dp"
                    android:bottom="6dp" />
    </shape>
</item>
</selector>
```

Custom EditText Box

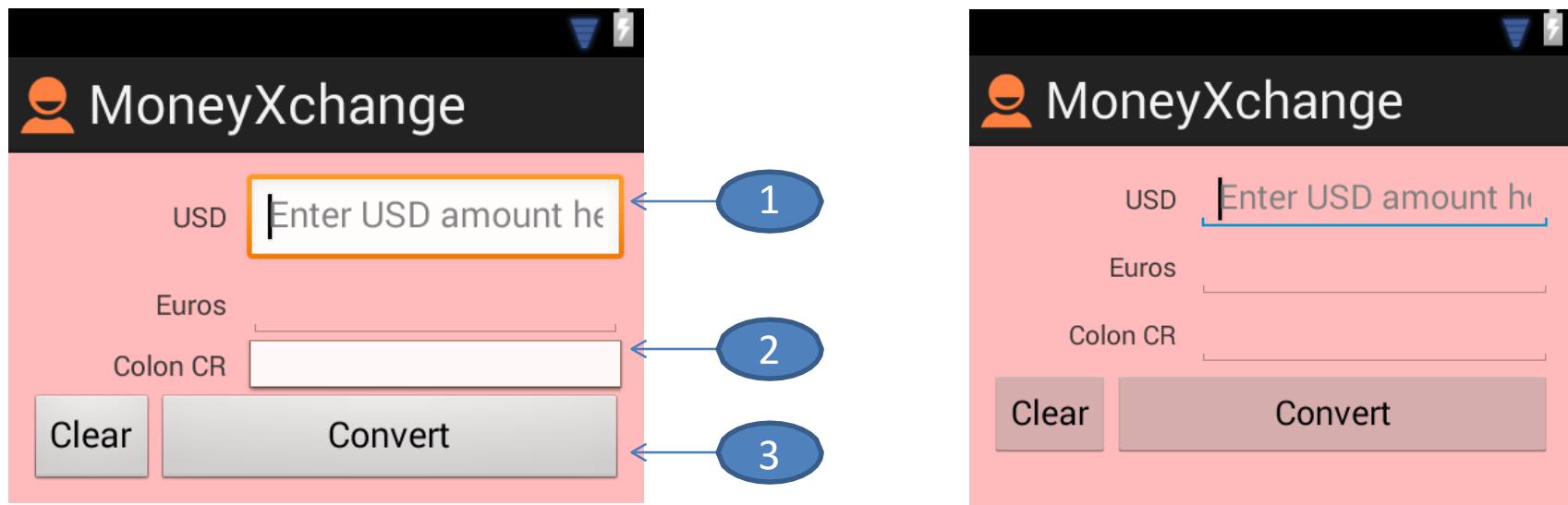
Custom EditText Box

Dodatek G. Problemy z tłem

Zmiana koloru tła może ograniczyć się do zmiany odpowiedniej właściwości w pliku XML
`android:background="#44ff0000"` (pół-przeźroczysty czerwony).

W zależności od skórki (theme) urządzenia może zdarzyć się problem z wypełnieniem komponentów – niektóre z nich przejmują globalny kolor tła.

Rozwiązanie jest bardzo proste – nadanie tym komponentom innego koloru tła poprzez wykorzystanie atrybutu `android:background`.



1. `android:background="@android:drawable/edit_text"`
2. `android:background="@android:drawable/editbox_dropdown_light_frame"`
3. `android:background="@android:drawable/btn_default"`

Programowanie urządzeń mobilnych

Wyświetlanie danych na listach

Tłumaczenie i adaptacja materiałów: dr Tomasz Xięski.
Na podstawie prezentacji udostępnionych przez Victor Matos, Cleveland State University.

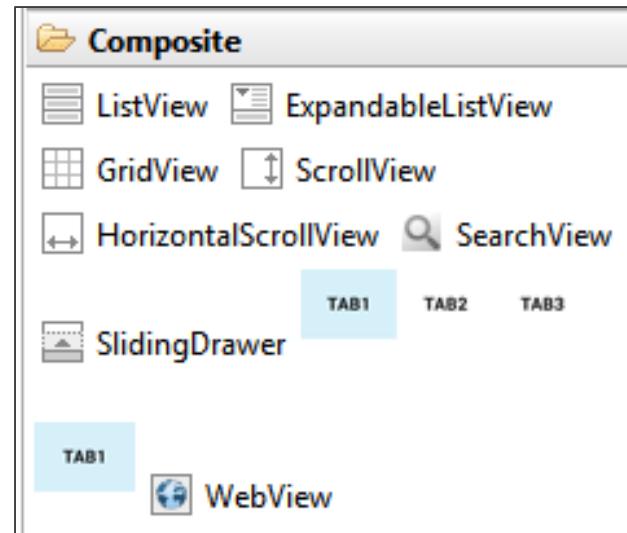
Portions of this page are reproduced from work created and [shared by Google and](#) used according to terms

Widżety listowe

- Widżety **RadioButton** oraz **CheckBox** są dobre przy wyborze małej liczby możliwych opcji.

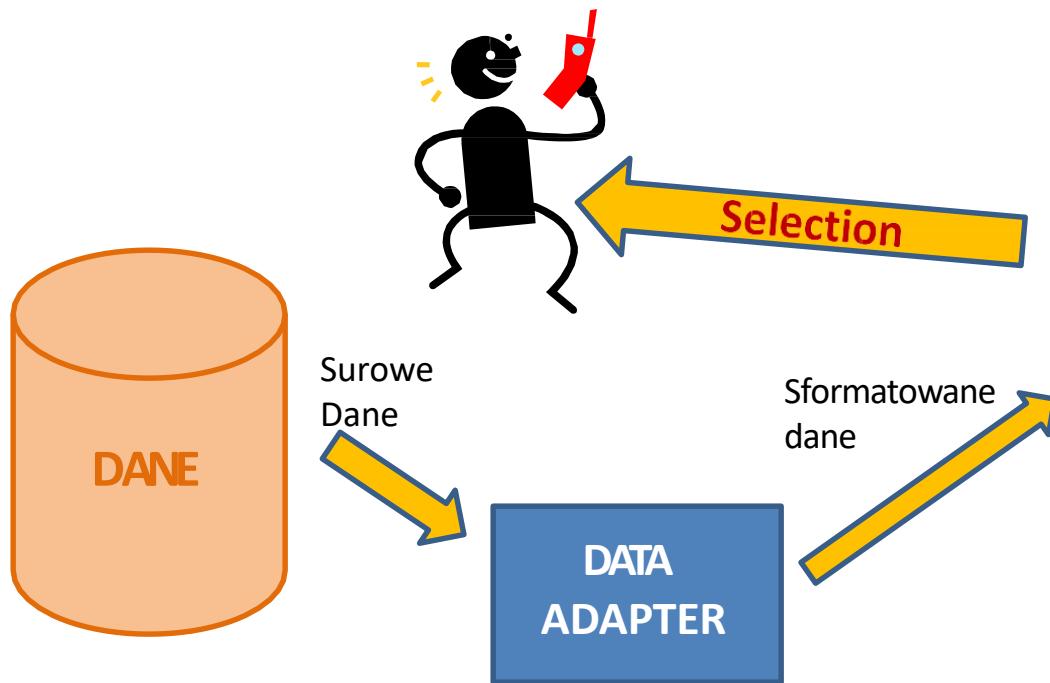


- Dla większej liczby opcji inne widżety oparte o **list** są bardziej adekwatne.
- Przykłady widżetów opartych o **listę** to:
 - *ListView*,
 - *Spinner*,
 - *GridView*
 - *Image Gallery*
 - *ScrollViews*, etc.

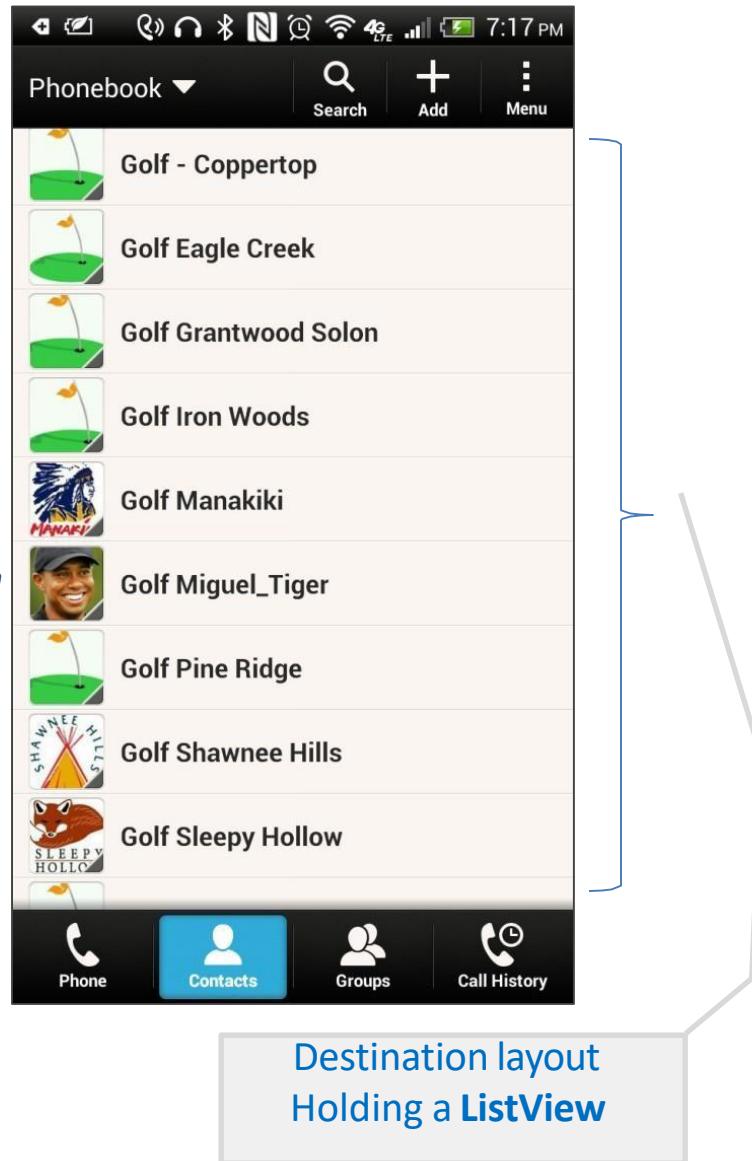


Widżety listowe

Przekazywanie danych



- Klasa **data adapter** jest wykorzystywana do populowania danymi listy.
- Surowe dane dla adaptera mogą pochodzić z wielu różnych źródeł, jak małe tablice czy też duże bazy danych.

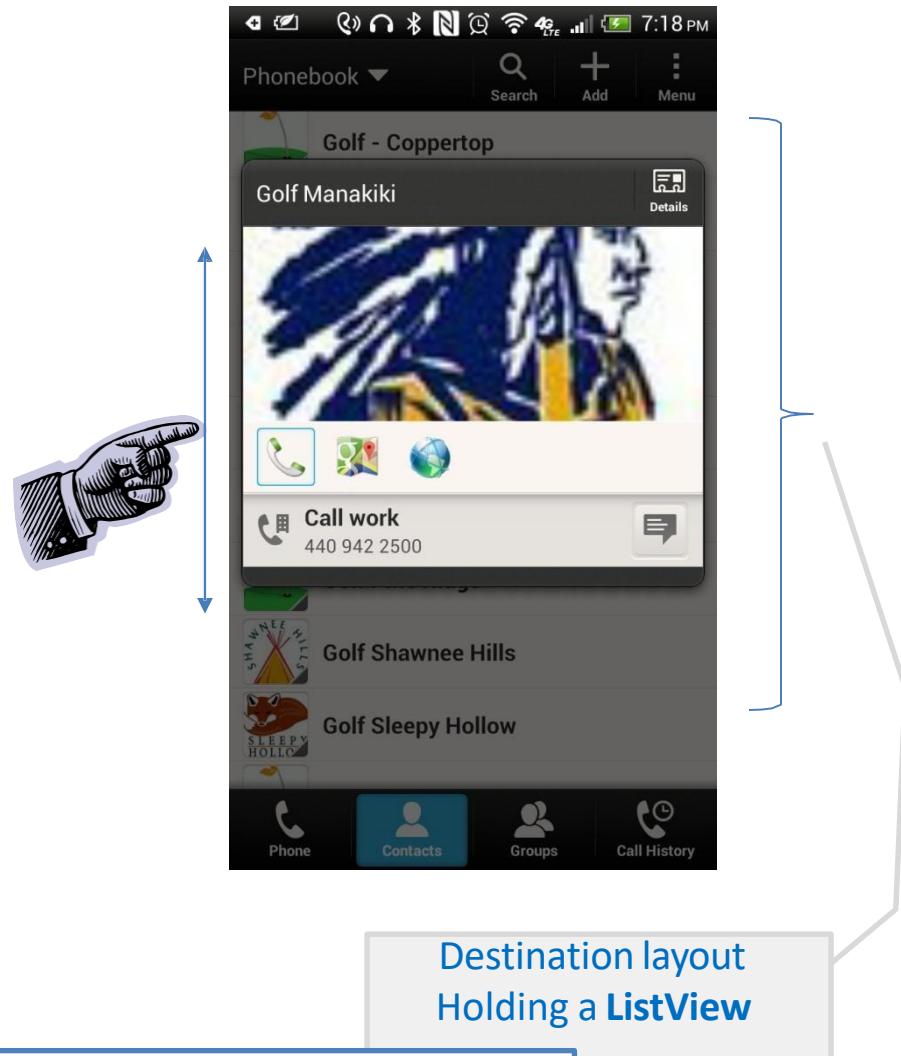


Widżety listowe

Widżet **ListView** jest najpopularniejszym widżetem, który umożliwia wyświetlanie danych gromadzonych przez **data adapter**.

ListView mają wbudowane paski **przewijania**, więc nie wszystkie wiersze muszą być widoczne.

Użytkownik **dotyka** **wiersza** powodując wybór.



Rekomendowany zamiennik RecyclerView

<https://developer.android.com/guide/topics/ui/layout/recyclerview>

ArrayAdapter

- **ArrayAdapter<T>** jest rzeczywistą implementacją abstrakcyjnego typu BaseAdapter class.
- Jego zadaniem jest pobranie tablicy elementów (dowolnego typu T), przetworzenie poszczególnych elementów poprzez wywołanie metody **toString()** oraz przekazanie sformatowanego wyjścia do **TextView**. Formatowanie wykonywane jest na podstawie dołączonej specyfikacji w formie pliku XML.
- **ArrayAdapter<T>** jest głównie wykorzystywany dla obiektów typu **<String>**. Dla innych typów danych należy przeciążyć metodę **toString()** by wyznaczyć jaki tekst będzie prezentowany dla konkretnego elementu listy.
- By ListView pokazywał bardziej skomplikowaną aranżację wizualną – tekst i obrazki – należy stworzyć własny adapter, którego metoda **getView(...)** określa jak należy stworzyć i wypożycjonować każdy element interfejsu.

ArrayAdapter

Dane źródłowe - array or java.util.List
{ object₁, object₂, ..., object_n }

Wyjście: ‘Ładne’ GUI



Specyfikacja XML

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

ArrayAdapter

Typowe użycie ArrayAdapter<String>

```
String[] items = { "Data-0", "Data-1", "Data-2", "Data-3",
                   "Data-4", "Data-5", "Data-6", "Data-7" };

ArrayAdapter<String> adapter = new ArrayAdapter<String>(
    this,
    android.R.layout.simple_list_item_1,
    items );
```

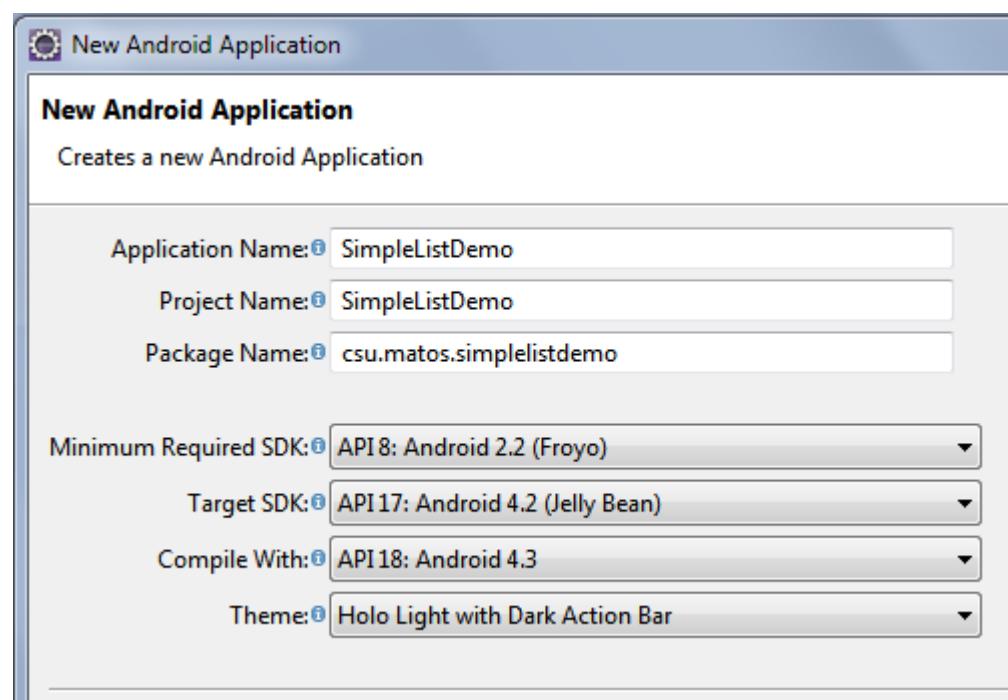
Parameters:

1. Kontekst danej aktywności (**this**)
2. Układ dla **TextView** określający jak każdy wiersz listy jest formatowany (**android.R.id.simple_list_item_1**).
3. Właściwe **źródło danych** (tablica lub Java.List zawierająca elementy, które należy zaprezentować).

Przykład 1: Wykorzystanie ArrayAdapter

Prosta lista

Celem przykładu jest wyświetlenie listy elementów tekstowych. Dane przechowywane są w prostej tabeli typu String[]. Każdy wiersz listy pokazuje poszczególny element tablicy źródłowej. Jeżeli użytkownik zaznaczy dany element, na ekranie pokazywana jest jego wartość oraz indeks z tablicy źródłowej.



Przykład 1: Wykorzystanie ArrayAdapter

Układ XML

```
<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="vertical">

    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#fffff00"
        android:text="Using ListView..." 

        android:textSize="16sp" />

    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" 
    >

    <TextView
        android:id="@+id/empty"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffff0000"
        android:text="empty list"  />

</LinearLayout>
```

Istotne jest nazewnictwo komponentów:

@android:id/list
@android:id/empty

Standardowy układ

Dla pustych list

Przykład 1: Wykorzystanie ArrayAdapter

Układ XML

```
package csu.matos;  
  
import ...  
  
public class ListViewDemo extends ListActivity {  
  
    TextView txtMsg;  
  
    String[] items = { "Data-0", "Data-1", "Data-2", "Data-3",  
                      "Data-4", "Data-5", "Data-6", "Data-7" };  
  
    // next time try an empty list such as:  
    // String[] items = {};
```



UWAGA:

ListActivity to nie to samo co Activity. Przypisana jest do ListView

Yellow arrow pointing to the code line: // next time try an empty list such as:
// String[] items = {};



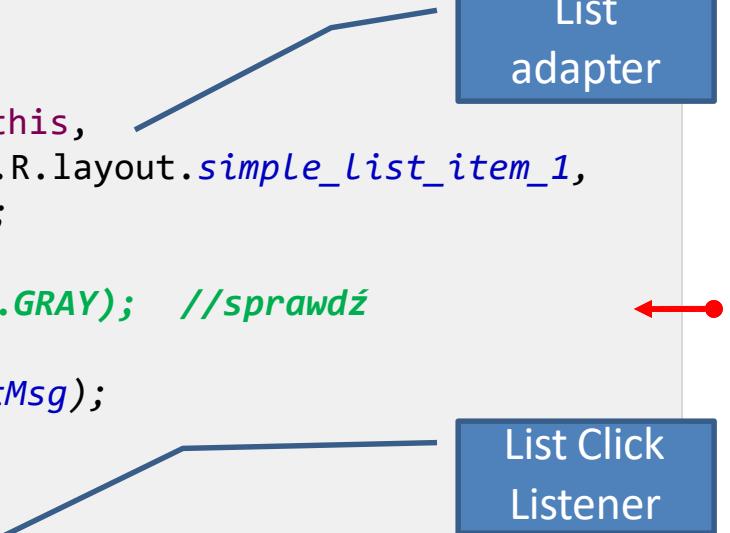
Źródło
danych

Uwaga: Klasa **ListActivity** jest niejawnie powiązana z obiektem o nazwie [@android:id/list](#)

Przykład 1: Wykorzystanie ArrayAdapter

Klasa Main Activity

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    setListAdapter(new ArrayAdapter<String>(this,  
                                         android.R.layout.simple_list_item_1,  
                                         items));  
  
    //getListView().setBackgroundColor(Color.GRAY); //sprawdź  
  
    txtMsg = (TextView) findViewById(R.id.txtMsg);  
}  
  
@Override  
protected void onListItemClick(ListView l, View v, int position, long id) {  
    super.onListItemClick(l, v, position, id);  
    String text = " Position: " + position + " " + items[position];  
    txtMsg.setText(text);  
}  
}
```

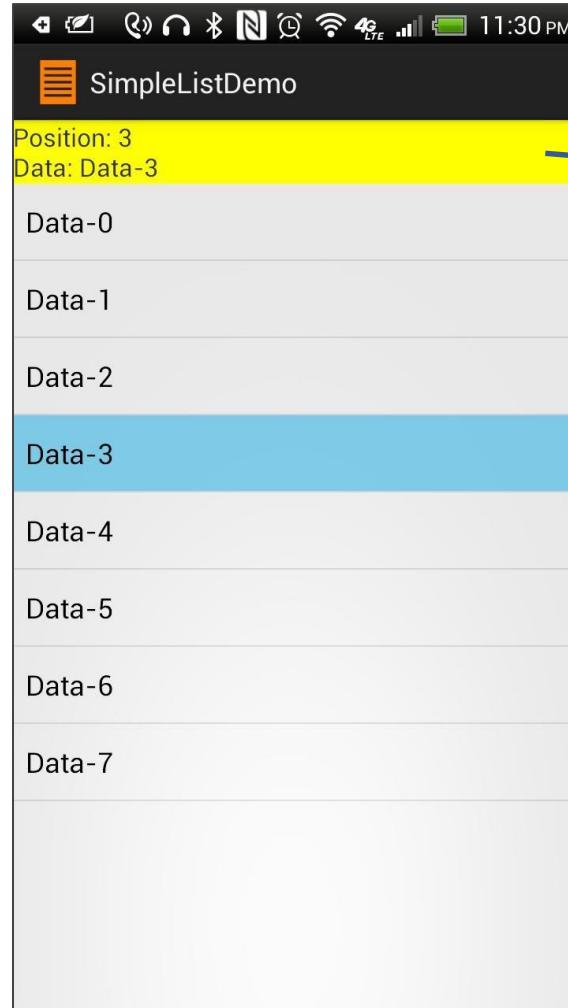
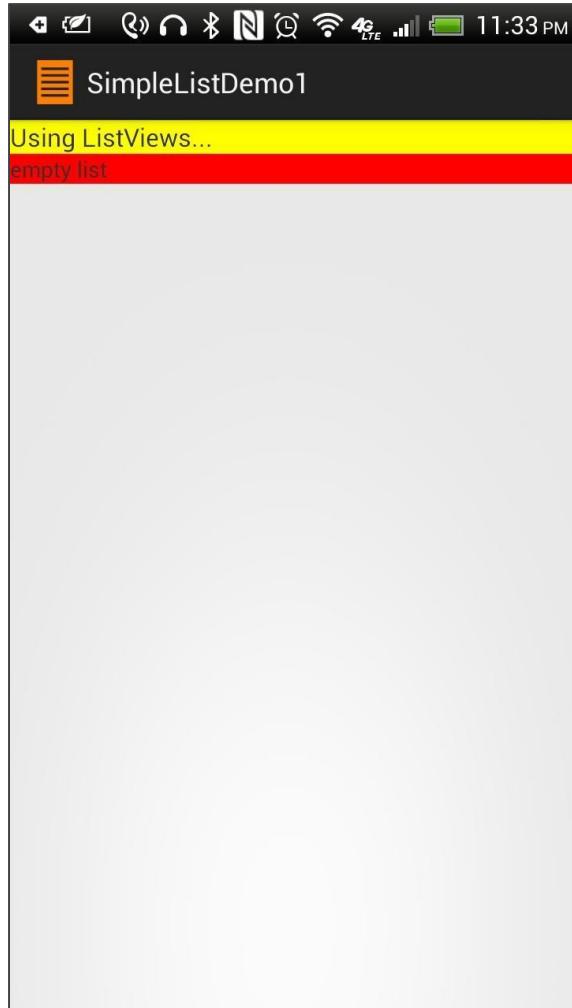


List adapter

List Click Listener

13

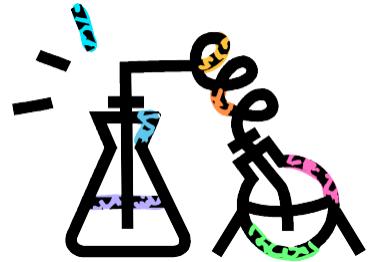
Przykład 1: Wykorzystanie ArrayAdapter



Wybrany element

Tło zmienia się na niebieskie by potwierdzić wybór użytkownika

Przykład 1: Wykorzystanie ArrayAdapter



Szybka zmiana grafiki

1. Otwórz plik `AndroidManifest.xml`. Pod tagiem `<Application>` znajdź klauzulę `android:theme="@style/AppTheme"`

2. Zmień znalezioną linię na:

`android:theme="@android:style/Theme.Black"`

3. Spróbuj innych wariantów:

`Theme.DeviceDefault`

`Theme.Dialog`

`Theme.Holo`

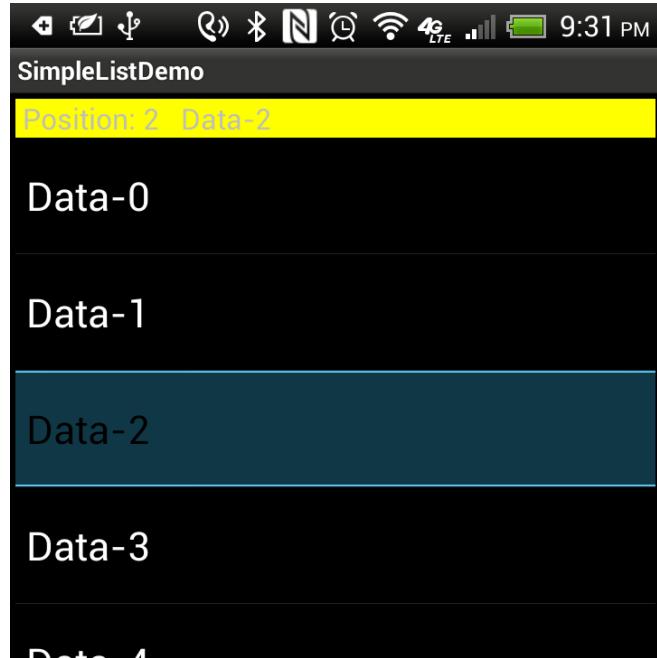
`Theme.Light`

`Theme.Panel`

`Theme.Translucent`

`Theme.Wallpaper`

itp.



Przykład 1B: Wykorzystanie ArrayAdapter

Alternatywna wersja przykładu 1

- Można używać zwykłej klasy **Activity** zamiast **ListActivity**.
- Układ poniżej wykorzystuje listę o identyfikatorze `@+id/my_list` (zamiast `@android:id/list` jak w poprzednim przypadku).

Przykład 1B – Układ XML

```
<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="vertical" >

    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#fffff00"
        android:text="Using ListViews..."
        android:textSize="16sp" />

    <ListView android:id="@+id/my_List"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </ListView>

</LinearLayout>

</LinearLayout>
```



Przykład 1B: Wykorzystanie ArrayAdapter

Alternatywna wersja przykładu 1

Przykład 1B – Klasa MainActivity



```
public class ListViewDemo2 extends Activity {
    String[] items = { "Data-0", "Data-1", "Data-2", "Data-3",
                      "Data-4", "Data-5", "Data-6", "Data-7" };
    ListView myListView;
    TextView txtMsg;

    @Override
    public void onCreate(Bundle savedInstanceState)
    { super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        myListView = (ListView) findViewById(R.id.my_list);

        ArrayAdapter<String> aa = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, ←
                // R.layout.my_text,   //try this
                later... items);
        myListView.setAdapter(aa);

        txtMsg = (TextView) findViewById(R.id.txtMsg);
    } //onCreate
}
```

Annotations in the code:

- A red circle highlights the word `Activity` in the first line of the class definition.
- Two red arrows point from the text "←" to the parameter of the `ArrayAdapter` constructor, specifically the line `android.R.layout.simple_list_item_1,`.
- One red arrow points from the text "←" to the line `// R.layout.my_text, //try this`.

Przykład 1B: Wykorzystanie ArrayAdapter

Alternatywna wersja przykładu 1

Example 1B – MainActivity Class

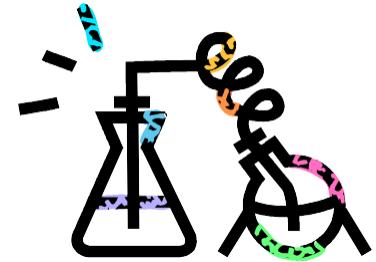
```
myListView.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> av, View v,
                        int position, long id) {

        String text = "Position: " + position
                    + "\nData: " + items[position];

        txtMsg.setText(text);
    }
});
```

Należy pamiętać o dodaniu nastuchiwacza dla zdarzenia kliknięcia elementu listy w metodzie **onCreate**.

Przykład 1C: Wykorzystanie ArrayAdapter



Alternatywna wersja przykładu 1

Wygląd ListView można dowolnie zmieniać zgodnie z własnymi preferencjami.

Przykładowo, należy zmienić `android.R.layout.simple_list_item_1` na `R.layout.my_text` gdzie `my_text` to układ zgodny z zaprezentowanym poniżej (przechowywany w `res/layout`).

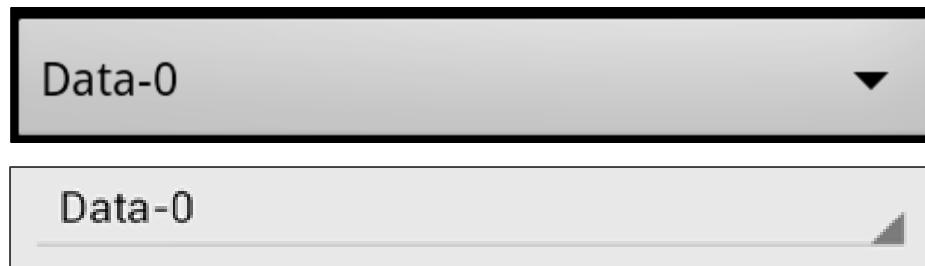
```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="2dp"
    android:paddingTop="5dp"
    android:paddingLeft="50dp"
    android:textColor="#ffff0000"
    android:background="#22ff0000"
    android:textSize="35sp" />
```



Wskazówka : Od SDK4.0 TextView może również zawierać obrazek (wykorzystując metodę `.setDrawableLeft(some_image)`)

Spinner

Spinner



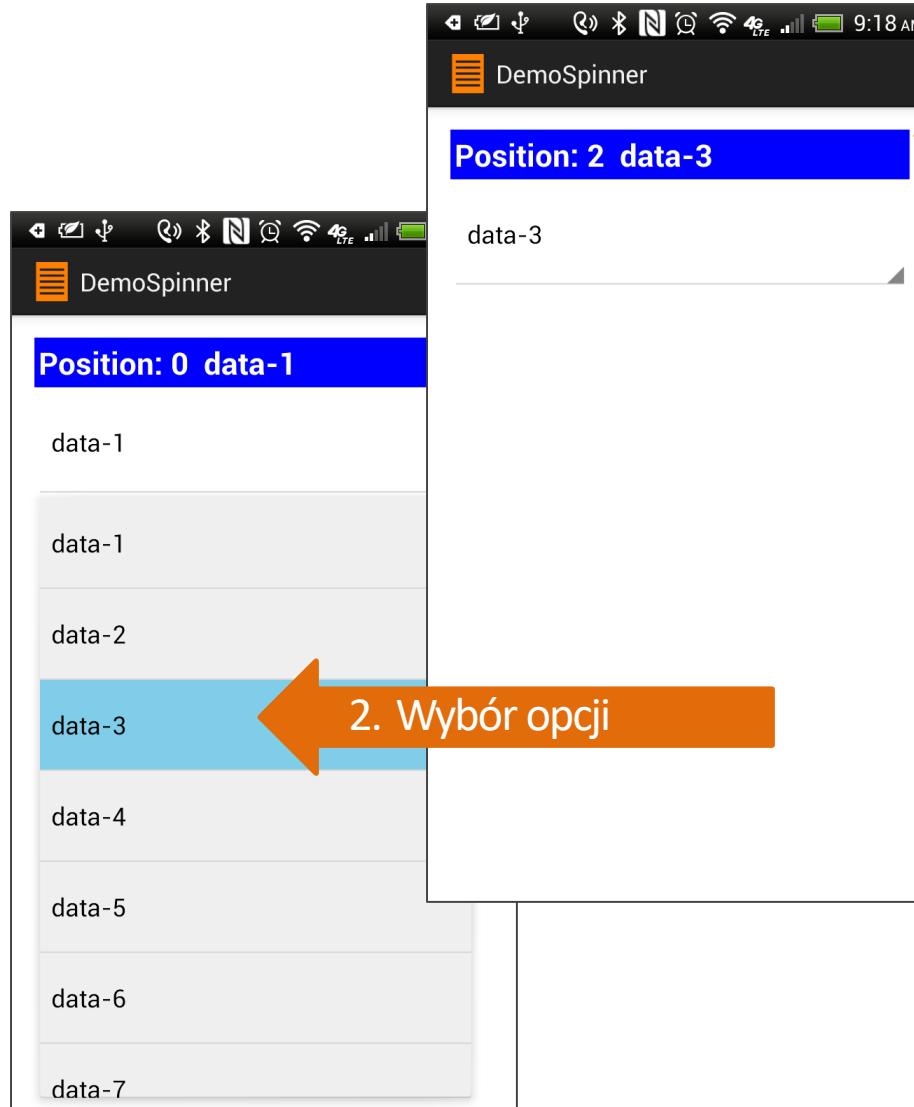
- **Spinner** jest ekwiwalentem *listy rozwijanej*.
- Spinner ma takie same możliwości jak ListView, ale zajmuje mniej miejsca.
- Obsługą źródła danych zajmuje się ArrayAdapter - ***setAdapter(...)***.
- Odpowiedni nasłuchiwacz rejestruje zdarzenia zaznaczenia listy ***setOnItemSelectedListener(...)***.
- Metoda ***setDropDownViewResource(...)*** pokazuje strzałkę rozwijania listy.



Przykład 2: Wykorzystanie Spinner'a



1. Rozwinięcie listy



Przykład 2: Wykorzystanie Spinner'a

Układ XML

```
<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="vertical"
    android:padding="3dp"
    tools:context=".MainActivity" >

    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#fffffff00"
        android:text="@string/hello_world" />

    <Spinner android:id="@+id/spinner1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```



Przykład2: Wykorzystanie Spinner'a

Klasa MainActivity

```
public class MainActivity extends Activity
    implements AdapterView.OnItemSelectedListener{
    // GUI objects
    TextView txtMsg;
    Spinner spinner;

    // options to be offered by the spinner
    String[] items = { "Data-0", "Data-1", "Data-2", "Data-3", "Data-4",
        "Data-5", "Data-6", "Data-7" };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtMsg = (TextView) findViewById(R.id.txtMsg);

        spinner = (Spinner) findViewById(R.id.spinner1);

        // use adapter to bind items array to GUI layout
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(
            this,
            android.R.layout.simple_spinner_dropdown_item,
            items);
```



Przykład2: Wykorzystanie Spinner'a

Klasa MainActivity

```
// bind everything together
spinner.setAdapter(adapter);

// add spinner a listener so user can meake selections by tapping an item
spinner.setOnItemSelectedListener(this);

}

// next two methods implement the spinner's listener
@Override
public void onItemSelected(AdapterView<?> parent, View v, int position,
    long id) {
    // echo on the textbox the user's selection
    txtMsg.setText(items[position]);
}

@Override
public void onNothingSelected(AdapterView<?> arg0) {
    // TODO do nothing - needed by the interface
}

}
```

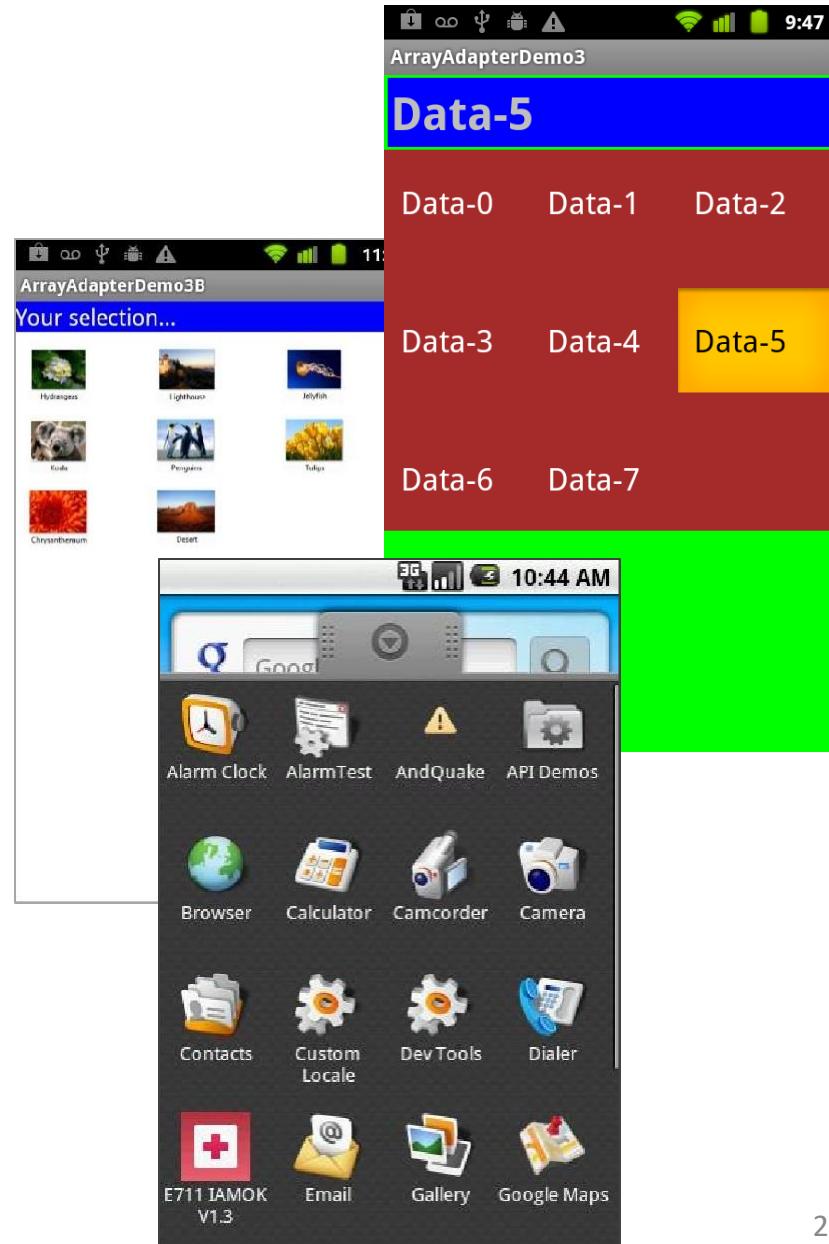
GridView

GridView

GridView jest implementacją **ViewGroup**, która wyświetla elementy w formie 2-wymiarowej siatki z możliwością przewijania.

Poszczególne dane wyświetlane w siatce dostarczane są przez warstwę data adaptera.

Komórki siatki mogą wyświetlać zarówno dane tekstowe jak i grafikę.



GridView

Przydatne właściwości

Poniższe właściwości są wykorzystywane by ustalić liczbę i rozmiar kolumn:

- **android:numColumns**

Określa ile kolumn należy pokazać. Jeżeli wykorzystano stałą “auto_fit”, Android sam ustala liczbę kolumną na podstawie dostępnego wolnego miejsca oraz pozostałych właściwości wymienionych poniżej.

- **android:verticalSpacing** oraz **android:horizontalSpacing**

określa jak dużo wolnego miejsca powinno zostać między elementami w siatce.

- **android:columnWidth**

Szerokość kolumn (w **dip**).

- **android:stretchMode**

Określa jak zmienić rozmiar grafiki, gdy wolne miejsce nie zostało zagospodarowane przez kolumny bądź wymuszone przez spacing.

GridView

Rozciąganie widżetu na cały ekran

Załóżmy, że ekran ma szerokość **320** (dip) pikseli oraz ustawiono właściwości:

android:columnWidth na **100dip** oraz
android:horizontalSpacing na **5dip**.

Trzy kolumny zajmują **310** pikseli (każda kolumna po 100 pikseli oraz dwa pasy pustej przestrzeni po 5 pikseli).

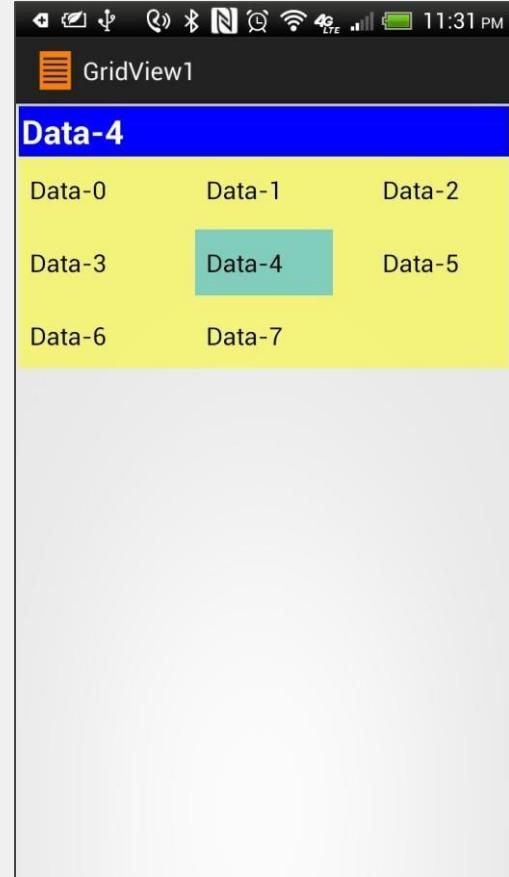
Jeżeli *android:stretchMode* ustawiono na *columnWidth*, każda z 3 kolumn rozciągnięta zostanie o 3-4 piksele by wykorzystać wolną przestrzeń 10 pikseli.

Jeżeli *android:stretchMode* ustawiono na *spacingWidth*, pusta przestrzeń zostanie rozciągnięta o 5 pikseli by zagospodarować pozostałe 10 pikseli.

Przykład 3A: Wykorzystanie GridView

Układ XML

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="2dp"
    tools:context=".MainActivity" >
    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:textSize="24sp"
        android:textStyle="bold"
        android:textColor="#ffffffff"
        android:padding="2dip" />
    <GridView android:id="@+id/grid"
        android:background="#77ffff00"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:verticalSpacing="5dip"
        android:horizontalSpacing="5dip"
        android:numColumns="auto_fit"
        android:columnWidth="100dip"
        android:stretchMode="spacingWidth" />
</LinearLayout>
```



Przykład 3A: Wykorzystanie GridView

Klasa Main Activity

```
public class ArrayAdapterDemo3 extends Activity {  
  
    TextView txtMsg;  
  
    String[] items = { "Data-0", "Data-1", "Data-2", "Data-3",  
                      "Data-4", "Data-5", "Data-6", "Data-7" };  
    @Override  
    public void onCreate(Bundle icicle) {  
        super.onCreate(icicle);  
        setContentView(R.layout.activity_main);  
  
        txtMsg = (TextView) findViewById(R.id.txtMsg);
```

Przykład 3A: Wykorzystanie GridView

Klasa Main Activity

```
GridView grid = (GridView) findViewById(R.id.grid);

ArrayAdapter<String> adapter = new ArrayAdapter<String>(
    this,
    android.R.layout.simple_list_item_1,
    items );

grid.setAdapter(adapter);

grid.setOnItemClickListener(new OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> container, View
                           v, int position, long id) {
        txtMsg.setText(items[position]);
    }
});

} //onCreate

}// class
```

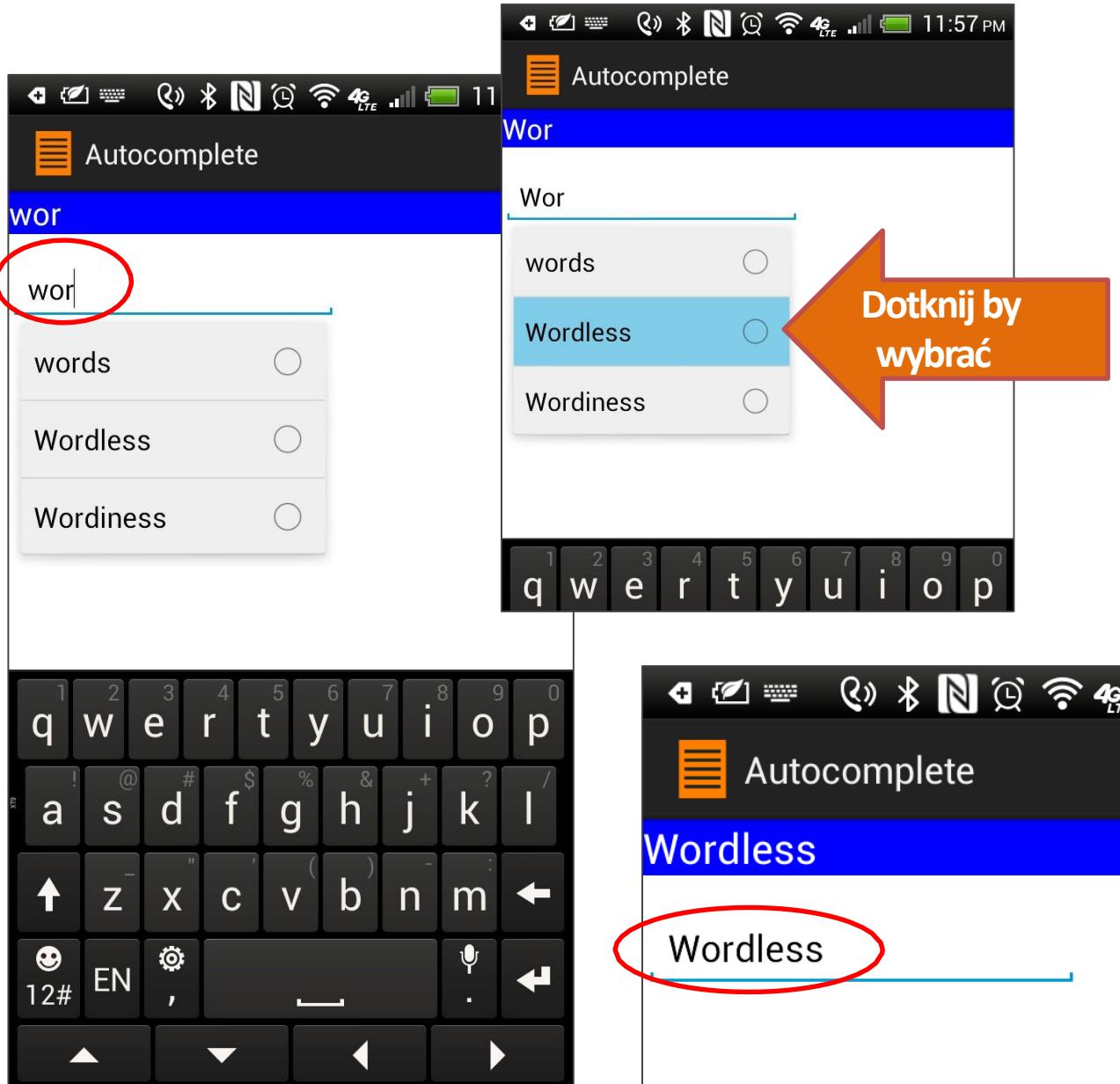
AutocompleteTextView

AutoCompleteTextView

- Jest to bardziej wyspecjalizowany komponent typu EditText.
- Dotychczas wprowadzone znaki są porównywane z początkiem słów przechowywanych w liście dostępnych sugestii.
- Sugestie dopasowane do określonego prefiksu są wyświetlane na *liście wyboru*.
- Użytkownik może wybrać z listy sugestii lub kontynuować wprowadzanie znaków.
- Właściwość **android:completionThreshold** jest wykorzystywana by uaktywnić wyświetlanie listy sugestii. Określa liczbę znaków, którą należy wprowadzić by próbować dopasowania do listy sugestii.

Inne właściwości omawianego widżetu dostępne są w Dodatku B

AutocompleteTextView



Przykład 4.

Słowa zaczynające się od prefiksu “**wor**” lub “**set**” są podstawą do aktywacji listy.

Jeżeli dowolne (3 literowe) prefiksy zostaną wprowadzone mechanizm TextWatcher wyświetli listę sugestii.

AutocompleteTextView

Układ 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" >

    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:textColor="#ffffffff"
        android:background="#ff0000ff" >
    </TextView>

    <AutoCompleteTextView
        android:id="@+id/autoCompleteTextView1"
        android:hint="type here..."
        android:completionThreshold="3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/txtMsg"
        android:layout_marginTop="15dp"
        android:ems="10" />
</RelativeLayout>
```

Wait 3 chars to work

AutocompleteTextView

Klasa Main Activity

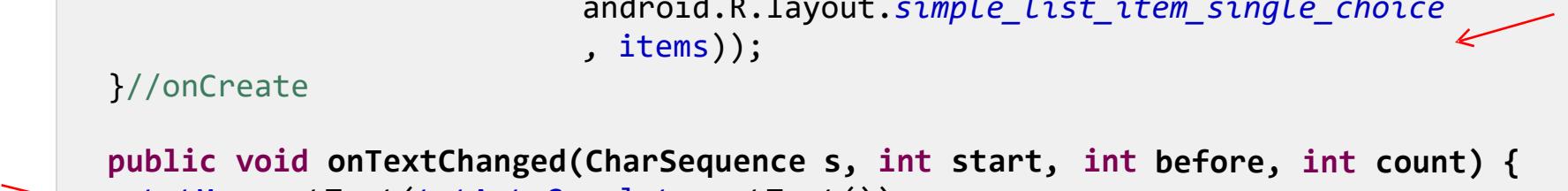


```
public class ArrayAdapterDemo4 extends Activity implements TextWatcher
{
    TextView txtMsg;
    AutoCompleteTextView txtAutoComplete;
    String[] items = { "words", "starting", "with", "set",
        "Setback", "Setline", "Setoffs", "Setouts", "Setters",
        "Setting", "Settled", "Settler", "Wordless", "Wordiness",
        "Adios" };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtMsg = (TextView) findViewById(R.id.txtMsg);
```

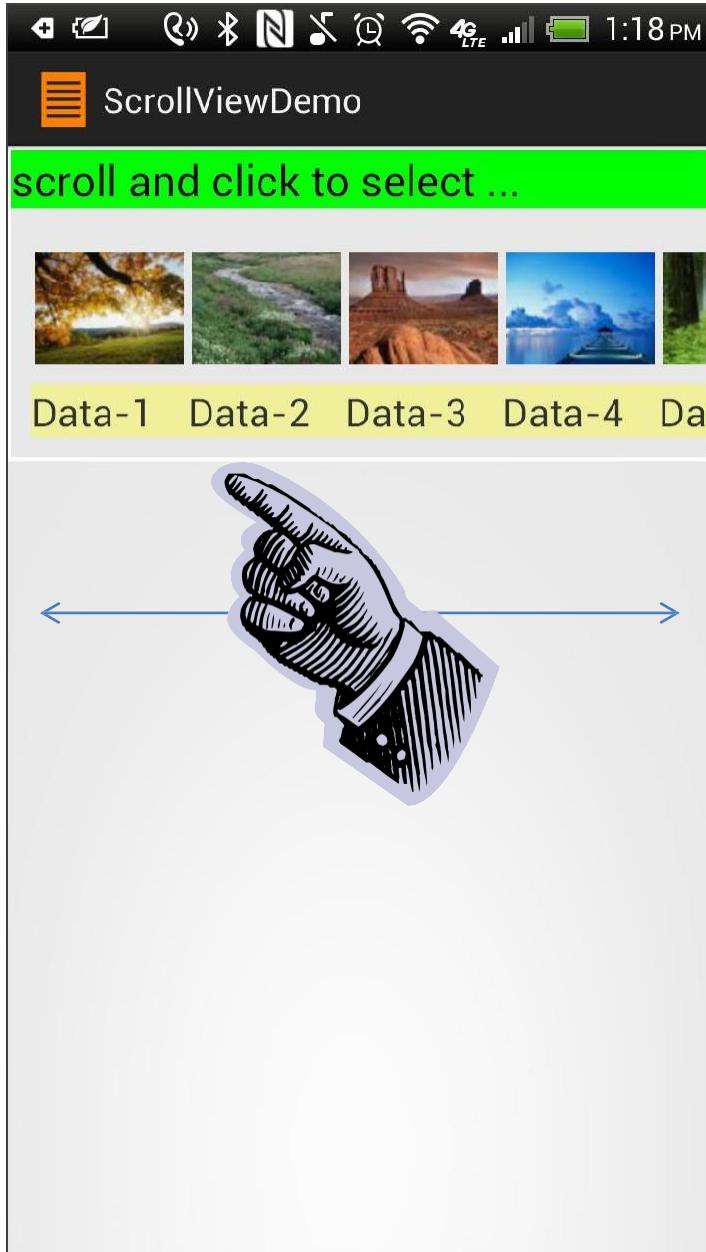
AutocompleteTextView

Klasa Main Activity

```
txtAutoComplete = (AutoCompleteTextView) findViewById(  
                    R.id.autoCompleteTextView1);  
txtAutoComplete.addTextChangedListener(this);  
  
txtAutoComplete.setAdapter(new ArrayAdapter<String>( this,  
                android.R.layout.simple_list_item_single_choice  
                , items));  
}//onCreate  
  
public void onTextChanged(CharSequence s, int start, int before, int count) {  
    txtMsg.setText(txtAutoComplete.getText());  
}  
  
public void beforeTextChanged(CharSequence s, int start, int count,int after) {  
    // needed for interface, but not used  
}  
  
public void afterTextChanged(Editable s) {  
    // needed for interface, but not used  
}  
}//class
```



HorizontalScrollView



Komponent HorizontalScrollView

umożliwia w sposób graficzny dokonać wyboru spośród listy elementów (np. miniatur obrazków).

Użytkownik wchodzi w interakcję z komponentem poprzez:

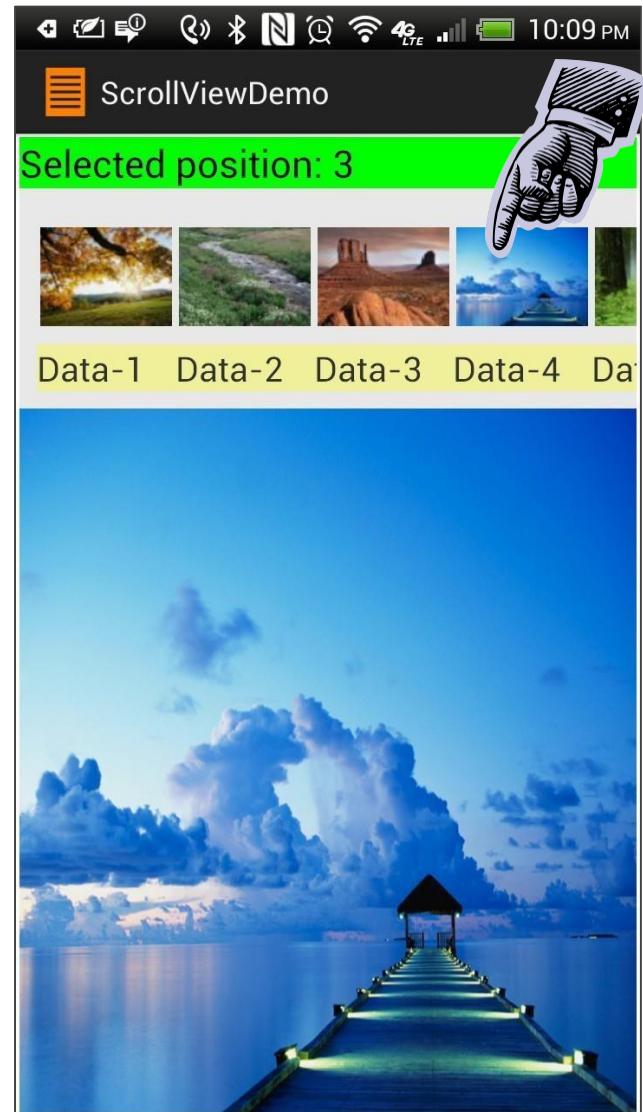
1. Przewijanie (lewo ↔ prawo)
2. Wybrać miniaturę by uaktywnić daną opcję.

W poniższym przykładzie jeżeli użytkownik kliknie w miniaturę aplikacja reaguje wyświetlając wersję obrazka w wyższej rozdzielczości.

- + Zwykle miniatura ma rozmiar 100x100 pikseli (lub mniej).

Przykład 5: Wykorzystanie HorizontalScrollView

- W tym przykładzie zostaje umieszczony **HorizontalScrollView** na górze ekranu, którego zadaniem jest zaprezentowanie zestawu miniatur.
- Użytkownik może przewijać miniatury i poprzez dotknięcie dokonać wyboru określonej.
- Wersja wybranego obrazka w wyżej rozdzielczości zostanie zaprezentowana na komponentie ImageView poniżej komponentu zawierającego listę miniatur.



Przykład 5: Wykorzystanie HorizontalScrollView

Jak stworzyć miniaturę?

- **Opcja-1.** Wykorzystać konwerter online:
<http://makeathumbnail.com>
- **Opcja-2.** Wykorzystać narzędzie Android Asset Studio
<http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html>



Przykład 5: Wykorzystanie HorizontalScrollView

Wypełnienie widżetu HorizontalScrollView

1. Widżet HorizontalScrollView wyświetli listę ramek, każda składająca się z grafiki oraz tekstu będącego jej podpisem.
2. Układ *frame_icon_caption.xml* określa formatowanie grafiki oraz jej podpisu. Układ zostanie poddanych procesowi **inflacji** by stworzyć odpowiednie elementy GUI.
3. Gdy dana *ramka* zostanie wypełniona danymi, zostanie dodana do zwiększającej się liczby widoków wewnętrz kontenera typu *scrollViewgroup* (*scrollViewgroup* jest osadzony wewnętrz komponentu odpowiedzialnego za poziomy pasek przewijania).
4. Każda *ramka* dostaje unikalne **ID** (określające pozycje wewnętrz *scrollViewgroup*) jak również unikalny nasłuchiwacz zdarzenia **onClick**.

Przykład 5: Wykorzystanie HorizontalScrollView

Układ XML: *activity_main.xml*

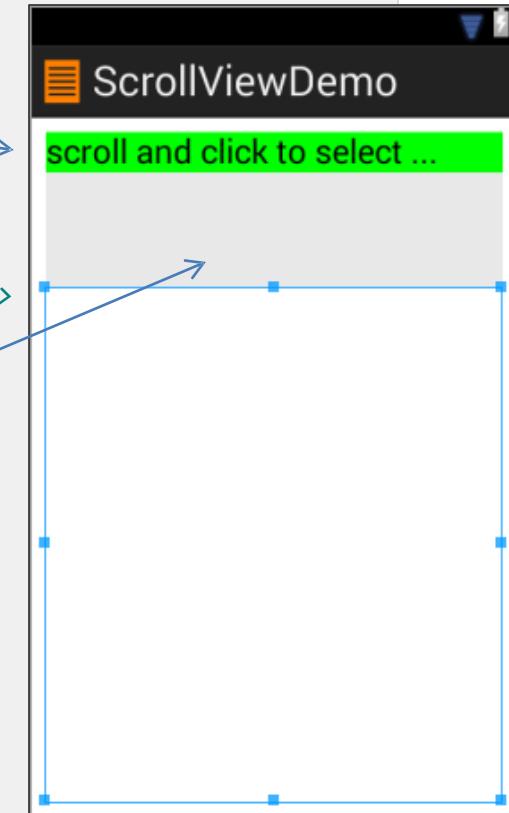
```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#ffffffff"
    android:orientation="vertical"
    android:padding="2dp" >

    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff00ff00"
        android:text="scroll and click to select ..."
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <HorizontalScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#44aaaaaa" >

        <LinearLayout
            android:id="@+id/viewgroup"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:padding="10dip" >

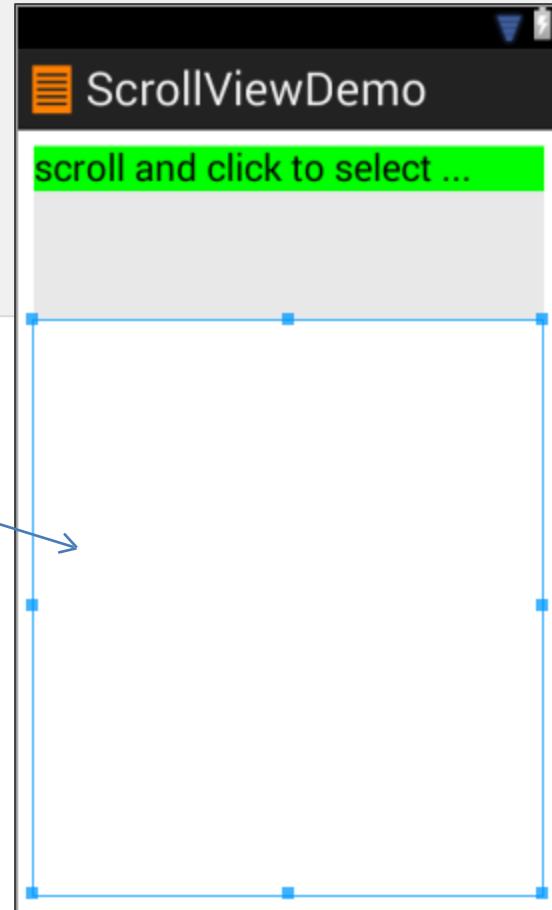
        </LinearLayout>
    </HorizontalScrollView>
```



Przykład 5: Wykorzystanie ScrollView

Układ XML : *activity_main.xml*

```
<ImageView  
    android:id="@+id/imageSelected"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_weight="2" />  
  
</LinearLayout>
```



Przykład 5: Wykorzystanie HorizontalScrollView

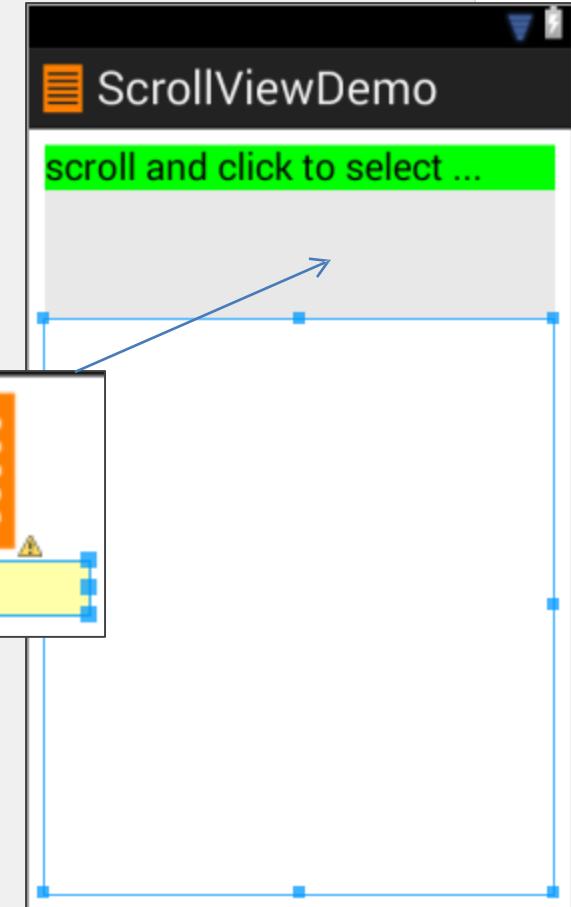
Układ XML : *frame_icon_caption.xml*

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

    <ImageView
        android:id="@+id/icon"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:paddingLeft="2dp"
        android:paddingRight="2dp"
        android:paddingTop="2dp"
        android:src="@drawable/ic_launcher" />

    <TextView
        android:id="@+id/caption"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#55ffff00"
        android:textSize="20sp" />

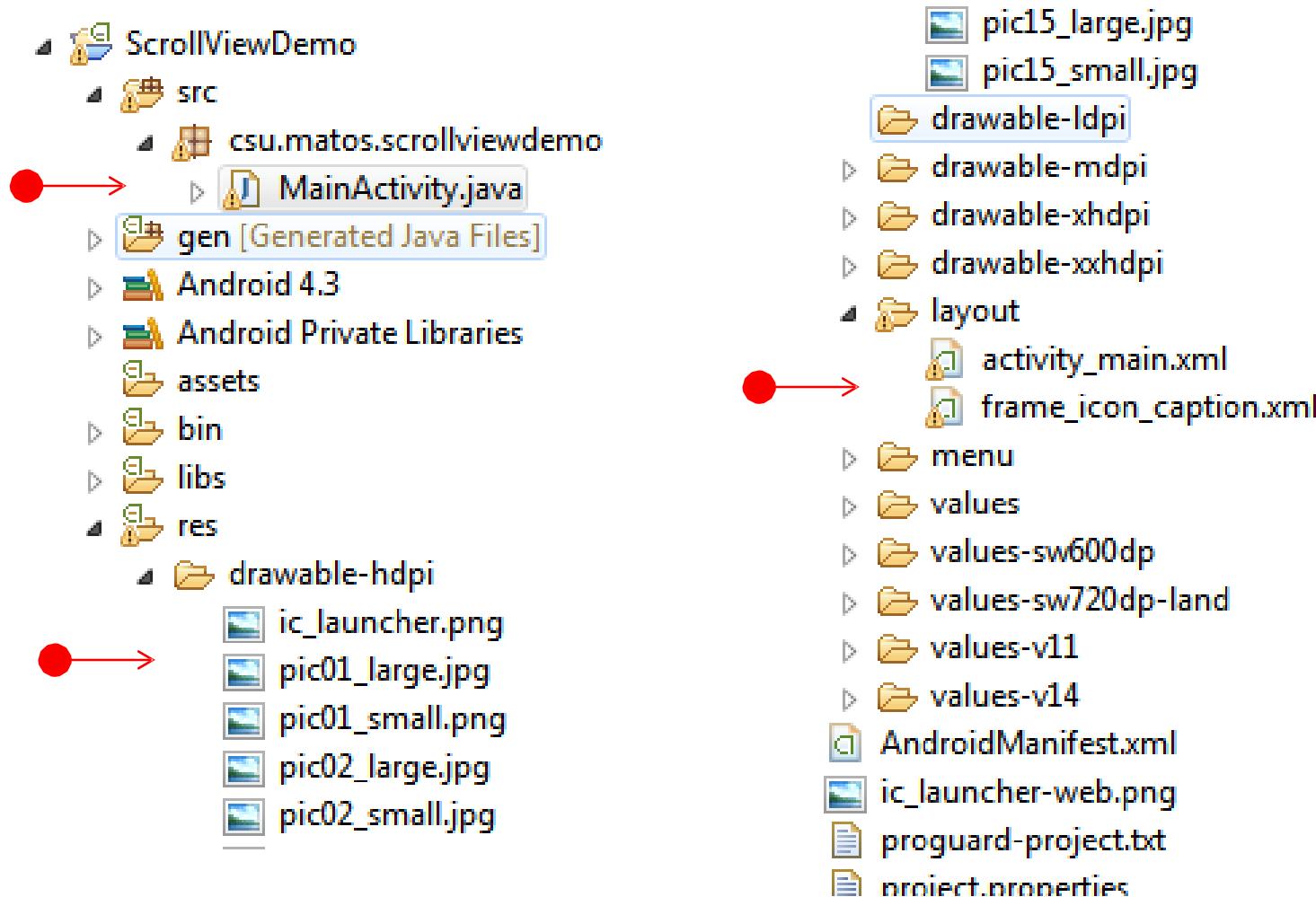
</LinearLayout>
```



This layout will be used by an **inflater** to dynamically create new views. These views will be added to the linear layout contained inside the HorizontalScrollView.

Przykład 5: Wykorzystanie ScrollView

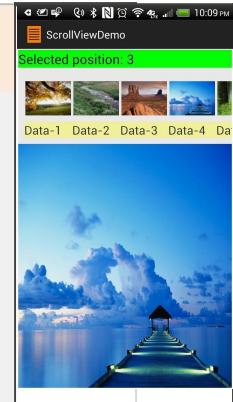
Struktura aplikacji



Przykład 5: Wykorzystanie HorizontalScrollView

Klasa MainActivity

```
public class MainActivity extends Activity {  
    //GUI controls  
    TextView txtMsg;  
    ViewGroup scrollViewgroup;  
    //each frame in the HorizontalScrollView has [icon, caption]  
    ImageView icon;  
    TextView caption;  
  
    String[] items = { "Data-1", "Data-2", "Data-3", "Data-4", "Data-5",  
                      "Data-3", "Data-7", "Data-8", "Data-9", "Data-10", "Data-11",  
                      "Data-12", "Data-13", "Data-14", "Data-15" };  
    //frame-icons ( 100X100 thumbnails )  
    Integer[] thumbnails = { R.drawable.pic01_small, R.drawable.pic02_small,  
                            R.drawable.pic03_small, R.drawable.pic04_small,  
                            R.drawable.pic05_small, R.drawable.pic06_small,  
                            R.drawable.pic07_small, R.drawable.pic08_small,  
                            R.drawable.pic09_small, R.drawable.pic10_small,  
                            R.drawable.pic11_small, R.drawable.pic12_small,  
                            R.drawable.pic13_small, R.drawable.pic14_small,  
                            R.drawable.pic15_small };  
    //large images to be shown (individually) in an ImageView  
    Integer[] largeImages = { R.drawable.pic01_large,  
                             R.drawable.pic02_large, R.drawable.pic03_large,  
                             R.drawable.pic04_large, R.drawable.pic05_large,  
                             R.drawable.pic06_large, R.drawable.pic07_large,  
                             R.drawable.pic08_large, R.drawable.pic09_large,  
                             R.drawable.pic10_large, R.drawable.pic11_large,  
                             R.drawable.pic12_large, R.drawable.pic13_large,  
                             R.drawable.pic14_large, R.drawable.pic15_large };  
}
```



Przykład 5: Wykorzystanie HorizontalScrollView

Klasa MainActivity

```
//large image frame for displaying high-quality selected image
ImageView imageSelected;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //bind GUI controls to Java classes
    txtMsg = (TextView) findViewById(R.id.txtMsg);
    imageSelected = (ImageView) findViewById(R.id.imageSelected);

    // this layout goes inside the HorizontalScrollView
    scrollViewgroup = (ViewGroup) findViewById(R.id.viewgroup);

    // populate the ScrollView
    for (int i = 0; i < items.length; i++) {
        //create single frames [icon & caption] using XML inflater
        final View singleFrame = getLayoutInflater().inflate(
                R.layout.frame_icon_caption, null );
        //frame-0, frame-1, frame-2, ... and so on
        singleFrame.setId(i);

        TextView caption = (TextView) singleFrame.findViewById(R.id.caption);
        ImageView icon = (ImageView) singleFrame.findViewById(R.id.icon);
        //put data [icon, caption] in each frame
        icon.setImageResource( thumbnails[i] );
        caption.setText( items[i] );
        //add frame to the scrollView
        scrollViewgroup.addView( singleFrame );
    }
}
```



Przykład 5: Wykorzystanie HorizontalScrollView

MainActivity Class

```
//each single frame gets its own click listener
singleFrame.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String text = "Selected position: " + singleFrame.getId();
        txtMsg.setText(text);
        showLargeImage(singleFrame.getId());
    }
}); // listener

} // for - populating ScrollView

}

//display a high-quality version of the image selected using thumbnails
protected void showLargeImage(int frameId) {
    Drawable selectedLargeImage = getResources()
        .getDrawable(largeImages[frameId]);
    imageSelected.setBackground(selectedLargeImage);

}
}
```



Siatka obrazków

Nieco ciekawszym wizualnie wykorzystaniem komponentu **GridView** jest wyświetlanie obrazków zamiast tekstu.

Poniższy przykład pokazuje jako wykorzystać w tym celu ten komponent:

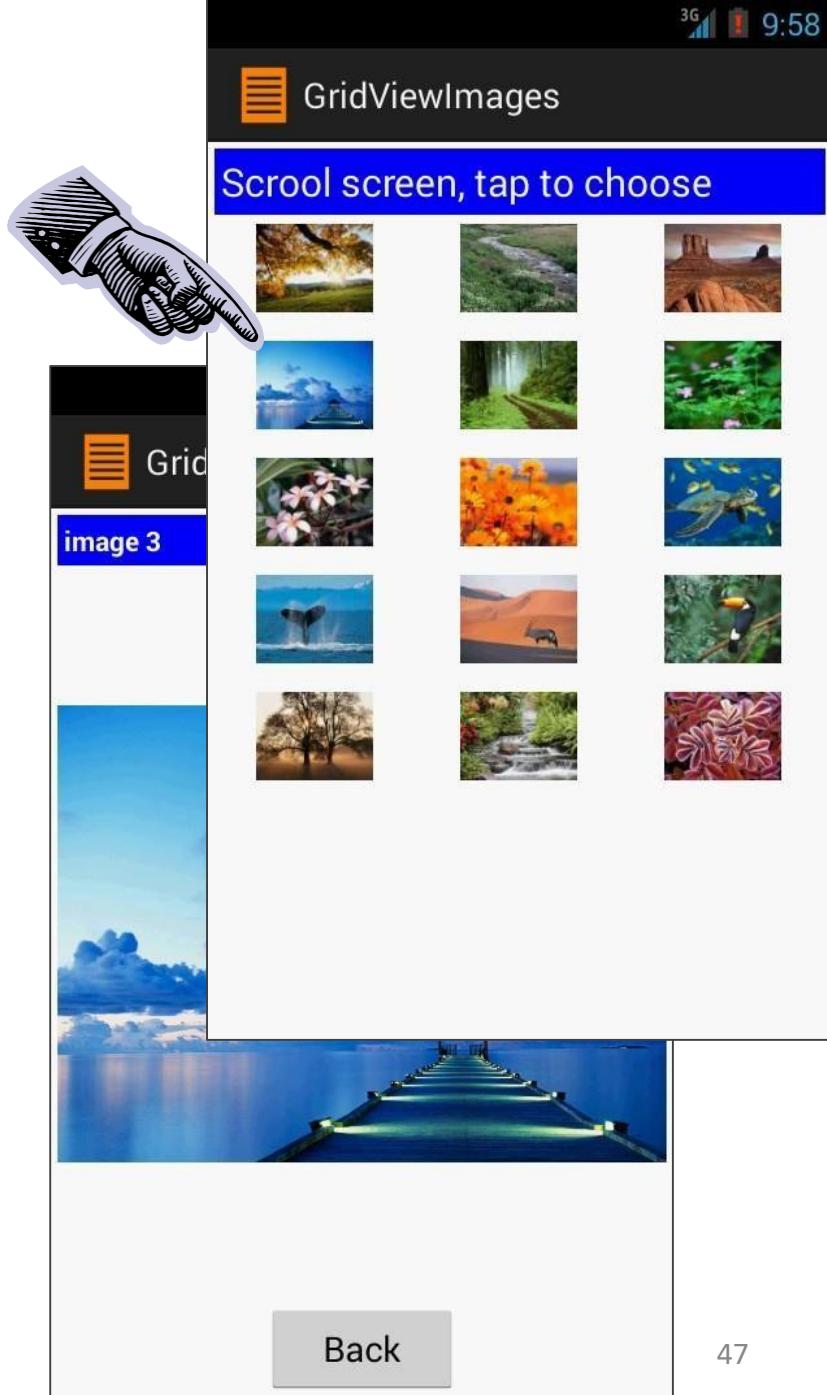
1. Ekran pokazuje tablicę miniatur.
2. Użytkownik dokonuje wyboru poprzez dotknięcie jednej z nich.
3. Aplikacja wyświetla na innym ekranie wersję obrazka o większej rozdzielczości.

Uwaga:

Programista musi stworzyć własny data adapter by określić sposób wyświetlania siatki obrazków.

Na podstawie:

<http://developer.android.com/guide/topics/ui/layout/gridview.html>



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

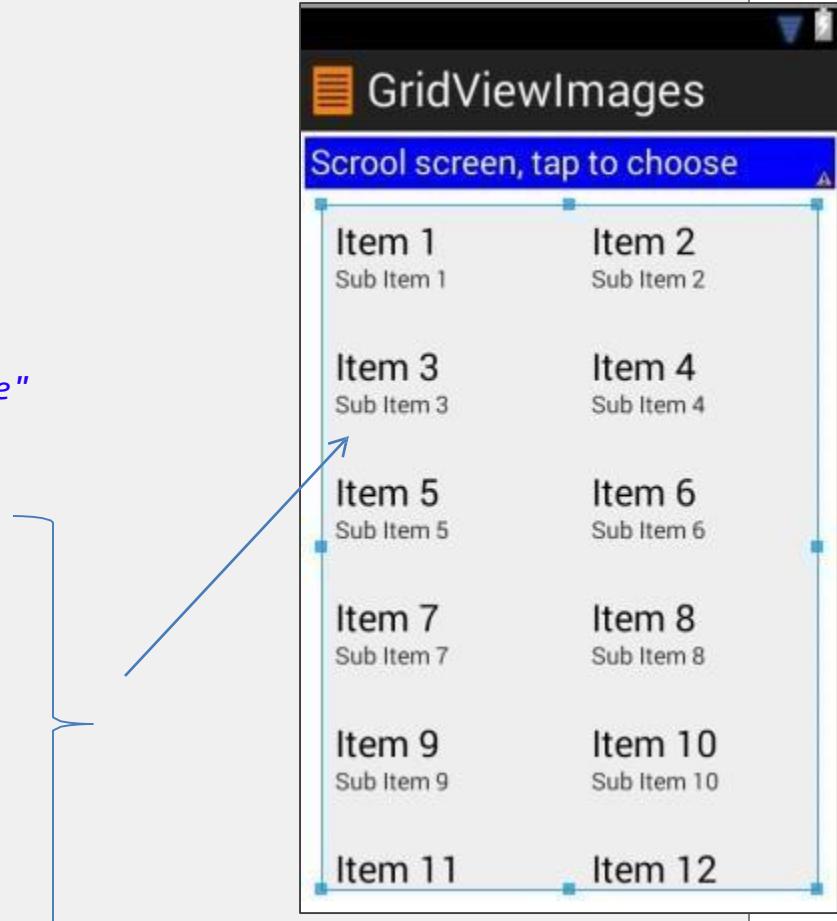
Układ XML: *activity_main.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="3dp" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:padding="3dp"
        android:text="Scrool screen, tap to choose"
        android:textColor="#ffffffff"
        android:textSize="20sp" />

    <GridView android:id="@+id/gridview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="1dp"
        android:columnWidth="100dp"
        android:gravity="center"
        android:horizontalSpacing="5dp"
        android:numColumns="auto_size"
        android:stretchMode="columnWidth"

        android:verticalSpacing="10dp" />
</LinearLayout>
```



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Układ XML: solo_picture.xml

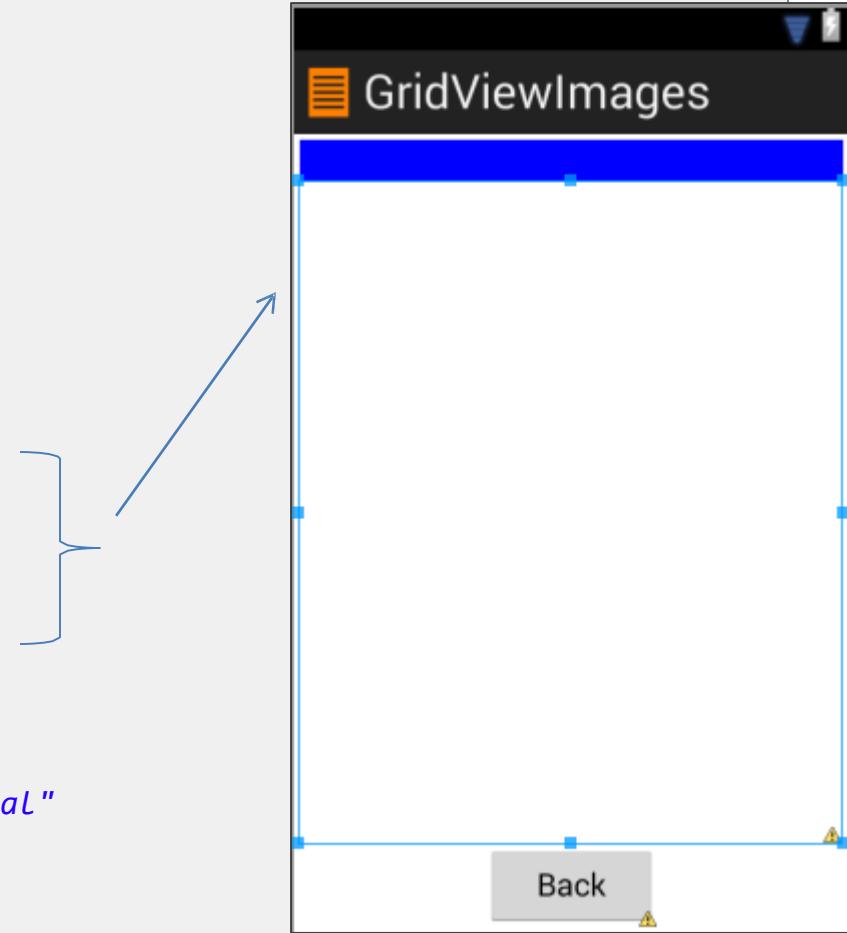
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="3dp"
    android:orientation="vertical" >

    <TextView android:id="@+id/txtSoloMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff" />

    <ImageView
        android:id="@+id/imgSoloPhoto"
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_gravity="center|fill"
        android:layout_weight="2" />

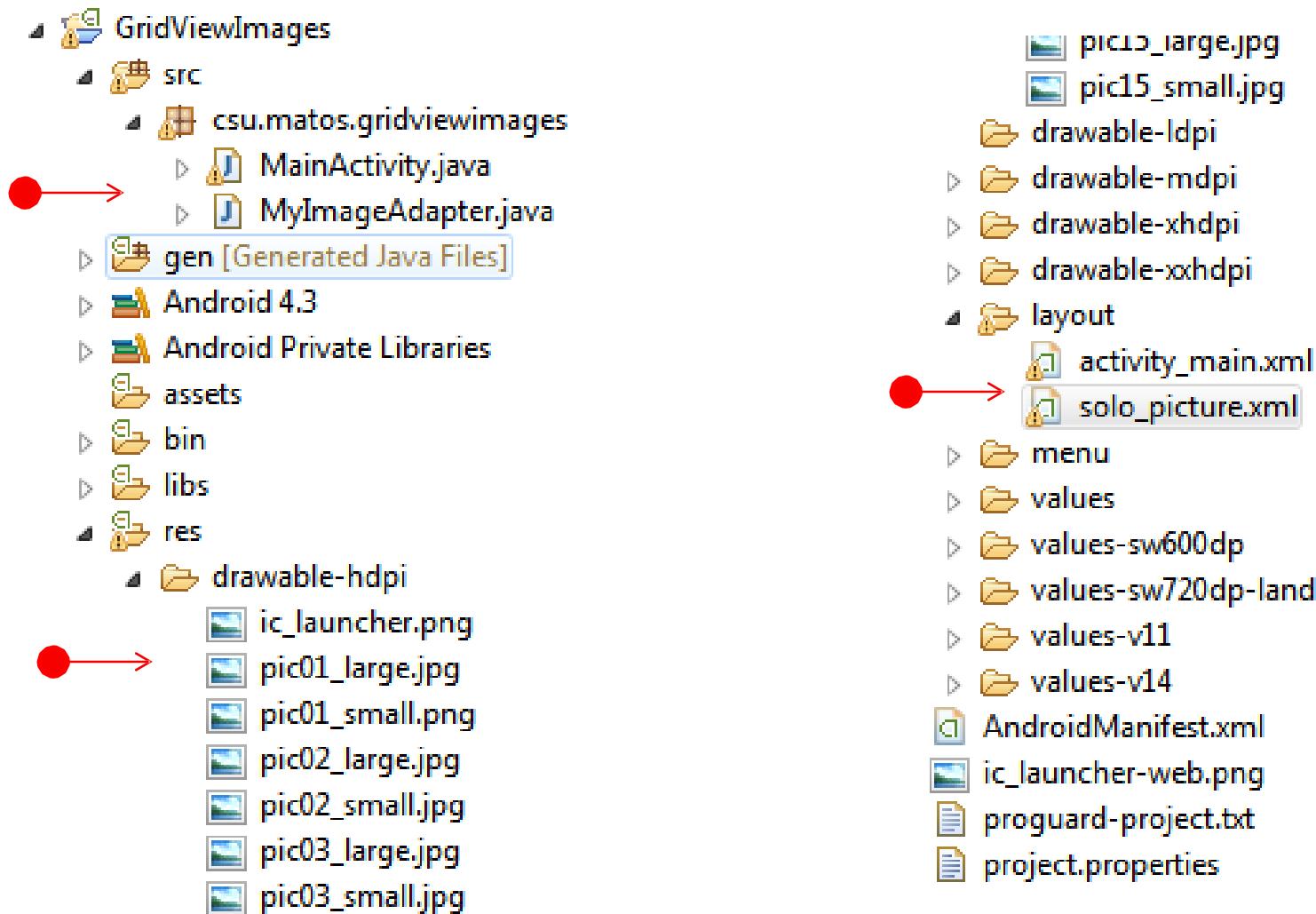
    <Button android:id="@+id	btnBack"
        android:layout_width="100dip"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Back" />

</LinearLayout>
```



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

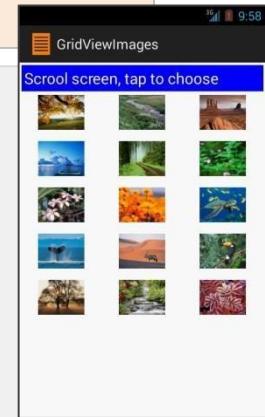
Struktura aplikacji



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Klasa MainActivity

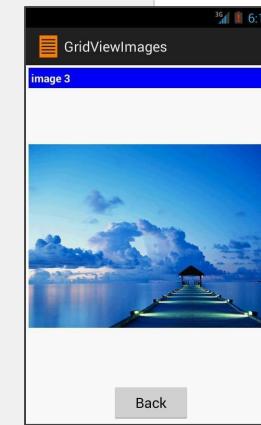
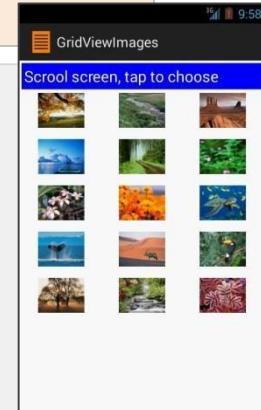
```
public class MainActivity extends Activity implements OnItemClickListener {  
    //GUI control bound to screen1 (holding GridView)  
    GridView gridview;  
    //GUI controls bound to screen2 (holding single ImageView)  
    TextView txtSoloMsg;  
    ImageView imgSoloPhoto;  
    Button btnBack;  
    // initialize array of smallImages (100x75 thumbnails)  
    Integer[] smallImages = { R.drawable.pic01_small,  
        R.drawable.pic02_small, R.drawable.pic03_small,  
        R.drawable.pic04_small, R.drawable.pic05_small,  
        R.drawable.pic06_small, R.drawable.pic07_small,  
        R.drawable.pic08_small, R.drawable.pic09_small,  
        R.drawable.pic10_small, R.drawable.pic11_small,  
        R.drawable.pic12_small, R.drawable.pic13_small,  
        R.drawable.pic14_small, R.drawable.pic15_small };  
    //initialize array of high-resolution images (1024x768)  
    Integer[] largeImages = { R.drawable.pic01_large,  
        R.drawable.pic02_large, R.drawable.pic03_large,  
        R.drawable.pic04_large, R.drawable.pic05_large,  
        R.drawable.pic06_large, R.drawable.pic07_large,  
        R.drawable.pic08_large, R.drawable.pic09_large,  
        R.drawable.pic10_large, R.drawable.pic11_large,  
        R.drawable.pic12_large, R.drawable.pic13_large,  
        R.drawable.pic14_large, R.drawable.pic15_large };  
  
    //in case you want to use-save state values  
    Bundle myOriginalMemoryBundle;
```



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Klasa MainActivity

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    myOriginalMemoryBundle = savedInstanceState;  
    // setup GridView with its custom adapter and listener  
    gridview = (GridView) findViewById(R.id.gridview);  
    gridview.setAdapter(new MyImageAdapter(this, smallImages));  
    gridview.setOnItemClickListener(this);  
  
}  
  
// on response to tapping, display a high-resolution version of the chosen thumbnail  
@Override  
public void onItemClick(AdapterView<?> parent, View v, int position, long id) {  
    showBigScreen(position);  
}//onItemClick  
  
private void showBigScreen(int position) {  
    // show the selected picture as a single frame  
    setContentView(R.layout.solo_picture);  
    txtSoloMsg = (TextView) findViewById(R.id.txtSoloMsg);  
    imgSoloPhoto = (ImageView) findViewById(R.id.imgSoloPhoto);  
    txtSoloMsg.setText("image " + position );  
  
    imgSoloPhoto.setImageResource( largeImages[position] );
```



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Klasa MainActivity

```
btnBack = (Button) findViewById(R.id.btnBack);

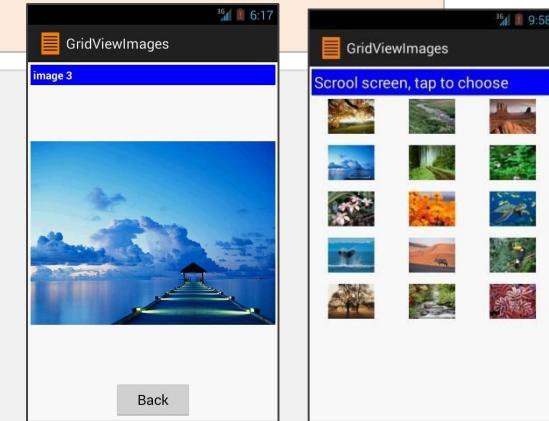
btnBack.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        // redraw the main screen showing the GridView
        onCreate(myOriginalMemoryBundle);
    }

});

} // showBigScreen

}
```



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Własny adapter: Klasa MyImageAdapter

```
// This custom adapter populates the GridView with a visual
// representation of each thumbnail in the input data set.
// It also implements a method -getView()- to access
// individual cells in the GridView.

public class MyImageAdapter extends BaseAdapter{

    private Context context; // calling activity context
    Integer[] smallImages; // thumbnail data set

    public MyImageAdapter(Context callingActivityContext,
                          Integer[] thumbnails) {
        context = callingActivityContext;
        smallImages = thumbnails;
    }

    // how many entries are there in the data set
    public int getCount() {
        return smallImages.length;
    }

    // what is in a given 'position' in the data set
    public Object getItem(int position) {
        return smallImages[position];
    }

    // what is the ID of data item in given 'position'
    public long getItemId(int position) {
        return position;
    }
}
```



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Własny adapter: Klasa MyImageAdapter

```
// create a view for each thumbnail in the data set
public View getView(int position, View convertView, ViewGroup parent) {

    ImageView imageView;

    // if possible, reuse (convertView) image already held in cache
    if (convertView == null) {
        // new image in GridView formatted to:
        // 100x75 pixels (its actual size)
        // center-cropped, and 5dp padding all around

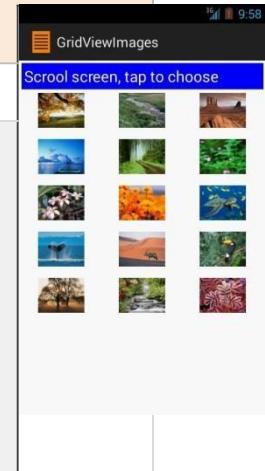
        imageView = new ImageView(context);
        imageView.setLayoutParams( new GridView.LayoutParams(100, 75) );
        imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
        imageView.setPadding(5, 5, 5, 5);

    } else {
        imageView = (ImageView) convertView;
    }

    imageView.setImageResource( smallImages[position] );

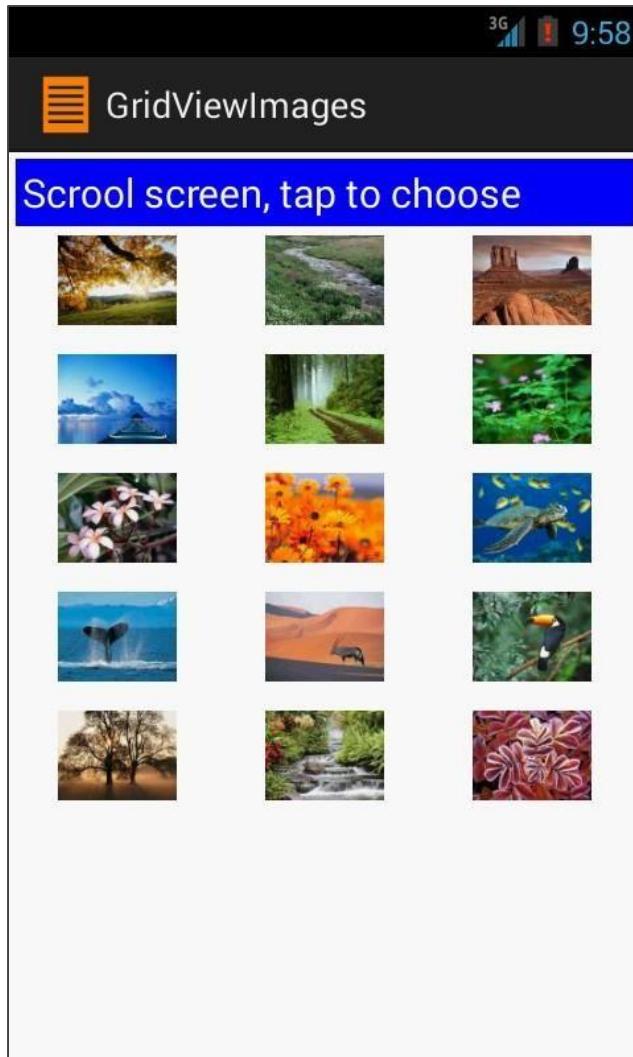
    return imageView;
}

}//MyImageAdapter
```

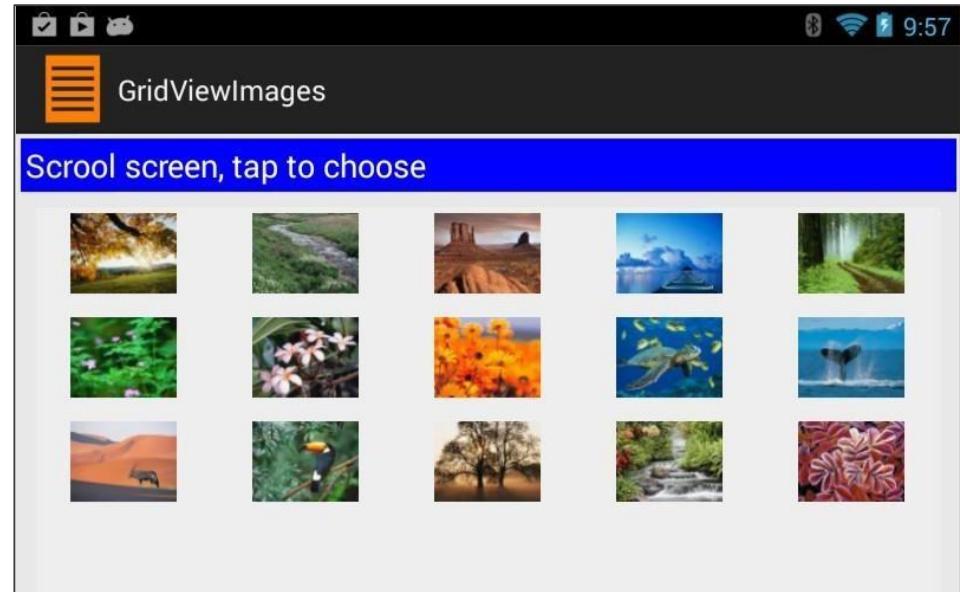


Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Wynik działania aplikacji na różnych urządzeniach



Widok na telefonie komórkowym



Widok z tabletu o rozdzielczości 1028x728
tablet. Klauzula:

`android:numColumns="auto_size"`

określa automatyczny rozkład i liczbę kolumn.

Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Eksperyment - Zmiana GridView na ListView

Celem eksperymentu jest zmiana poprzedniego przykładu by wyświetlał grafikę w **ListView** zamiast komponentu **GridView**.

Tak jak poprzednio, po wyborze danej miniatury z listy, prezentowany jest obrazek w wyższej rozdzielcości.

KROKI

1. Zmień układ **activity_main.xml**. Zmień znacznik **<GridView ...** na **<ListView**. Resztę pozostaw bez zmian.
2. W głównej klasie java zamień wszystkie referencje dotyczące **GridView** na **ListView**. Przykładowy kod po zmianach:

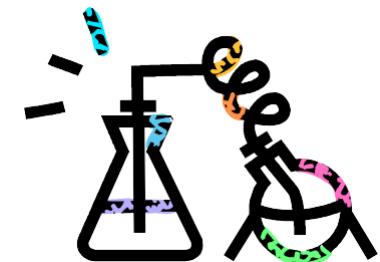
```
ListView gridview;
```

```
...
```

```
gridview = (ListView) findViewById(R.id.gridview);
```

3. We własnym adapterze dokonaj następującej zmiany by nowy obrazek (**imageView**) powinien zostać dodany do **ListView** (zamiast)

```
imageView.setLayoutParams(new ListView  
.LayoutParams(100, 75) );
```
4. Pozostały kod adaptera pozostaw bez zmian.



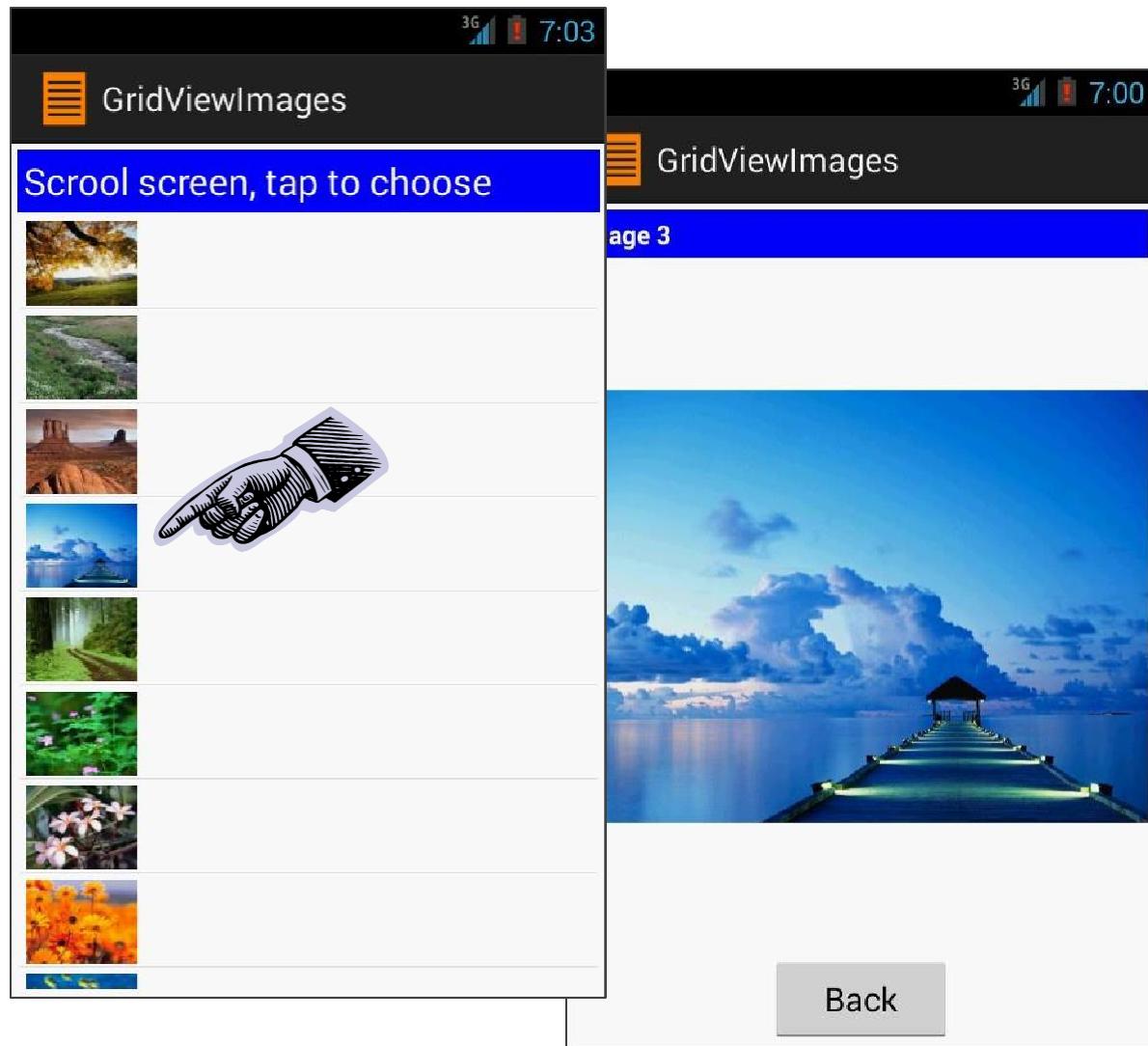
Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Eksperyment - Zmiana GridView na ListView

Nowa aplikacja powinna wyglądać następująco.

Proszę zauważyć, że główna aktywność prezentuje dane w formie listy.

Więcej w przykładzie 8.

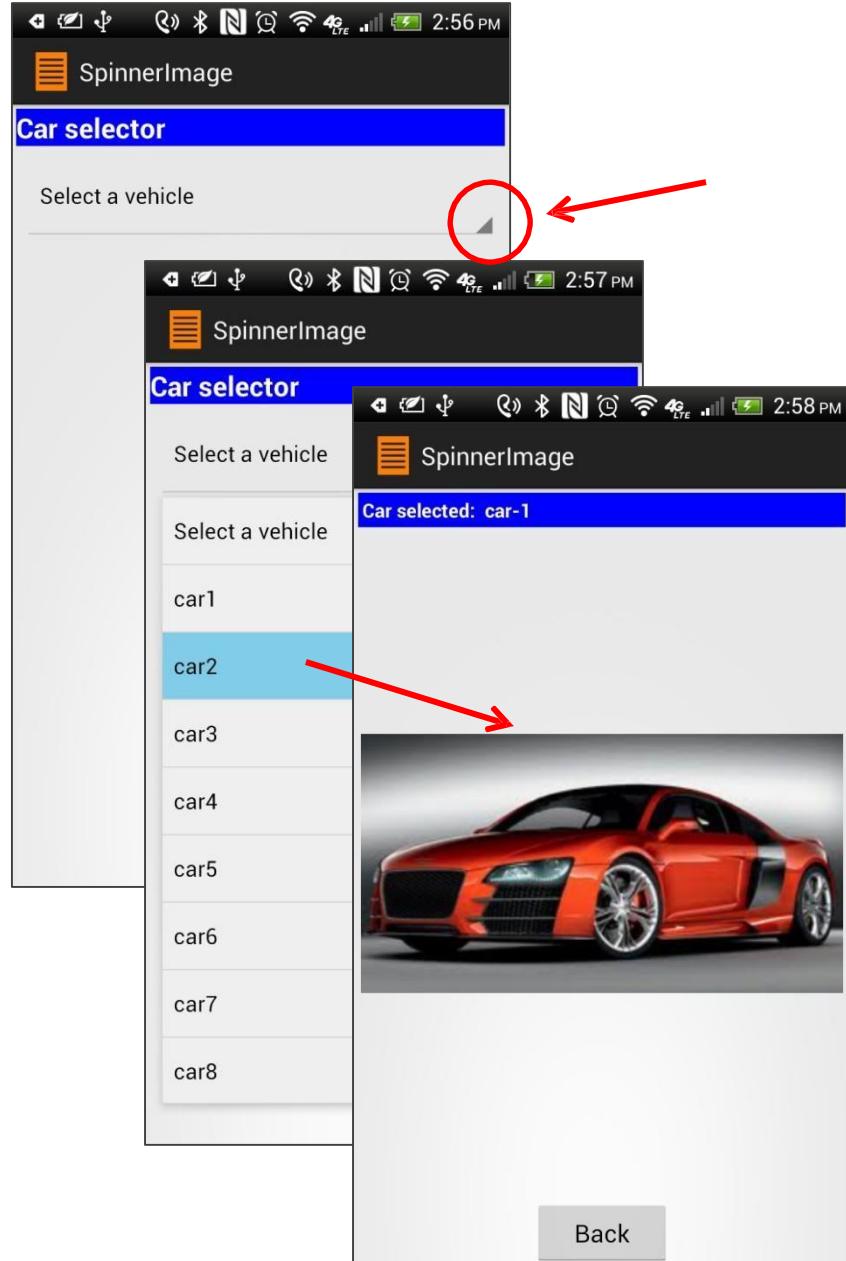


Przykład 7: Wykorzystanie Spinner'a

Jest to inna wersja poprzednio prezentowanego przykładu.

Tym razem lista możliwych wyborów prezentowana jest za pomocą komponentu typu Spinner.

Po kliknięciu przez użytkownika na określonym wierszu, obrazek dotyczący wybranej opcji prezentowany jest na ekranie.



Przykład 7: Wykorzystanie Spinner'a

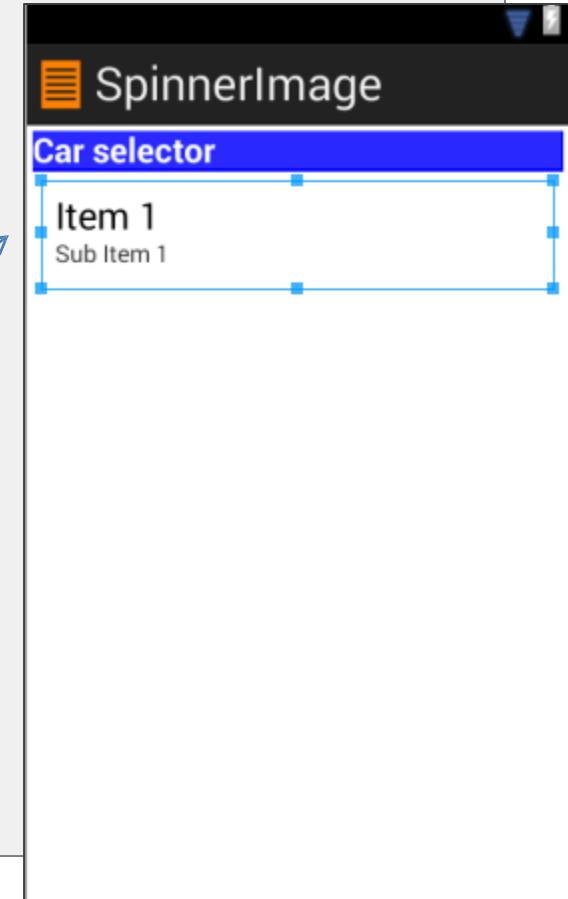
1. Układ XML: *activity_main.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="3dp" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:text="Car selector"
        android:textColor="#ffffffff"
        android:textSize="20sp"
        android:textStyle="bold" />

    <Spinner android:id="@+id/spinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dip" />

</LinearLayout>
```



Przykład 7: Wykorzystanie Spinner'a

2. Układ XML: solo_picture.xml

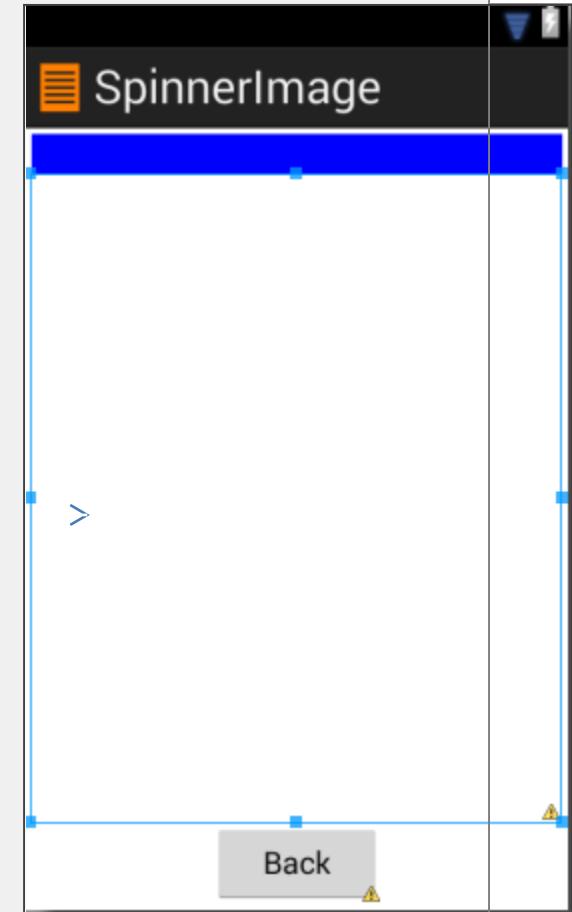
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="3dp"
    android:orientation="vertical" >

    <TextView android:id="@+id/txtSoloMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:textColor="#FFFFFF"
        android:padding="3dp"
        android:background="#ff0000ff" />

    <ImageView
        android:id="@+id/imgSoloPhoto"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_gravity="center|fill"
        android:layout_weight="2" />

    <Button android:id="@+id	btnBack"
        android:layout_width="100dip"
        android:layout_height="wrap_content"

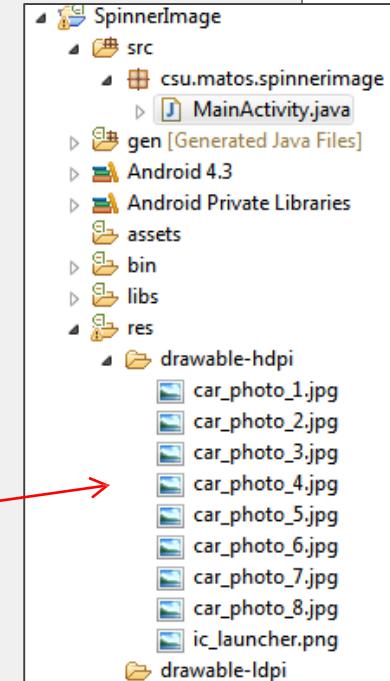
        android:layout_gravity="center_horizontal"
        android:text="Back" />
</LinearLayout>
```



Przykład 7: Wykorzystanie Spinner'a

Klasa MainActivity

```
public class MainActivity extends Activity implements AdapterView.OnItemSelectedListener {  
  
    // GUI controls in the main screen  
    Spinner spinner;  
  
    // GUI controls in the solo_picture screen  
    TextView txtSoloMsg;  
    ImageView imageSelectedCar;  
    Button btnBack;  
  
    // captions to be listed by the spinner  
    String[] items = { "Select a vehicle", "car1", "car2", "car3", "car4",  
                      "car5", "car6", "car7", "car8" };  
    // object IDs of car pictures  
    Integer[] carImageArray = new Integer[] { R.drawable.car_photo_1,  
                                              R.drawable.car_photo_2, R.drawable.car_photo_3,  
                                              R.drawable.car_photo_4, R.drawable.car_photo_5,  
                                              R.drawable.car_photo_6, R.drawable.car_photo_7,  
                                              R.drawable.car_photo_8, };  
    Bundle myStateInfo;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        myStateInfo = savedInstanceState;  
  
        setContentView(R.layout.activity_main);  
    }  
}
```



Przykład 7: Wykorzystanie Spinner'a

Klasa MainActivity

```
spinner = (Spinner) findViewById(R.id.spinner);
spinner.setAdapter(new ArrayAdapter<String>(this,
                                         android.R.layout.simple_spinner_dropdown_item,
                                         items));
spinner.setOnItemSelectedListener(this);

}// onCreate

// display screen showing image of the selected car
private void showBigImage(int position) {
    // show the selected picture as a single frame
    setContentView(R.layout.solo_picture);
    txtSoloMsg = (TextView) findViewById(R.id.txtSoloMsg);
    imageSelectedCar = (ImageView) findViewById(R.id.imgSoloPhoto);
    txtSoloMsg.setText("Car selected: car-" + position);

    imageSelectedCar.setImageResource(carImageArray[position]);

    btnBack = (Button) findViewById(R.id.btnBack);
    btnBack.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            // redraw the main screen showing the spinner
            onCreate(myStateInfo);
        }
    });
}

}// showBigScreen
```

Przykład 7: Wykorzystanie Spinner'a

Klasa MainActivity

```
// next two methods implement the spinner listener
@Override
public void onItemSelected(AdapterView<?> parent, View v, int position,
    long id) {
    //ignore position 0. It holds just a label ("SELECT A VEHICLE...")
    if (position > 0) {
        showBigImage(position - 1);
    }
}

@Override
public void onNothingSelected(AdapterView<?> parent) {
    // DO NOTHING - needed by the interface
}

}
```

Własne widżety wykorzystujące listy

Niestandardowe listy

- Android dostarcza wiele układów do prezentowania danych na listach (takie jak: `android.R.layout.simple_list_item_1`, `android.R.layout.simple_list_item_2`, itp).
- Jednakże, w niektórych przypadkach istnieje potrzeba zdecydowanie większej kontroli nad formatowaniem i wyglądem poszczególnych elementów.
- W takich przypadkach należy ***stworzyć własną podklasę rozszerzającą Data Adapter.***
- Zostanie to pokazane na następnym przykładzie.

Przykład 8: Własne listy

Tworzenie własnego Data Adapter

By stworzyć własny Data Adapter należy:

1. Stworzyć klasę dziedziczącą po klasie `ArrayAdapter`
2. Przeciążyć metodę `getView()`
3. Stworzyć (przez inflację) poszczególne wiersze samemu.

```
public class MyCustomAdapter extends ArrayAdapter{  
  
    // class variables go here ...  
  
    public MyCustomAdapter(...) {    }  
  
    public View getView(...) {      }  
  
}//MyCustomAdapter
```

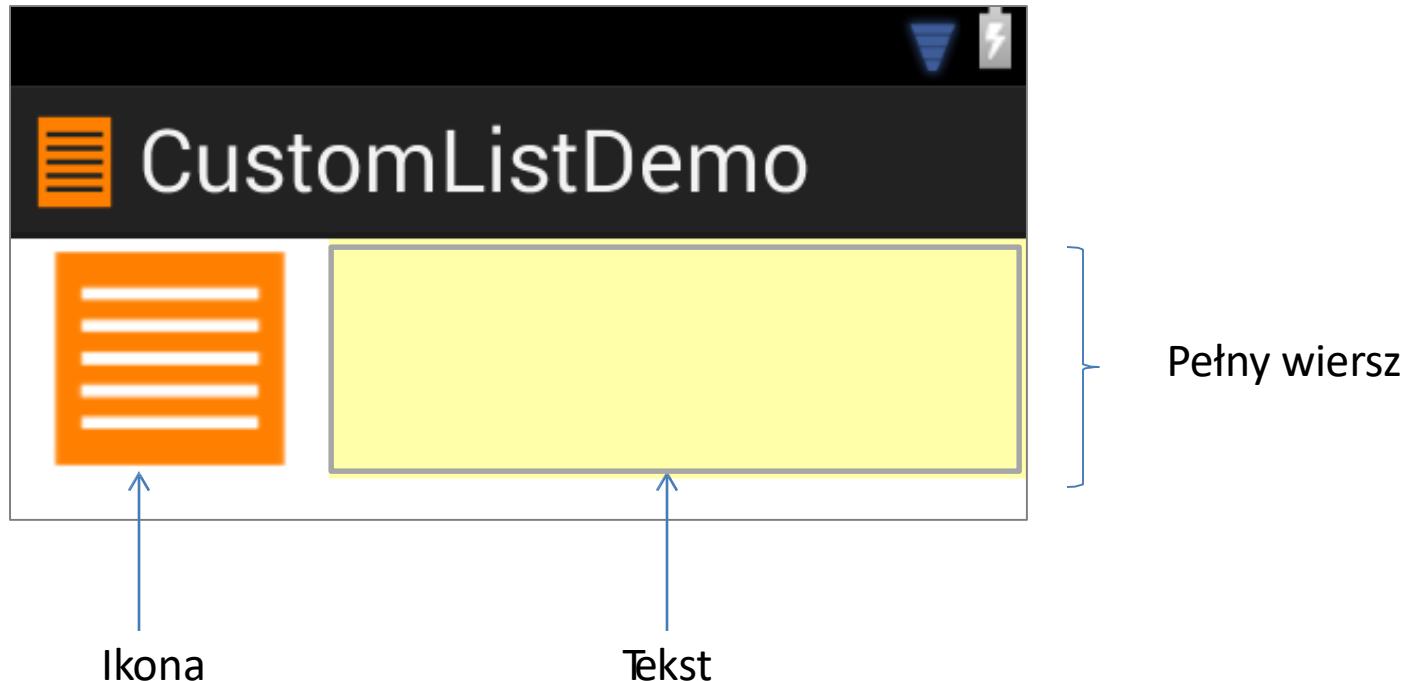
Dla każdego elementu dostarczanego przez adapter metoda `getView()` zwraca jego wizualną reprezentację.



Przykład 8: Własne listy

Problem: Stworzenie własnego wyglądu dla wierszy

Każdy wiersz będzie pokazywał ikonę (po lewej) oraz tekst (po prawej).



Przykład 8: Własne listy

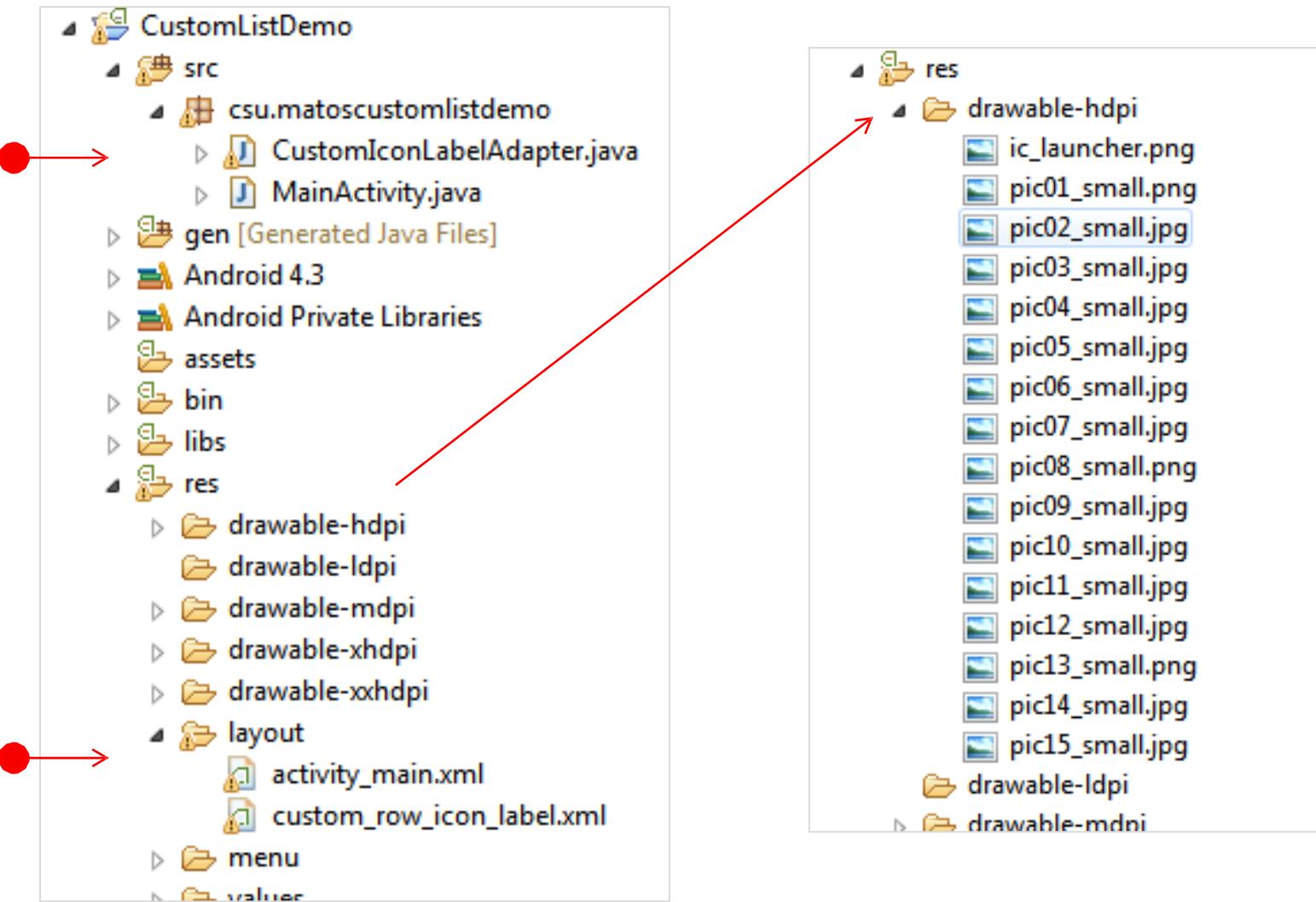


Wygląd aplikacji

Każdy wiersz składa się z ikony i tekstu

Przykład 8: Własne listy

1. Struktura aplikacji



Przykład 8: Własne listy

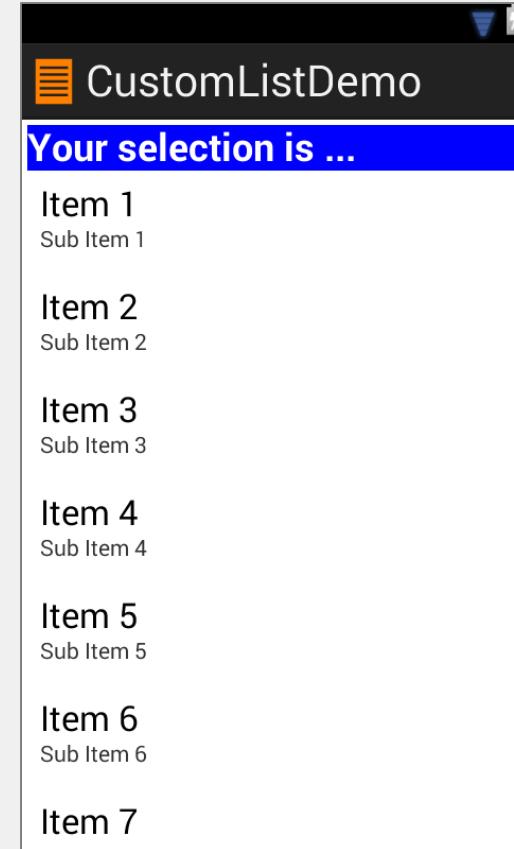
1. Układ XML: *activity_main.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="3dp"
    android:orientation="vertical" >

    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:text="Your selection is ..."
        android:textColor="#ffffffff"
        android:textSize="24sp"
        android:textStyle="bold" />

    <ListView android:id="@+id>List"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </ListView>

</LinearLayout>
```



Przykład 8: Własne listy

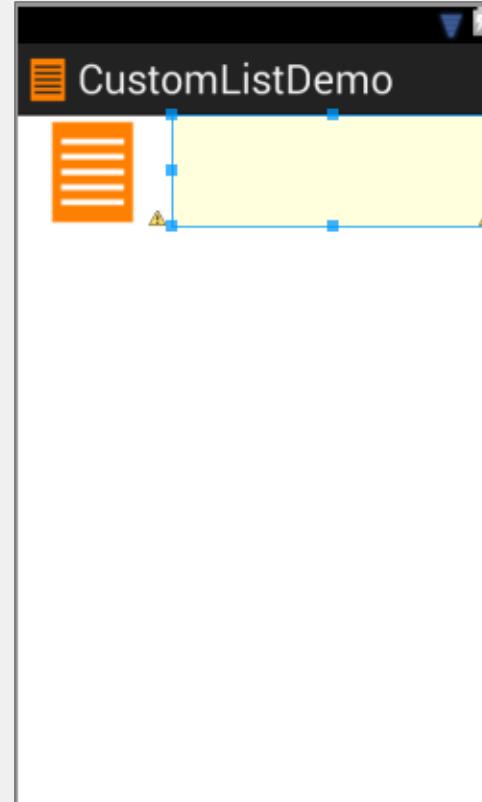
2. Układ XML: *custom_row_icon_label.xml*

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

    <ImageView android:id="@+id/icon"
        android:layout_width="100dp"
        android:layout_height="75dp"
        android:layout_marginRight="3dp"
        android:src="@drawable/ic_launcher" />

    <TextView android:id="@+id/label"
        android:layout_width="match_parent"
        android:layout_height="75dp"
        android:background="#22ffff00"
        android:textSize="20sp" />

</LinearLayout>
```



Przykład 8: Własne listy

2. Klasa MainActivity

```
public class MainActivity extends ListActivity {  
    TextView txtMsg;  
    // The n-th row in the list will consist of [icon, label]  
    // where icon = thumbnail[n] and label=items[n]  
    String[] items = { "Data-1", "Data-2", "Data-3", "Data-4", "Data-  
        5", "Data-6", "Data-7", "Data-8", "Data-9", "Data-10", "Data-  
        11", "Data-12", "Data-13", "Data-14", "Data-15" };  
  
    Integer[] thumbnails = { R.drawable.pic01_small,  
        R.drawable.pic02_small, R.drawable.pic03_small,  
        R.drawable.pic04_small, R.drawable.pic05_small,  
        R.drawable.pic06_small, R.drawable.pic07_small,  
        R.drawable.pic08_small, R.drawable.pic09_small,  
        R.drawable.pic10_small, R.drawable.pic11_small,  
        R.drawable.pic12_small, R.drawable.pic13_small,  
        R.drawable.pic14_small, R.drawable.pic15_small };  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState)  
    { super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        txtMsg = (TextView) findViewById(R.id.txtMsg);  
    }  
}
```

Przykład 8: Własne listy

2. Klasa MainActivity

```
// the arguments of the custom adapter are:  
// activityContex, layout-to-be-inflated, labels, icons  
CustomIconLabelAdapter adapter = new CustomIconLabelAdapter(  
    this,  
    R.layout.custom_row_icon_label,  
    items,  
    thumbnails);  
// bind intrinsic ListView to custom  
adapter setListAdapter(adapter);  
  
}//onCreate  
  
// react to user's selection of a row  
@Override  
protected void onListItemClick(ListView l, View v, int position, long id) {  
    super.onListItemClick(l, v, position, id);  
    String text = " Position: " + position + " " + items[position];  
    txtMsg.setText(text);  
}//listener  
  
}//class
```

Przykład 8: Własne listy

3. Klasa CustomIconLabelAdapter

```
class CustomIconLabelAdapter extends ArrayAdapter <String>
{ Context context;
Integer[] thumbnails;
String[] items;

public CustomIconLabelAdapter( Context context, int
                                layoutToBeInflated, String[] items,
                                Integer[] thumbnails) {
    super(context, R.layout.custom_row_icon_label, items);
    this.context = context;
    this.thumbnails = thumbnails;
    this.items = items;
}
@Override
public View getView(int position, View convertView, ViewGroup parent)
{
    LayoutInflator inflater = ((Activity)
        context).getLayoutInflator(); View row =
    inflater.inflate(R.layout.custom_row_icon_label, null);
    TextView label = (TextView) row.findViewById(R.id.Label);
    ImageView icon = (ImageView)
    row.findViewById(R.id.icon);

    label.setText(items[position]);
    icon.setImageResource(thumbnails[position]);
    return row;
}
```

Przykład 8: Własne listy

LayoutInflater()

- Klasa **LayoutInflater** konwertuje układ XML w rzeczywista hierarchię obiektów klasy View. Obiekty poddane inflacji dołączane są do aktualnego widoku. Omawiana klasa działa zwykle w porozumieniu z ArrayAdapter.
- Prosty **ArrayAdapter** wymaga podania 3 argumentów: aktualnego **kontekstu**, **układu** wedle którego wiersze są formatowane, danych **źródłowych**.
 - Przeciążona metoda `getView()` tworzy niestandardowy układ wierszy układając grafikę i test z danych źródłowych na podstawie podanej specyfikacji układu XML.
 - Po stworzeniu, widok (wiersz) jest prezentowany (dodawany do listy).
 - Ten proces jest powtarzany dla każdego elementu dostarczonego przez ArrayAdapter.

Przykład 9: Przechowywanie grafik na karcie SD

Wariant przykładu 5.

W poprzednich przykładach, obrazki były przechowywane w pamięci aplikacji (przestrzeni jej zasobów).

Tym razem wykorzystywany będzie zewnętrzny nośnik (**karta SD**).

Zakłada się, że zdjęcia znajdują się w katalogu karty pamięci **/Pictures/thumbnails/**



Name	Size
proc	
root	
sbin	
sdcard	
storage	
sdcard0	
Alarms	
Android	
Audible	
Audiobooks	
Autodesk	
DCIM	
Download	
Movies	
Music	
New Folder	
Notifications	
Pictures	
cache	
large_images	
thumbnails	
pic01_small.png	3477
pic02_small.jpg	3078
pic03_small.jpg	2701
pic04_small.jpg	2219
pic05_small.jpg	2836
pic06_small.jpg	3071
pic07_small.jpg	3293
pic08_small.png	3067
pic09_small.jpg	2871
pic10_small.jpg	2186
pic11_small.ipq	1894

Przykład 9: Przechowywanie grafik na karcie SD

Klasa MainActivity

```
public class MainActivity extends Activity {  
  
    ViewGroup scrollViewgroup;  
    ImageView icon;  
    TextView caption;  
    TextView txtMsg;  
    int index;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        txtMsg = (TextView) findViewById(R.id.txtMsg);  
  
        // this layout is contained inside the horizontalScrollView  
        scrollViewgroup = (ViewGroup) findViewById(R.id.scrollViewgroup);  
  
        try {  
            // Environment class allows access to your 'MEDIA' variables  
            String absolutePath2SdCard = Environment.getExternalStorageDirectory()  
                .getAbsolutePath();  
  
            String path2PicturesOnSdCard = absolutePath2SdCard + "/Pictures/thumbnails/";  
            // .listFiles() returns an array of files contained in the directory  
            // represented by this file-path (or NULL if not a directory).  
            File sdPictureFiles = new File(path2PicturesOnSdCard);  
            File[] files = sdPictureFiles.listFiles();  
        }  
    }  
}
```

In this app we use the same layouts earlier introduced in Example 5

Przykład 9: Przechowywanie grafik na karcie SD

Klasa MainActivity

```
txtMsg.append("\nNum files: " + files.length);

File file;

for (index = 0; index < files.length; index++) {
    file = files[index];

    final View frame = getLayoutInflater().inflate(
        R.layout.frame_icon_label, null);

    TextView caption = (TextView) frame.findViewById(R.id.caption);
    ImageView icon = (ImageView) frame.findViewById(R.id.icon);

    // convert (jpg, png,...) file into an acceptable bitmap
    BitmapFactory.Options bmOptions = new BitmapFactory.Options();

    bmOptions.inSampleSize = 1; // one-to-one scale

    Bitmap bitmap = BitmapFactory.decodeFile( file.getAbsolutePath(),
        bmOptions);

    icon.setImageBitmap(bitmap);

    caption.setText("File-" + index);
```



Przykład 9: Przechowywanie grafik na karcie SD

Klasa MainActivity

```
scrollViewgroup.addView(frame);

frame.setId(index);

frame.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String text = "Selected position: " + frame.getId();
        txtMsg.setText(text);
    }
}); // listener

}// for

} catch (Exception e) {

    txtMsg.append("\nError: " + e.getMessage());
}

} // onCreate
} // class
```



Przykład 9: Przechowywanie grafik na karcie SD

Wykorzystanie klasy BitmapFactory

BitmapFactory

Tworzy obiekty bitmapy z różnych źródeł, wliczając pliki, strumieni, czy tablic bajtów.

```
Bitmap bitmap= BitmapFactory.decodeFile(  
    file.getAbsolutePath(),  
    Options);
```

Skalowanie wykorzystując właściwość **inSampleSize**:

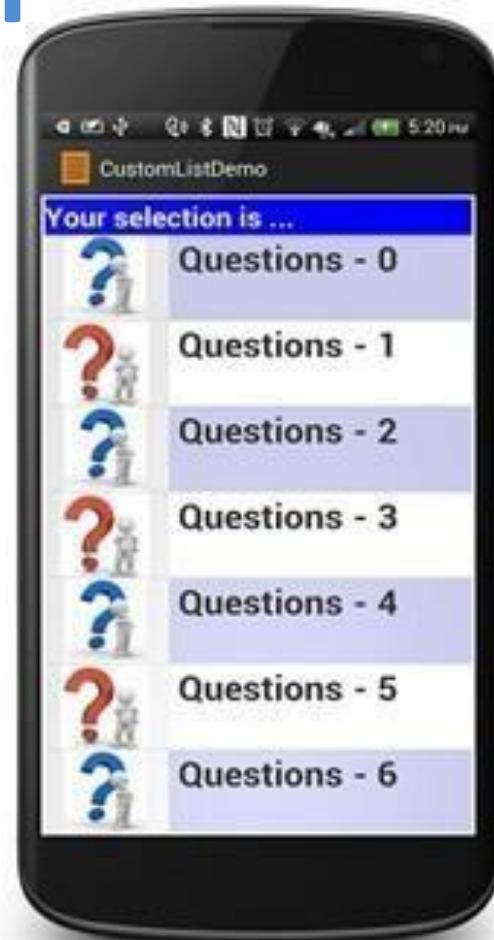
- Dla wartości **1**, zwraca oryginalny obrazek.
- Jeżeli ustawiono wartość **> 1**, dekoder zwraca mniejszy obrazek.
- Przykładowo, dla **inSampleSize == 4** zostanie zwrócony obrazek o wielkości **1/4** oryginału (co implikuje 1/16 liczbę oryginalnych pikseli).
- Wartość **< 1** jest traktowana jak w przypadku pierwszym.

Źródło:

<http://developer.android.com/reference/android/graphics/BitmapFactory.Options.html>

• <code>inBitmap : Bitmap - BitmapFactory.Options</code>
• <code>inDensity : int - BitmapFactory.Options</code>
• <code>inDither : boolean - BitmapFactory.Options</code>
• <code>inInputShareable : boolean - BitmapFactory.Options</code>
• <code>inJustDecodeBounds : boolean - BitmapFactory.Options</code>
• <code>inMutable : boolean - BitmapFactory.Options</code>
• <code>inPreferQualityOverSpeed : boolean - BitmapFactory.Options</code>
• <code>inPreferredConfig : Bitmap.Config - BitmapFactory.Options</code>
• <code>inPurgeable : boolean - BitmapFactory.Options</code>
• <code>inSampleSize : int - BitmapFactory.Options</code>
• <code>inScaled : boolean - BitmapFactory.Options</code>
• <code>inScreenDensity : int - BitmapFactory.Options</code>
• <code>inTargetDensity : int - BitmapFactory.Options</code>
• <code>inTempStorage : byte[] - BitmapFactory.Options</code>
• <code>mCancel : boolean - BitmapFactory.Options</code>
• <code>outHeight : int - BitmapFactory.Options</code>
• <code>outMimeType : String - BitmapFactory.Options</code>
• <code>outWidth : int - BitmapFactory.Options</code>

Wyświetlanie danych na listach



Grafiki zrealizowano za pomocą narzędzia:
<http://android-ui-utils.googlecode.com/hg/asset-studio/dist/device-frames.html>

Dodatek A: Gotowe zasoby

SDK Androida zawiera wiele uprzednio zdefiniowanych szablonów/stylów.

Niektóre z nich mogą zostać znalezione w:

C:\ Your-Path \Android\android-sdk\platforms\android-xx\data\res\layout

C:\ Your-Path \Android\android-sdk\platforms\android-xx\data\res\values\styles.xml

Przykład:

Poniżej znajduje się definicja szablonu o nazwie:

android.R.layout.simple_list_item_1. Składa się z pojedynczego pola tekstowego TextView nazwanego “text1”, jego zawartość jest wycentrowana, użyto dużej czcionki i dodano margines.

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/text1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:minHeight="?android:attr/listPreferredItemHeight"
    android:paddingLeft="6dip"
    android:textAppearance="?android:attr/textAppearanceLarge"
/>
```

Dodatek A: Gotowe zasoby

Globalne, zdefiniowane szablony

Poniżej znajduje się definicja szablonu: ***simple_spinner_dropdown_item*** w którym każdy wiersz zawiera pole wyboru i tekst.

Źródło: http://code.google.com/p/pdn-slatedroid/source/browse/trunk/eclair/frameworks/base/core/res/res/layout/simple_spinner_dropdown_item.xml?r=44

```
<?xml version="1.0" encoding="utf-8"?>
<!--
** Copyright 2008, The Android Open Source Project
** etc...
-->
<CheckedTextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/text1"
    style="?android:attr/spinnerDropDownItemStyle"
    android:singleLine="true"
    android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:ellipsize="marquee" />
```

Dodatek B: Pola tekstowe i klawiatury

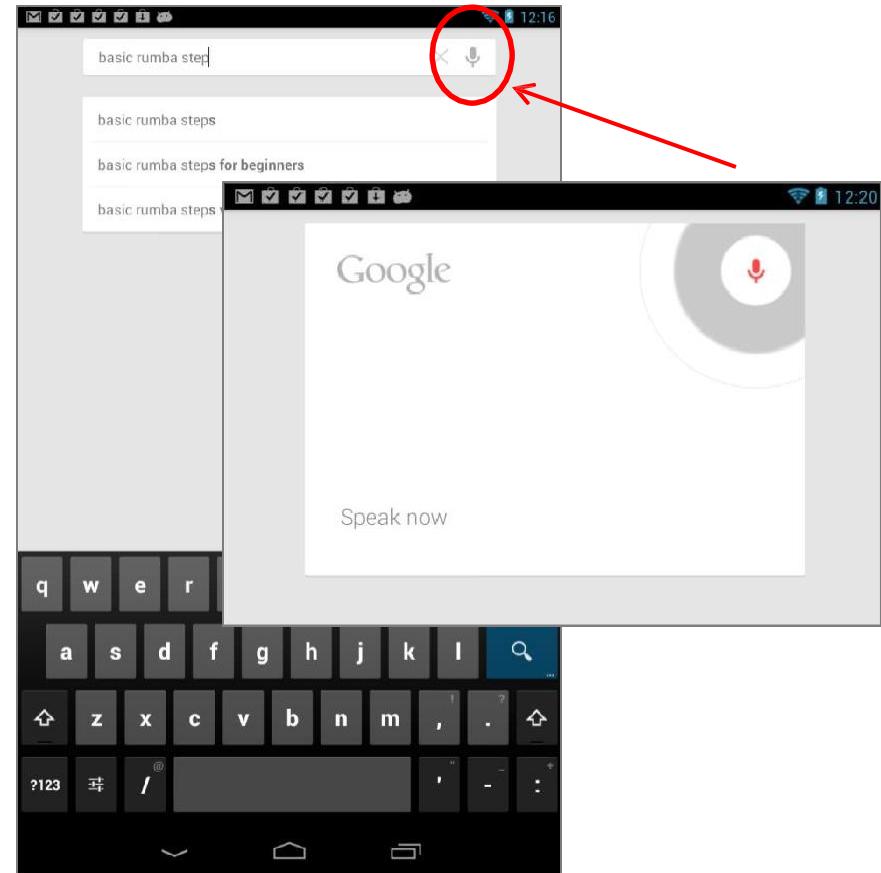
Sposób wprowadzania danych różni się w zależności od fizycznych możliwości danego urządzenia.



Wbudowana klawiatura fizyczna, dostępna po otwarciu pokrywy



Urządzenie na fizyczną klawiaturę dostępną stale.



Za dane wejściowe odpowiada wirtualna klawiatura lub rozpoznanie mowy.

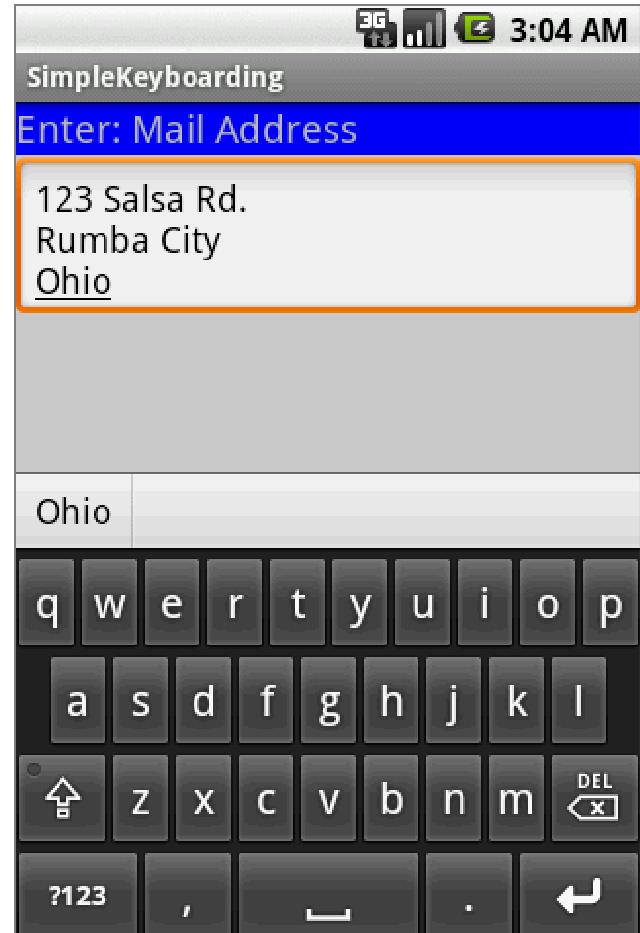
Dodatek B: Pola tekstowe i klawiatury

Gdy użytkownik kliknie w pole **EditText**, Input Media Framework (**IMF**) zapewnia dostęp do:

1. Fizycznej klawiatury (jeśli dostępna) lub
2. Wirtualnej klawiatury (znanej jako IME) najczęściej spotykanej.

Zamknięcie klawiatury wymaga wcisnięcia klawisza **Wstecz**.

IME Soft
Keyboard



IME: Input Media Editor

Dodatek B: Pola tekstowe i klawiatury

Informacja jakich danych należy się spodziewać

Komponenty TextView mogą używać elementów **XML** lub kodu **Java** by określić jakiego typu dane pole tekstowe powinno akceptować. Przykładowo:

XML

```
android:inputType="phone"
```

Java

```
editTextBox.setInputType(  
    android.text.InputType.TYPE_CLASS_PHONE);
```

Pole `inputType` ma wpływ na klawiatury wirtualne (oprogramowanie może wówczas wyświetlić najlepszą wersję klawiatury do wprowadzenia danego ciągu znaków)

Dodatek B: Pola tekstowe i klawiatury

Dostępne warianty inputType

`editTextBox.setInputType(android.text.InputType.XXX);`

```
SF TYPE_CLASS_DATETIME : int - InputType
SF TYPE_CLASS_NUMBER : int - InputType
SF TYPE_CLASS_PHONE : int - InputType
SF TYPE_CLASS_TEXT : int - InputType
SF TYPE_DATETIME_VARIATION_DATE : int - InputType
SF TYPE_DATETIME_VARIATION_NORMAL : int - InputType
SF TYPE_DATETIME_VARIATION_TIME : int - InputType
SF TYPE_MASK_CLASS : int - InputType
SF TYPE_MASK_FLAGS : int - InputType
SF TYPE_MASK_VARIATION : int - InputType
SF TYPE_NULL : int - InputType
SF TYPE_NUMBER_FLAG_DECIMAL : int - InputType
SF TYPE_NUMBER_FLAG_SIGNED : int - InputType
SF TYPE_NUMBER_VARIATION_NORMAL : int - InputType
SF TYPE_NUMBER_VARIATION_PASSWORD : int - InputType
SF TYPE_TEXT_FLAG_AUTO_COMPLETE : int - InputType
SF TYPE_TEXT_FLAG_AUTO_CORRECT : int - InputType
SF TYPE_TEXT_FLAG_CAP_CHARACTERS : int - InputType
SF TYPE_TEXT_FLAG_CAP_SENTENCES : int - InputType
SF TYPE_TEXT_FLAG_CAP_WORDS : int - InputType
```

```
FLAG ime_MULTI_LINE : int - InputType
FLAG_MULTI_LINE : int - InputType
FLAG_NO_SUGGESTIONS : int - InputType
VARIATION_EMAIL_ADDRESS : int - InputType
VARIATION_EMAIL_SUBJECT : int - InputType
VARIATION_FILTER : int - InputType
VARIATION_LONG_MESSAGE : int - InputType
VARIATION_NORMAL : int - InputType
VARIATION_PASSWORD : int - InputType
VARIATION_PERSON_NAME : int - InputType
VARIATION_PHONETIC : int - InputType
VARIATION_POSTAL_ADDRESS : int - InputType
VARIATION_SHORT_MESSAGE : int - InputType
VARIATION_URI : int - InputType
VARIATION_VISIBLE_PASSWORD : int - InputType
VARIATION_WEB_EDIT_TEXT : int - InputType
```



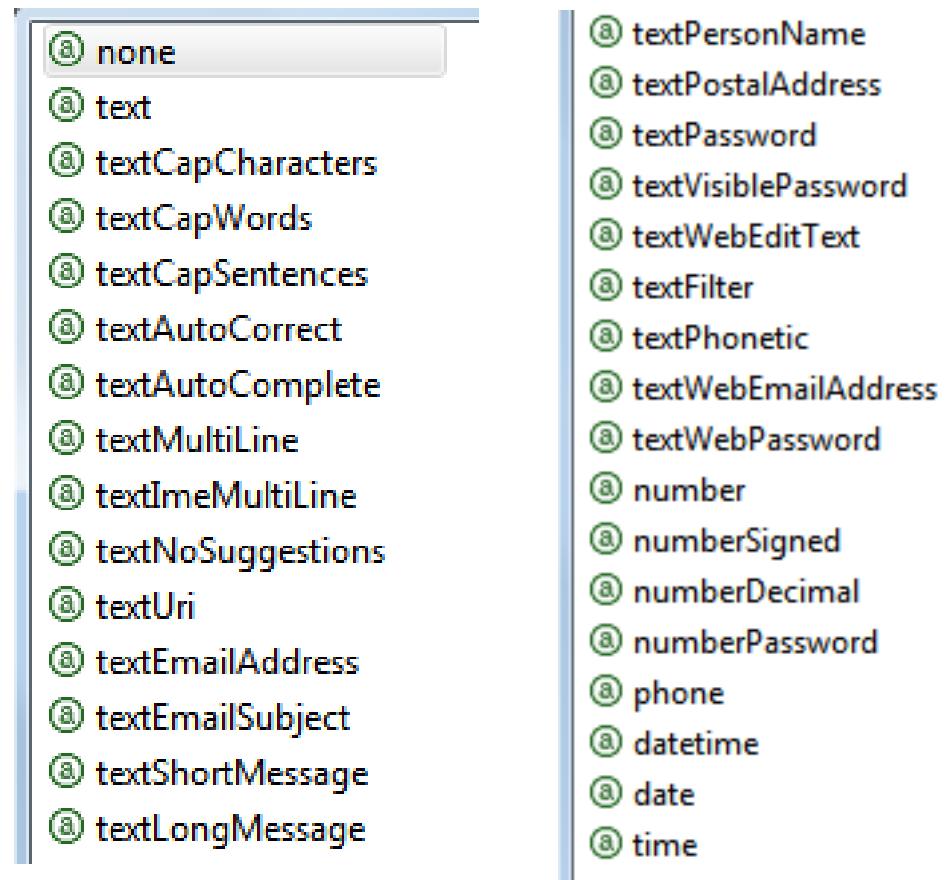
Dodatek B: Pola tekstowe i klawiatury

Określenie inputType z poziomu XML:

<EditText

...

 android:inputType="numberSigned|numberDecimal"
 ... />



Źródło:

http://developer.android.com/reference/android/R.styleable.html#TextView_inputType

Dodatek B: Pola tekstowe i klawiatury

Przykład 10: Wykorzystanie wielu wartości naraz

android:inputType="text|textCapWords"

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

    <TextView android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:text="inputType: text|textCapWords"
        android:textStyle="bold"
        android:textSize="22sp" />

    <EditText android:id="@+id/editTextBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="5dip"
        android:textSize="18sp"
        android:inputType="text|textCapWords"      />
</LinearLayout>
```

Używaj “potoku”
(ozn. |) by
separować opcje.

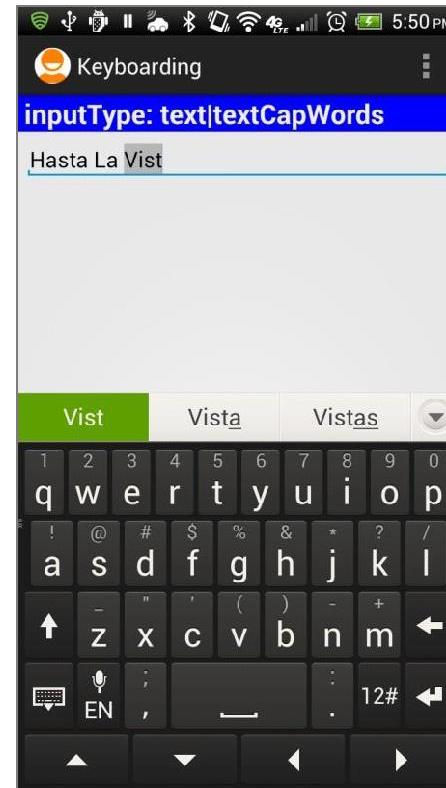
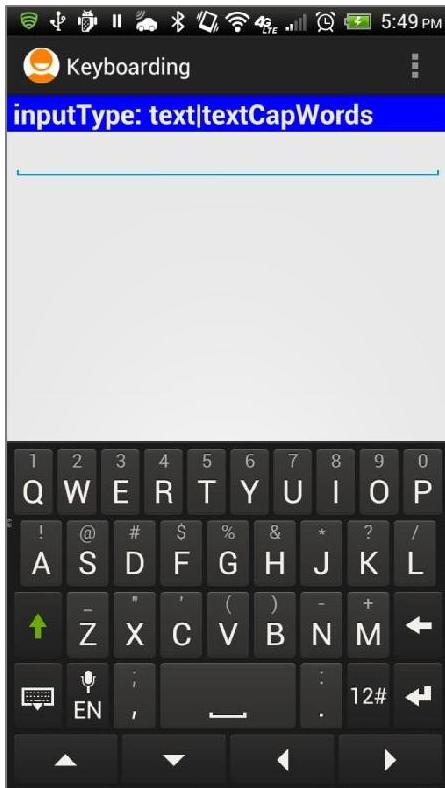
W tym przykładzie
zostanie użyta
wirtualna klawiatura.

Każde słowo
będzie traktowane
jak nazwa własna.



Dodatek B: Pola tekstowe i klawiatury

Przykład 10: Użycie `android:inputType= "text|textCapWords"`



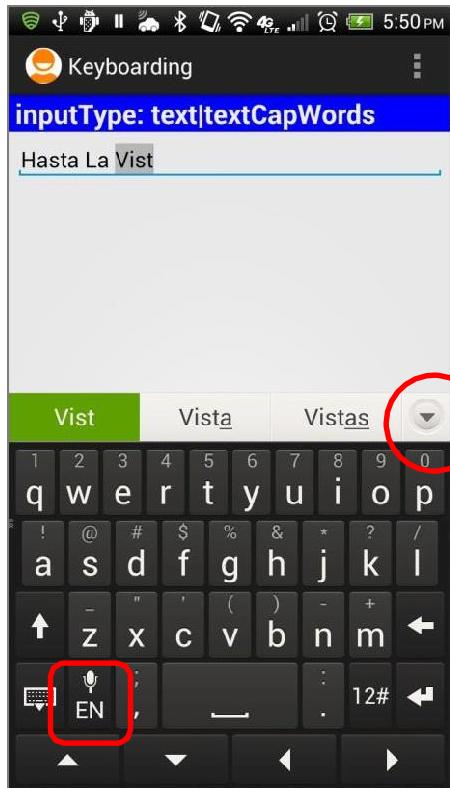
Po kliknięciu na EditText pokazywana jest wirtualna klawiatura z wielkimi literami.

Po wprowadzeniu pierwszej litery klawiatura przechodzi w tryb małych liter.

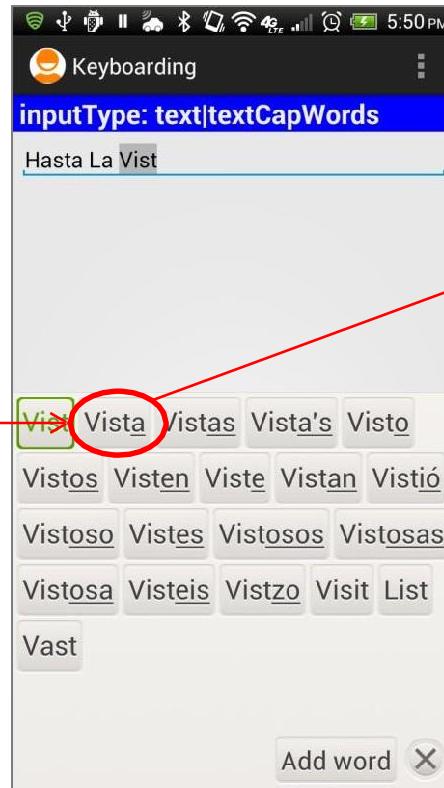
Po wprowadzeniu spacji cykl się powtarza.

Dodatek B: Pola tekstowe i klawiatury

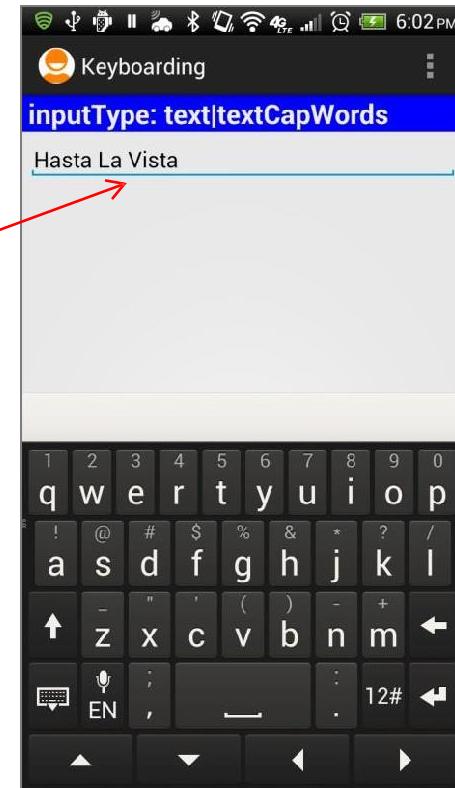
Przykład 10: Użycie `android:inputType="text|textCapWords"`



Angielski i Hiszpański są wybrane jako dostępne języki wprowadzania danych.



Można przyspieszyć wprowadzenie danych korzystając z listy sugestii (dwujęzyczne)

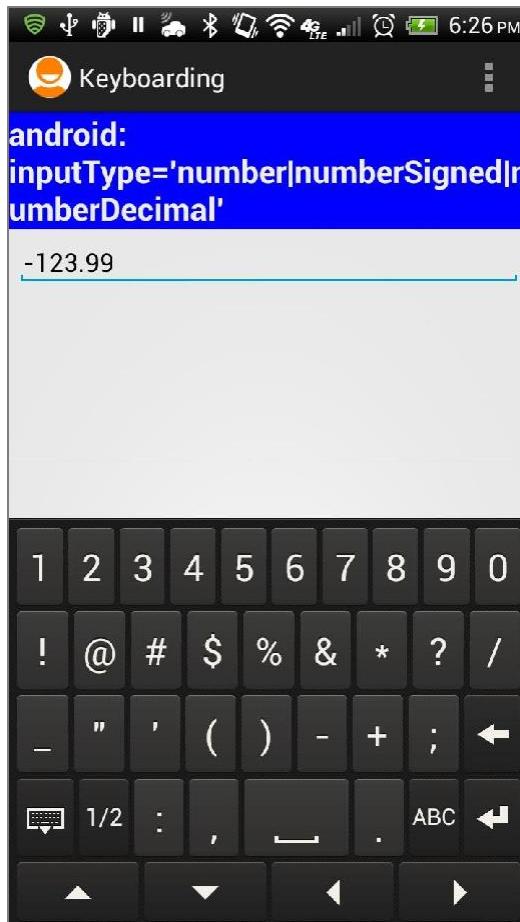


Wybrane słowo wprowadzane jest do pola tekstowego.

Dodatek B: Pola tekstowe i klawiatury

Przykład 11: Wykorzystanie

`android:inputType="number|numberSigned|numberDecimal"`



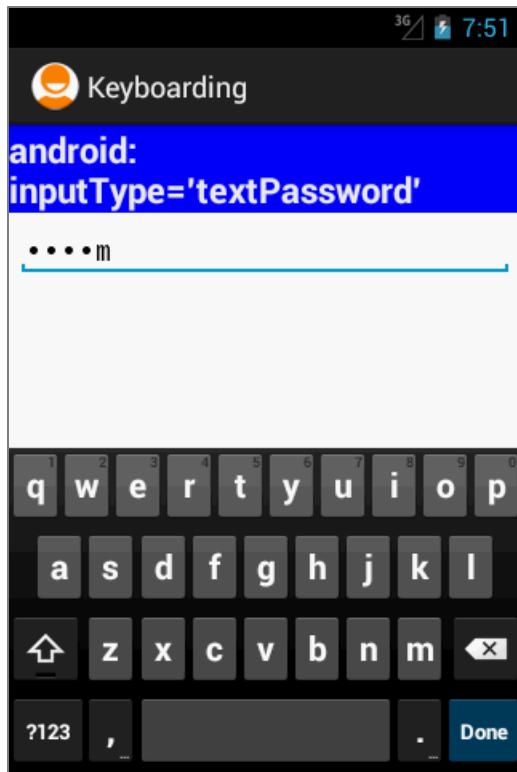
1. Klawiatura wyświetla liczby.
2. *Nie-numeryczne* klawisze (jak !@#\$%&*?/_) mogą być widoczne ale są nieaktywne.
3. Tylko poprane wyrażenia numeryczne mogą być wprowadzone.
4. Typ **number|numberSigned** akceptuje liczby całkowite.
5. Typ **numberDecimal** akceptuje liczby rzeczywiste.

Przy założeniu, że pole tekstowe ma nazwę **editTextBox** ustawienie maski wprowadzania podczas działania programu odbywa się przez:

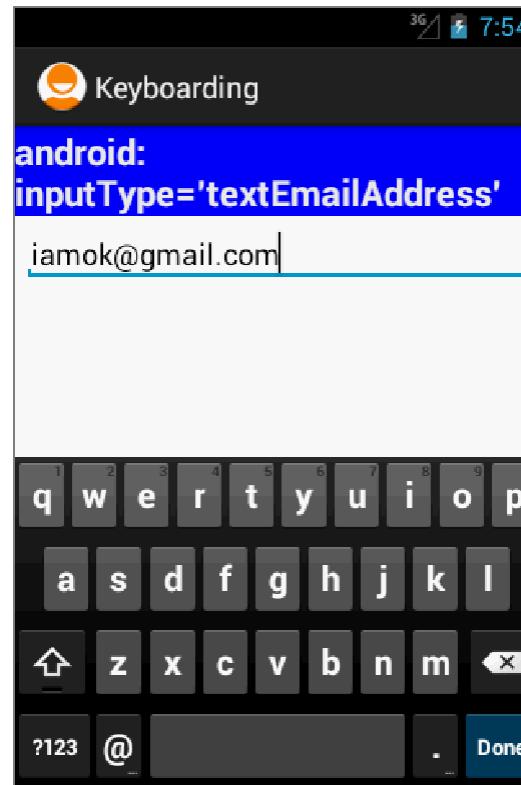
```
editTextBox.setInputType(  
    android.text.InputType.TYPE_CLASS_NUMBER  
    |  
    android.text.InputType.TYPE_NUMBER_FLAG_SIGNED);
```

Dodatek B: Pola tekstowe i klawiatury

Przykład 12: Użycie
`android:inputType="textPassword"`



Przykład 13: Użycie
`android:inputType="textEmailAddress"`



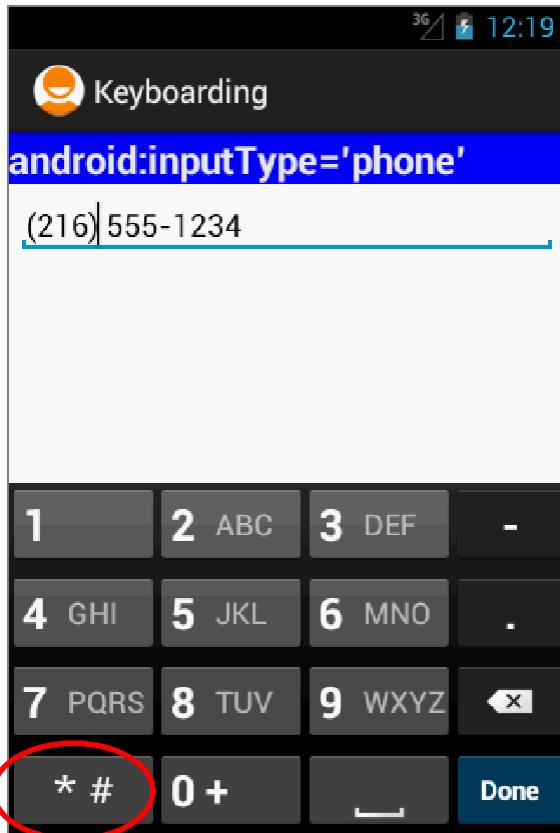
Wirtualna klawiatura pokazuje klawisze używane w komunikacji mailowej (jak litery, @, kropka).

Wybierz [?123] by wyświetlić pozostałe symbole.

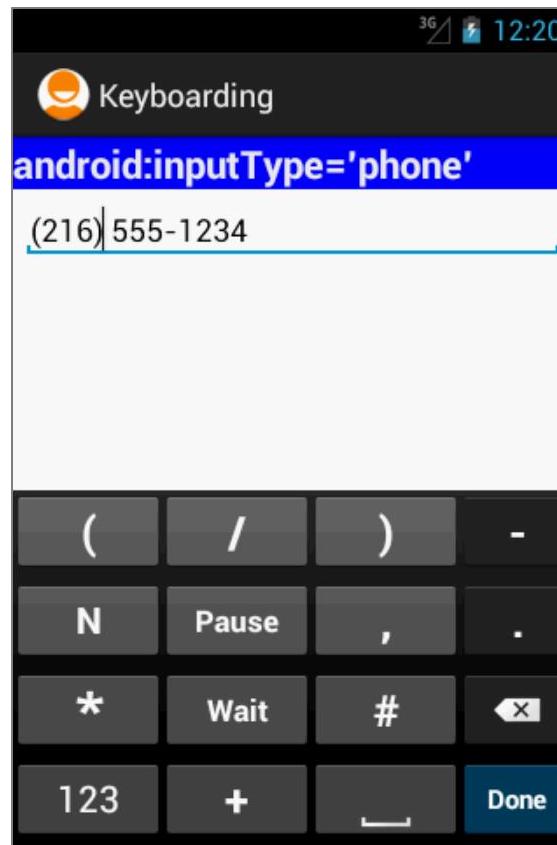
- Klawiatura wyświetla wszystkie klawisze.
- Aktualny znak prezentowany jest na chwilę dla referencji.
- Aktualny znak jest ukrywany za znakiem gwiazdki.

Dodatek B: Pola tekstowe i klawiatury

Przykład 14: Użycie `android:inputType="phone"`



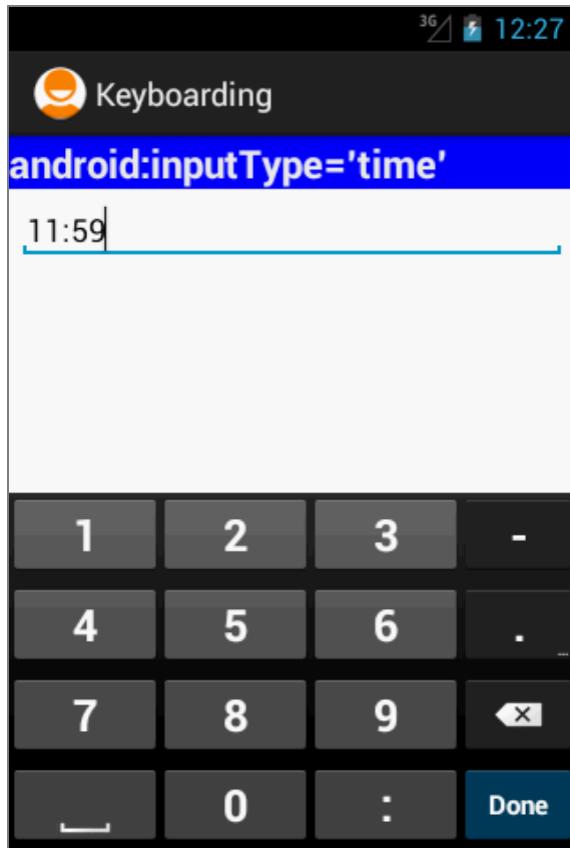
Dodatkowe symbole



Wirtualna klawiatura prezentuje układ typowy dla telefonu plus dodatkowe symbole jak: () . / # - +

Dodatek B: Pola tekstowe i klawiatury

Przykład 15: Użycie `android:inputType="time"`

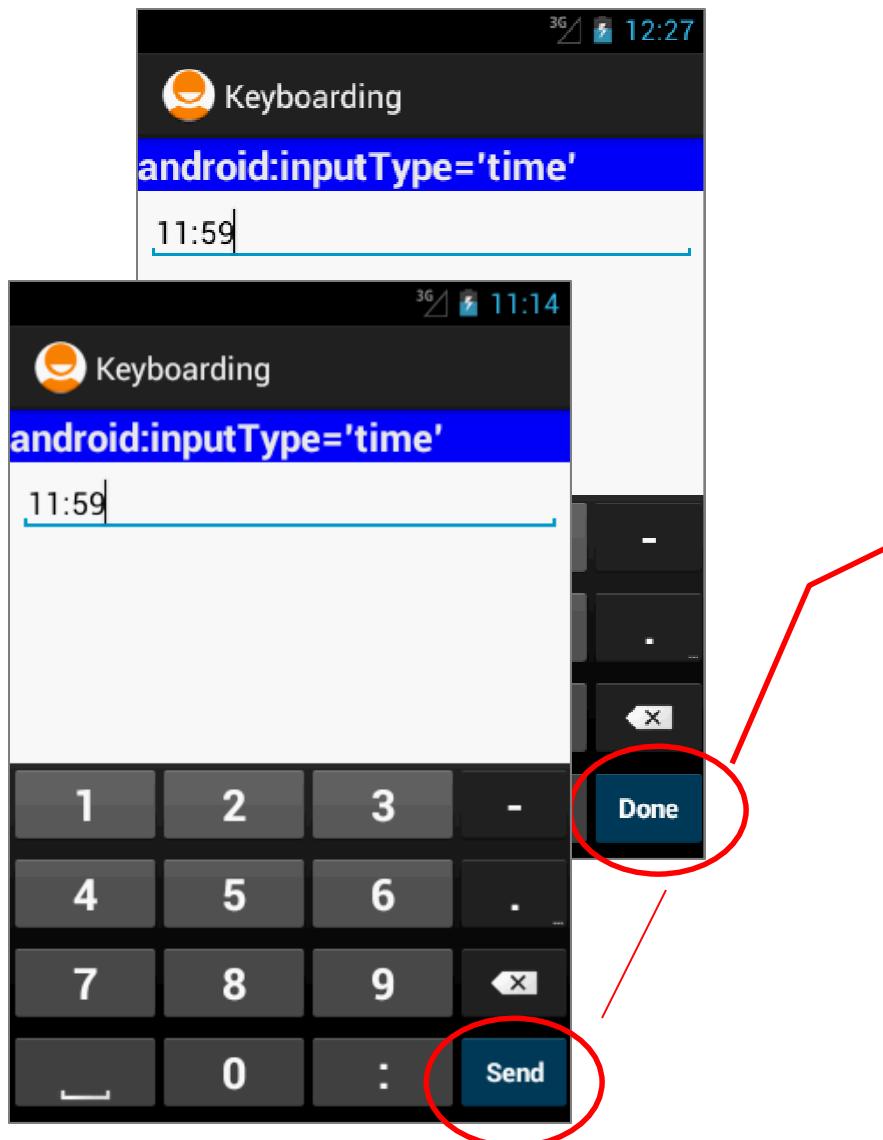


Wirtualna klawiatura prezentuje układ numeryczny.

Tylko numery oraz ":" mogą zostać użyte.

Dodatek B: Pola tekstowe i klawiatury

Przykład 16: Użycie `android:inputType="time"`



Gdy kliknięto, przycisk **dodatkowy DONE** klawiatura zostaje zamknięta.

Można zmienić opis przycisku oraz ustawić nasłuchiwacz by zdefiniować reakcję na zdarzenie kliknięcia przycisku. Należy wówczas dodać do szablonu XML następującą deklarację:

`android:imeAction="actionSend"`

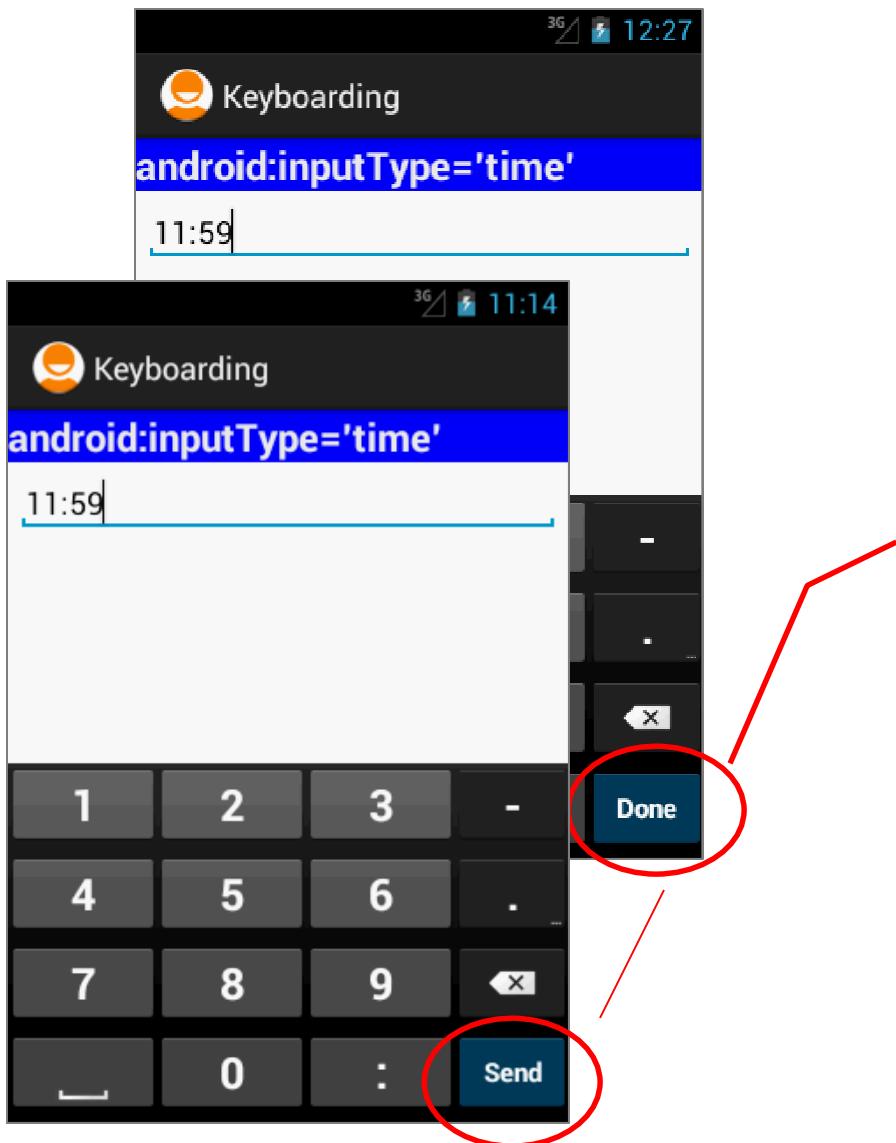
Należy także podać implementację metody:

```
editTextBox  
    .setOnEditorActionListener()
```

by zdefiniować reakcję na to zdarzenie.

Dodatek B: Pola tekstowe i klawiatury

Przykład 16: Użycie `android:inputType="time"`



Inne opcje dla
 `android:imeAction=". . ."`
to

- ⑧ "normal"
- ⑧ "actionUnspecified"
- ⑧ "actionNone"
- ⑧ "actionGo"
- ⑧ "actionSearch"
- ⑧ "actionSend"
- ⑧ "actionNext"
- ⑧ "actionDone"
- ⑧ "actionPrevious"
- ⑧ "flagNoFullscreen"
- ⑧ "flagNavigatePrevious"
- ⑧ "flagNavigateNext"
- ⑧ "flagNoExtractUi"
- ⑧ "flagNoAccessoryAction"
- ⑧ "flagNoEnterAction"
- ⑧ "flagForceAscii"

Dodatek B: Pola tekstowe i klawiatury

Przykład 17: Użycie `android:inputType="datetime"`



Wirtualna klawiatura prezentuje układ numeryczny.

Jedynie liczby i symbole występujące w dacie/czasie są akceptowalne:

Przykłady poprawnych wyrażeń:

12/21/2012 12:12

12/31/2011

12:30

Dodatek B: Pola tekstowe i klawiatury

Wyłączenie wirtualnej klawiatury dla pola tekstowego

- By wyłączyć pokazywanie wirtualnej klawiatury dla pola tekstowego należy ustawić jego input type na null:

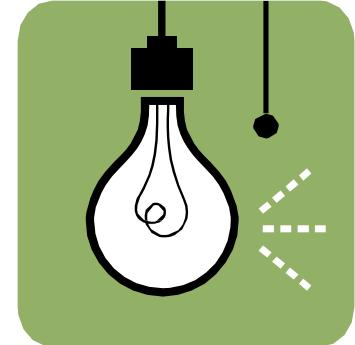
```
editTextBox.setInputType( InputType.TYPE_NULL );
```

- By tymczasowo ukryć wirtualną klawiaturę można wywołać metodę:

```
public void hideVirtualKeyboard() {  
    Context context = getActivity().getApplicationContext();  
    ((InputMethodManager) context  
        .getSystemService(Activity.INPUT_METHOD_SERVICE))  
        .toggleSoftInput(InputMethodManager.SHOW_IMPLICIT, 0);  
}
```

- By wyświetlić wirtualną klawiaturę należy użyć:

```
public void showVirtualKeyboard() {  
    Context context = getActivity().getApplicationContext();  
    ((InputMethodManager) context  
        .getSystemService(Activity.INPUT_METHOD_SERVICE))  
        .toggleSoftInput(InputMethodManager.SHOW_IMPLICIT, 0);  
}
```



Programowanie urządzeń mobilnych

Wykorzystanie fragmentów i tabhost

Tłumaczenie i adaptacja materiałów: dr Tomasz Xięski.
Na podstawie prezentacji udostępnionych przez Victor Matos, Cleveland State University.

Portions of this page are reproduced from work created and [shared by Google and](#) used according to terms

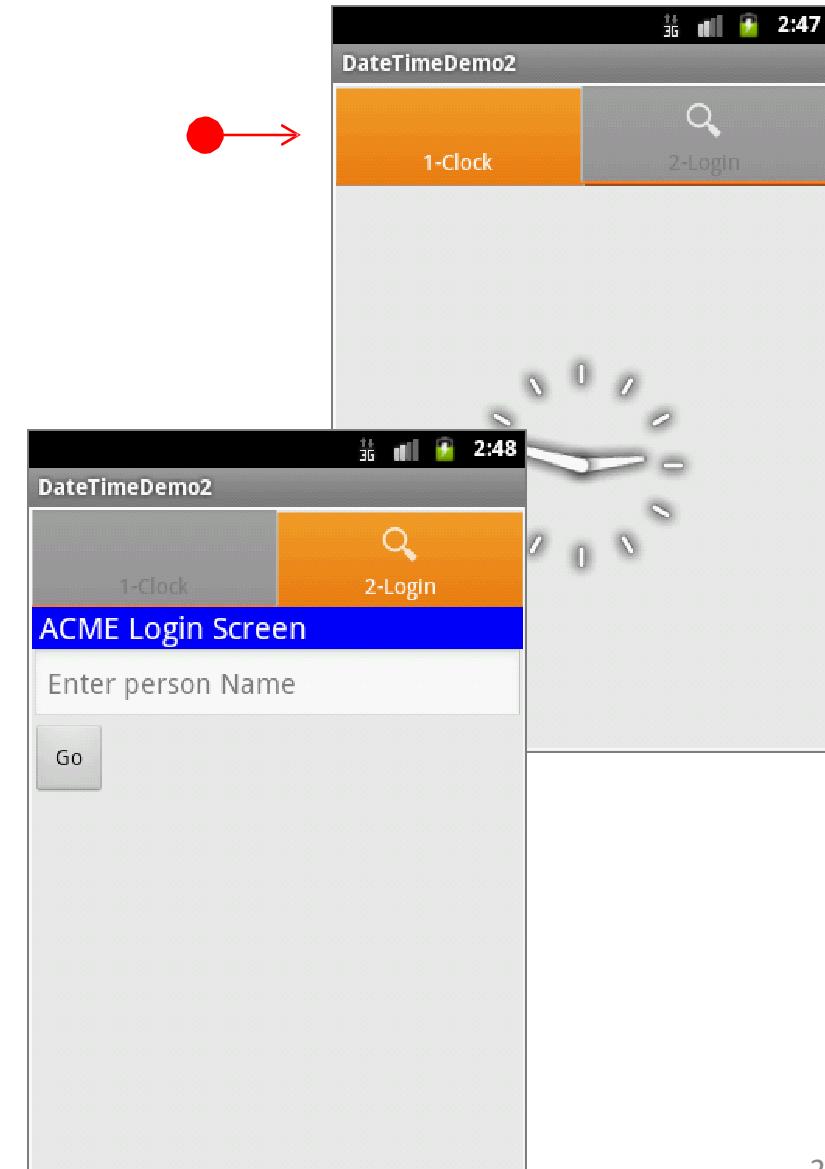
Zakładki TabHost

Zakładki TabHost

1. Zwykle urządzenia mobilne posiadają ograniczony obszar roboczy.
2. Aplikacje, które posiadają skomplikowany interfejs wizualny można posegregować używając **komponentu TabHost** który wyświetla ograniczoną liczbę widżetów jednocześnie.

Informacja:

*Omawiany komponent jest dość rzadko stosowany (rekomendowany dla urządzeń z SDK 4.0 i starszych). Lepsza wersja:
<http://www.android4devs.com/2015/01/how-to-make-material-design-sliding-tabs.html>*

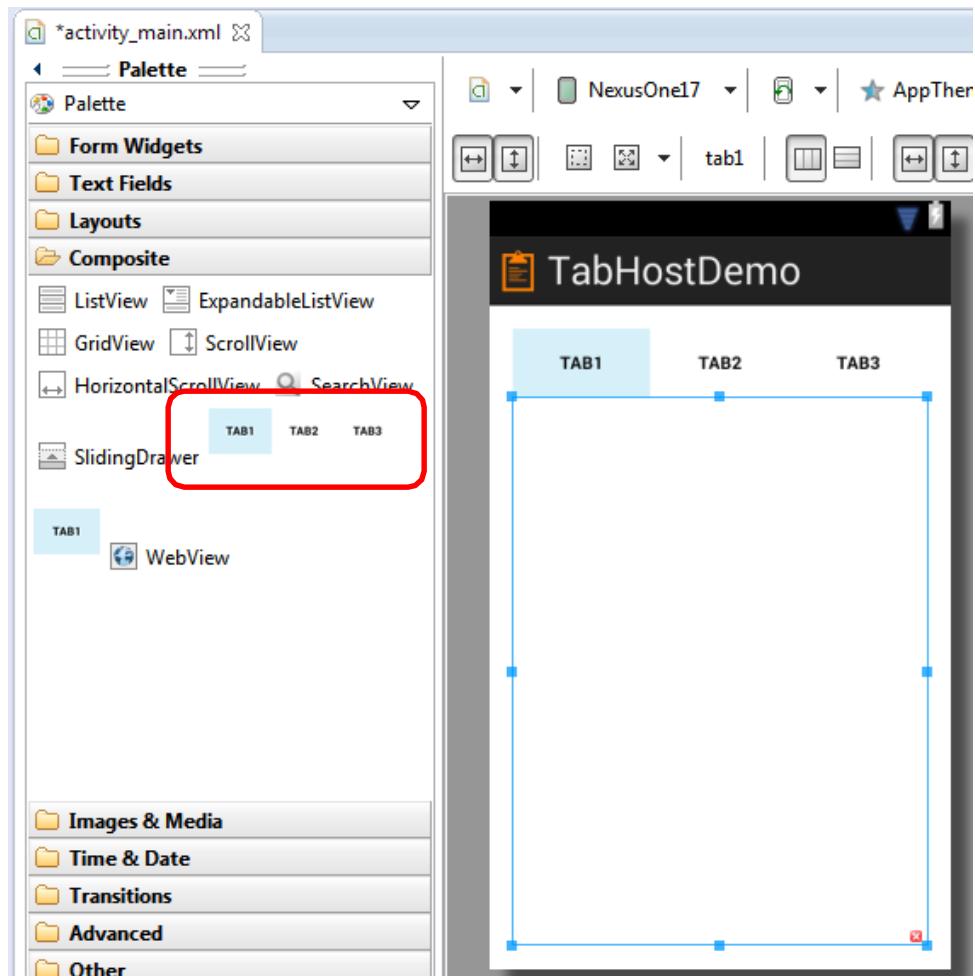


Zakładki TabHost

Struktura TabHost

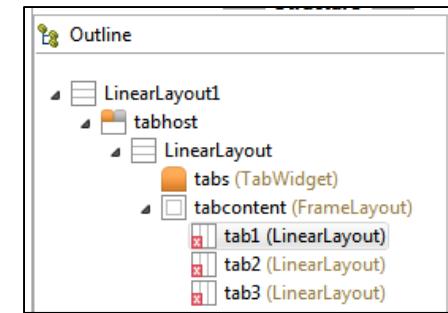
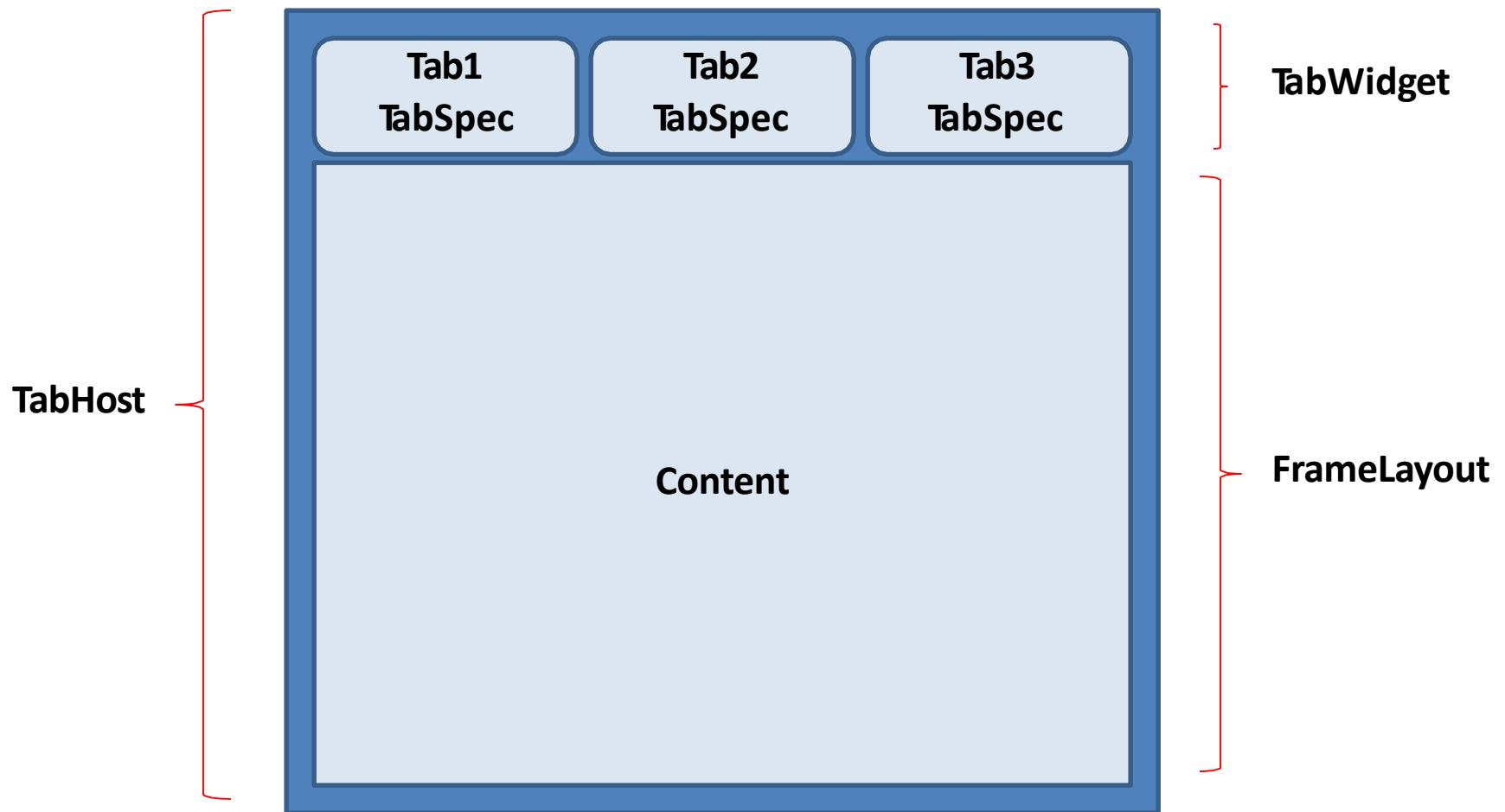
Komponent TabHost w rzeczywistości składa się z 3 pomniejszych widżetów:

1. **TabHost** – główny kontener zawierający zawartość zakładek
2. **TabSpec** – implementuje wiersz przycisków zakładek zawierających tekst (oraz opcjonalnie grafikę)
3. **FrameLayout** – układ wedle którego są pozycjonowane elementy na zakładce



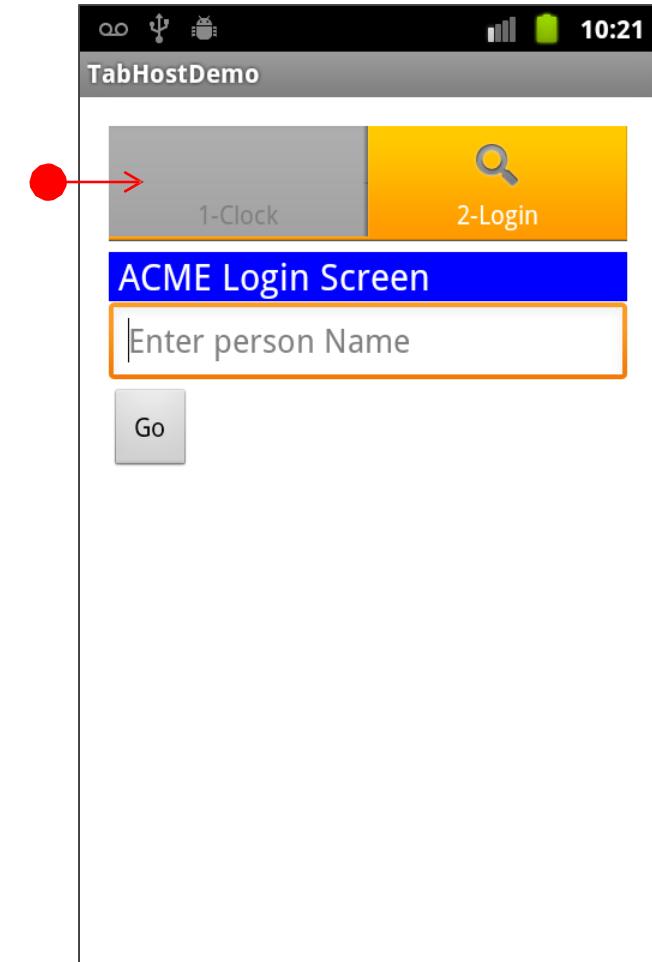
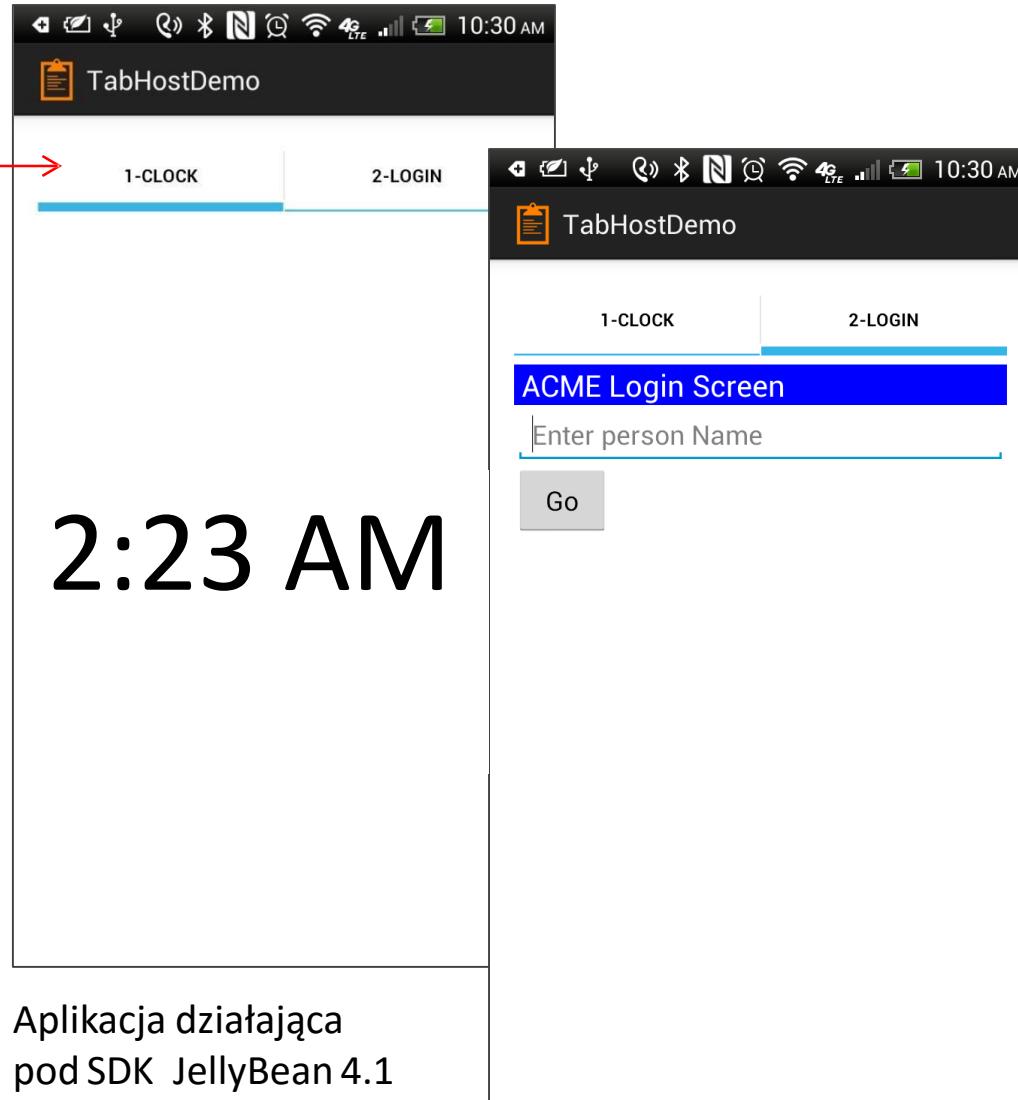
Zakładki TabHost

Struktura TabHost graficznie



Przykład 1: Zakładki TabHost

Wykorzystanie komponentu TabHost



Przykład 1: Zakładki TabHost

Układ XML main_activity.xml

```
<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:background="#ffffffff"               android:orientation="vertical"
    android:padding="2dp"                      tools:context=".MainActivity" >
    <TabHost android:id="@+id/tabhost"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" >
            <TabWidget
                android:id="@+id/tabs"
                android:layout_width="match_parent"
                android:layout_height="wrap_content" >
                </TabWidget>
                <FrameLayout
                    android:id="@+id/tabcontent"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:paddingTop="6dip" >
                    <include layout="@+id/main_tab1" />
                    <include layout="@+id/main_tab2" />
                </FrameLayout>
            </LinearLayout>
        </TabHost>
    </LinearLayout>
```

Można wpisać konkretną specyfikację XML lub dołączyć ją z wykorzystaniem polecenia include

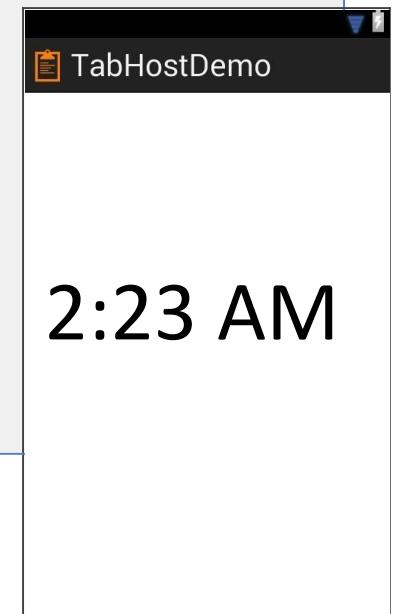
Przykład 1: Zakładki TabHost

Układ XML main_tab1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tab1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextClock android:id="@+id/tab1Clock"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center_horizontal"
        />

</LinearLayout>
```

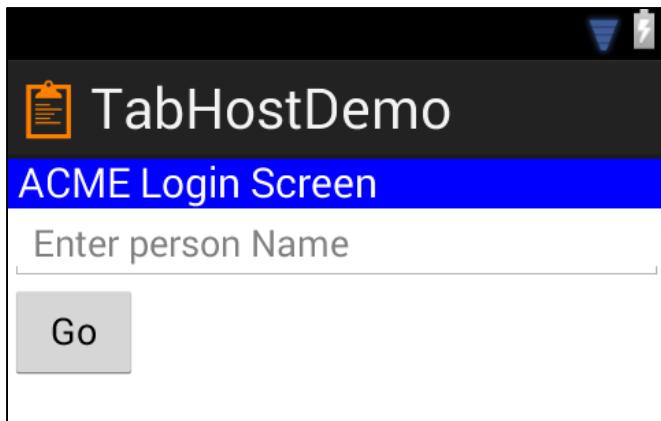


- To jest specyfikacja układu **main_tab1.xml**.
- Dodano do `activity_main.xml` wykorzystując klauzulę:
`<include layout="@Layout/main_tab1" />`
- Ekran zawiera widżet `TextClock`

Przykład 1: Zakładki TabHost

To jest układ **main_tab2.xml**. Definiuje *LinearLayout* zawierający etykietę, pole tekstowe i przycisk.

Dodano do main.xml przez
<include layout=@Layout/... >



XML Layout – TabHostDemo – main_tab2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tab2"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:text=" ACME Login Screen"
        android:textColor="@android:color/white"
        android:textSize="20sp" />

    <EditText
        android:id="@+id/tab2TxtPerson"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="Enter person Name"
        android:inputType="textCapWords"
        android:textSize="18sp" />

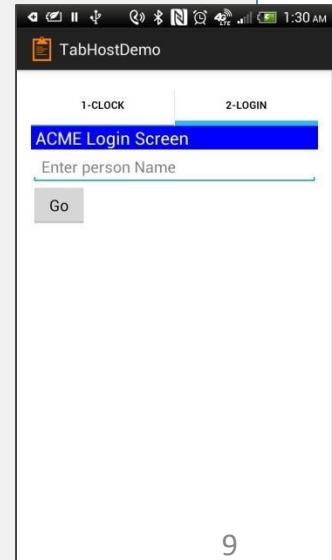
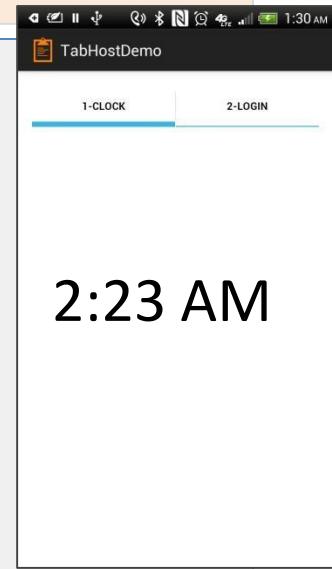
    <Button android:id="@+id/tab2BtnGo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" Go " />

</LinearLayout>
```

Przykład 1: Zakładki TabHost

Klasa ActivityMain

```
public class MainActivity extends Activity {  
  
    TabHost tabhost;  
  
    @Override  
    public void onCreate(Bundle icicle) {  
        super.onCreate(icicle);  
        setContentView(R.layout.activity_main);  
        // wiring UI widgets shown in the various user-layouts  
        // screen-1 components:  
        final AnalogClock clock1 = (AnalogClock) findViewById(R.id.tab1Clock);  
        // screen-2 components:  
        final Button btnGo = (Button) findViewById(R.id.tab2BtnGo);  
        final EditText txtPerson = (EditText) findViewById(R.id.tab2TxtPerson);  
  
        // setting up Tabhost selector  
        tabhost = (TabHost) findViewById(android.R.id.tabhost);  
        tabhost.setup();  
        TabHost.TabSpec tabspec;  
  
        tabspec = tabhost.newTabSpec("screen1");  
        tabspec.setContent(R.id.tab1);  
        tabspec.setIndicator("1-Clock", null);  
        tabhost.addTab(tabspec);  
  
        tabspec = tabhost.newTabSpec("screen2");  
        tabspec.setContent(R.id.tab2);  
        tabspec.setIndicator("2-Login",  
                           getResources().getDrawable(R.drawable.ic_menu_search));  
        tabhost.addTab(tabspec);  
    }  
}
```



Przykład 1: Zakładki TabHost

Klasa ActivityMain

```
tabhost.setCurrentTab(0);
// alternatively, you may also say
// tabhost.setCurrentTabByTag("screen1");
btnGo.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        String theUser = txtPerson.getText().toString();
        txtPerson.setText("Hola " + theUser + "\n" + new Date());
        hideVirtualKeyboard();
    }
});
tabhost.setOnTabChangedListener(new OnTabChangeListener() {
    @Override
    public void onTabChanged(String tabId) {
        // do something useful with the selected screen
        String text = "Im currently on: " + tabId + "\nindex: "
            + tabhost.getCurrentTab();

        switch (tabhost.getCurrentTab()) {
            case 0: // do something for layout-0
                hideVirtualKeyboard();
                break;
            case 1: // do something for layout-1
                break;
        }
        Toast.makeText(getApplicationContext(), text, 1).show();
    }
});
```

Reakcja na kliknięcie przycisku GO

Reakcja na wybór określonej zakładki

Przykład 1: Zakładki TabHost

Klasa ActivityMain

```
// onCreate

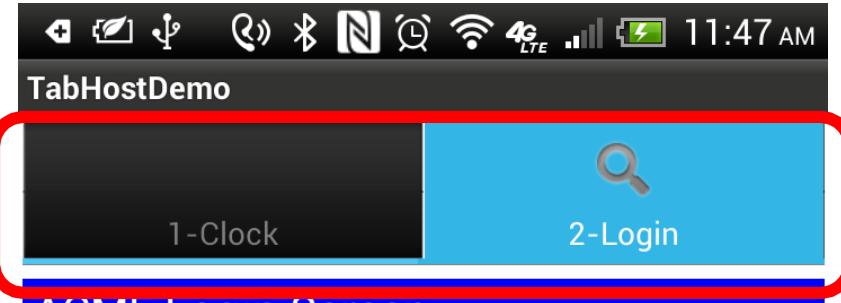
public void hideVirtualKeyboard() {
    // temporarily remove the virtual keyboard
    ((InputMethodManager) getSystemService(Activity.INPUT_METHOD_SERVICE))
        .toggleSoftInput(InputMethodManager.SHOW_IMPLICIT, 0);
}

public void showVirtualKeyboard() {
    // no used - shown for completeness
    ((InputMethodManager) getSystemService(Activity.INPUT_METHOD_SERVICE))
        .toggleSoftInput(InputMethodManager.SHOW_IMPLICIT, 0);
}
```

Przykład 1: Zakładki TabHost

Dodawanie grafiki do zakładek

Można udekorować zakładki dodając do nich
prócz tekstu dowolną grafikę w sposób
określony poniżej:



```
tabspec = tabhost.newTabSpec("screen2");  
  
tabspec.setContent(R.id.tab2);  
  
tabspec.setIndicator("2-Login",  
    getResources().getDrawable(R.drawable.ic_action_search));  
  
tabhost.addTab(tabspec);
```



Wskazówka:

Spróbuj poeksperymentować z wyglądem aplikacji zmieniając w pliku AndroidManifest.xml
w znaczniku <Application> klauzulę: android:theme="@android:style/Theme.Black"

Wskazówka:

Wiele ikon dostępnych jest w: <android-sdk-folder\docs\images\icon-design>

Zobacz także: <http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html>

Tworzenie skomplikowanych GUI

- Głównym celem platformy Android jest zaoferowanie przyjemnego wizualnie, bogatego, intuicyjnego i jednorodnego doświadczenia użytkownika.
- Jest to wyzwanie dla projektantów. Twoje aplikacje powinny zapewnić poczucie intuicyjnej obsługi oraz zapewnić odpowiedni poziom estetyki.
- Wdrożenie zestawu rekomendacji zwanych **Material Design** jest wysoce rekomendowane przez twórców platformy dla zachowania jednolitego wyglądu całego systemu.
- Poprzez wdrożenie tych sugestii wszystkie nowe aplikacje będą wydawać się dla użytkownika jednorodne a nawigacja po nich będzie zgodna z oczekiwaniami.
- W następnych przykładach zaprezentowany zostanie podstawowy budulec wpisujący się w zestaw tych rekomendacji – wykorzystanie fragmentów.



Fragmenty

- Android jest wielowątkowym systemem operacyjnym, a specyfikacje obecnych urządzeń pozwalają na osiągnięcie pełnej wielozadaniowości. Jednakże w danym czasie tylko jedna aktywność może być "widoczna" i "aktywna". Ten fakt stanowi jednak poważne ograniczenie biorąc pod uwagę przestrzeń roboczą oferowaną przez urządzenia o dużym ekranie (tablety, telewizory itp). Fragmenty oferują częściowe rozwiązanie problemu.
- Klasa **Fragment** dostarcza obiektów które mogą być *dołączane dynamicznie* do określonych miejsc GUI. Każdy fragment posiada unikalną warstwę widoku oraz sposób interakcji z użytkownikiem.
- Fragmenty **muszą** być powiązane z konkretną aktywnością, która pełni rolę gospodarza bądź bazy.
- W ramach jednej aktywności może istnieć wiele fragmentów. Każdy fragment może być wówczas w stanie aktywnym oraz widoczny.

Fragmenty

- Fragmenty zachowują się jak niezależne od siebie wątki, mogące wchodzić w kooperację by osiągnąć wyznaczony cel; jednakże każdy z nich posiada indywidualną logikę biznesową i sposób reakcji na pojawiające się zdarzenia.
- Fragmenty mogą uzyskać dostęp do *danych globalnych* dostępnych w ramach aktywności do której zostały przypisane. Podobnie mogą wysyłać swoje dane do głównej aktywności, jako forma wymiany z innymi fragmentami.
- Fragmenty posiadają swój własny cykl życia, w ramach którego metoda **onCreateView** wykonuje wszelkie niezbędne czynności by powołać dany fragment do życia.
- Fragmenty po raz pierwszy zostały wprowadzone w Honeycomb SDK (API 11).

Fragmenty

ACTIVITY (Główny kontener)

Fragment1
(View 1)

Fragment2
(View 2)

Fragment3
(View 3)

Możliwa aranżacja fragmentów w ramach zadanej aktywności

Cykl życia fragmentu

onAttach() Jest wywoływana gdy fragment został osadzony w aktywności gospodarza.

onCreate() Służy do inicjalizacji składników bez wizualnej reprezentacji potrzebnych przez fragment

onCreateView() Używany do stworzenia całej hierarchii danego widoku. Zazwyczaj tworzone są tutaj instancje obiektów oraz dopisywane nasłuchiwacze.

onPause() Sesja z aktywnością dobiera końca. Tutaj należy zapisać stan fragmentu, potrzeby w przypadku jego ewentualnej, ponownej inicjalizacji.

onDetach() Jest wywoływana, gdy nieaktywny fragment ma zostać odłączony od powiązanej aktywności.

Activity State Fragment Callbacks

Created



onAttach()

onCreate()

onCreateView()

onActivityCreated()

Started

onStart()

Resumed

onResume()

Paused

onPause()

Stopped

onStop()

Destroyed

onDestroyView()

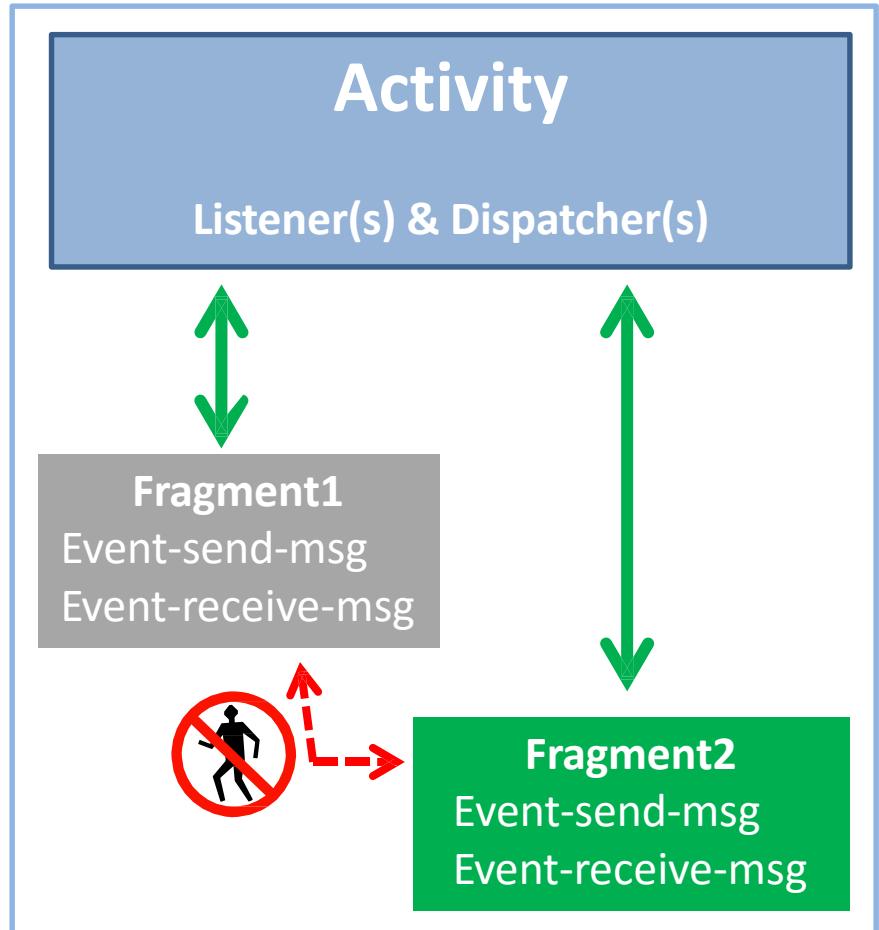
onDestroy()

onDetach()

Fragmenty

Komunikacja między fragmentami

- Cała komunikacja fragment-fragment odbywa się w sposób skoncentrowany przez aktywność bazową.
- Dwa fragmenty **NIGDY nie powinny** komunikować się bezpośrednio..
- Aktywność bazowa oraz jej fragmenty komunikują się ze sobą poprzez nasłuchiwanie i zdarzenia.
- Gdy fragment posiada dane przeznaczone dla innego fragmentu, przesyła je do głównej aktywności, która przekazuje je do drugiego fragmentu.



Integracja aktywności bazowej i jej fragmentów

Generalnie **fragmenty** umieszczane są na widoku aktywności wykorzystując następujące podejścia:

Dynamiczne wiązanie

Fragment ma zazwyczaj zdefiniowane miejsce na głównej aktywności. Zajętość tego miejsca nie jest stała tzn. później aktywność bazowa może podmienić fragment w tym miejscu (patrz **Przykład 1**)

Statyczne wiązanie

Aktywność deklaruje porcje swojego ekranu jako <fragment> oraz bezpośrednio wskazuje referencję do fragmentu, który ma zostać tam osadzony wykorzystując klauzulę “android:name=fragmentName”. Wówczas nie jest wymagane wywoływanie odpowiednich konstruktorów. Jednakże statyczne wiązanie jest permanentne – fragmenty nie mogą być podmieniane podczas działania programu (patrz **Przykład-2**)

Wiele fragmentów

Aktywność może osadzić wiele fragmentów używając dowolnej kombinacji powyższych strategii. Fragmenty mogą wchodzić w interakcję poprzez wspomnianą aktywność bazową (patrz **Przykład-3**).

Fragmenty

Dynamiczne wiązanie

- Fragmenty muszą zostać stworzone w bloku **FragmentTransaction**.
- Można wykorzystać metodę **add()** by dodać fragment do danej aktywności. Widoki stworzone w ramach fragmentu umieszczane są w danym miejscu aktywności.
- Gdy używana jest metoda **replace** by odświeżyć GUI, aktualny widok jest *usuwany* i nowy fragment jest *dodawany* do interfejsu aktywności.
- Fragment nie posiadający reprezentacji wizualnej również *może zostać dodany* do aktywności, nie tworząc hierarchii widoków.

KROKI

1. Uzyskaj referencję do obiektu *FragmentManager* i rozpoczęj transakcję:

```
FragmentTransaction ft= getFragmentManager().beginTransaction();
```

2. Stwórz instancję danego fragmentu.

```
FragmentBlue blueFragment= FragmentBlue.newInstance("some-value");
```

3. Umieść fragment w zadanym miejscu aktywności.

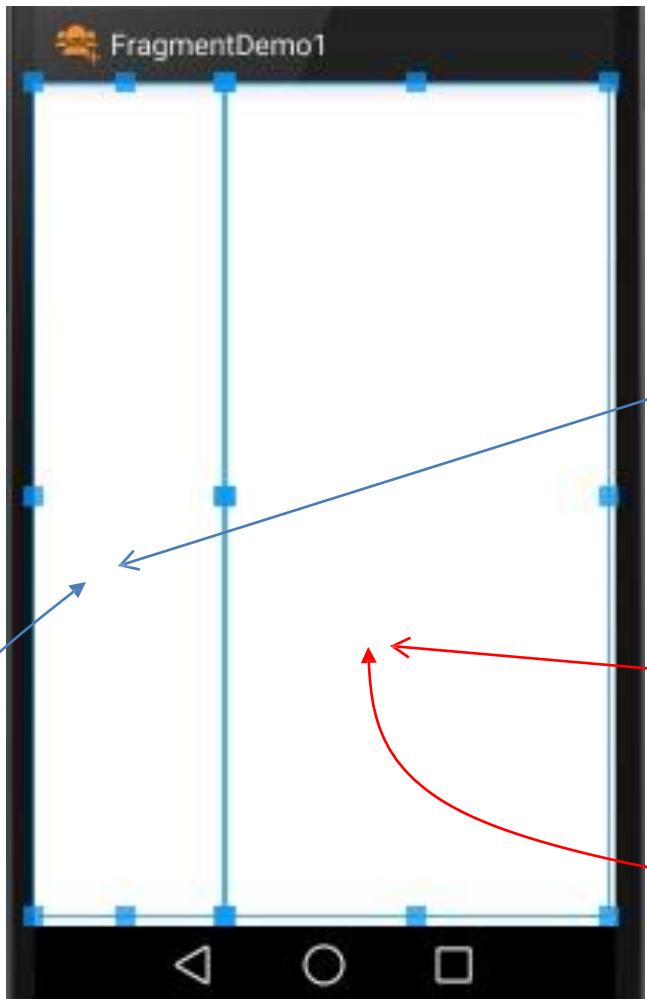
```
ft.replace(R.id.main_holder_blue, blueFragment);
```

4. Zakończ transakcję.

```
ft.commit();
```

Fragmenty

Integracja aktywności i fragmentów

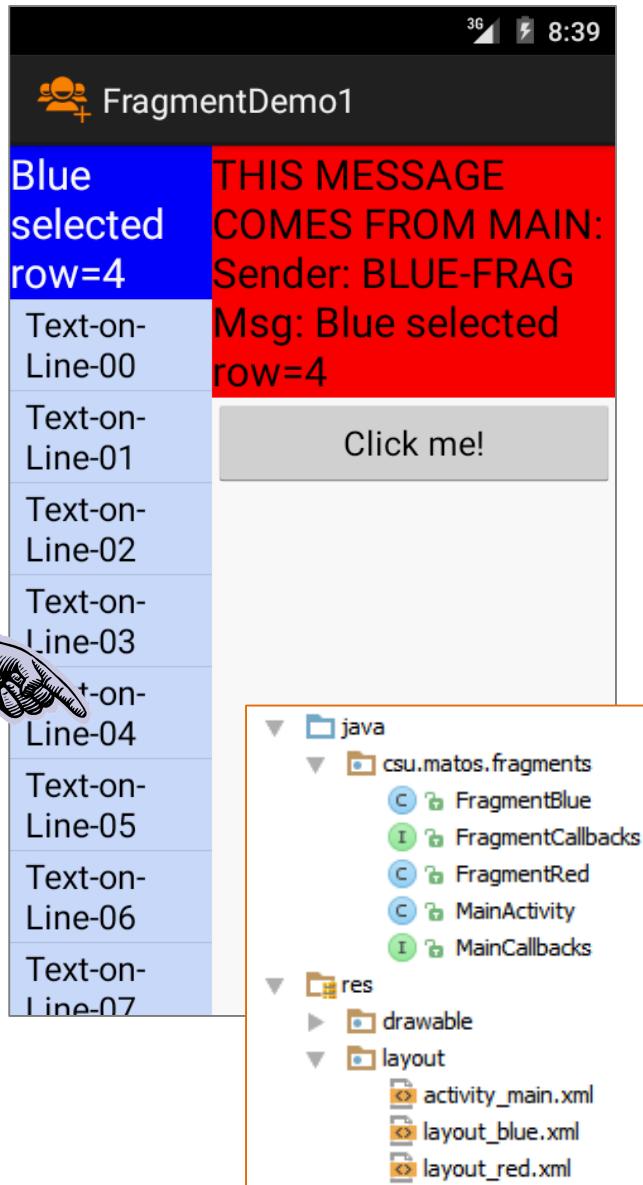


Przykład dynamicznego wiązania. Instancje klas **FragmentRed** i **FragmentBlue** tworzone są w trakcie działania programu, by zastąpić lewą i prawą stronę ekranu.

```
// create a new BLUE fragment - show it
ft = getFragmentManager().beginTransaction();
blueFragment = FragmentBlue.newInstance("new-blue");
ft.replace(R.id.main_holder_blue, blueFragment);
ft.commit();
```

```
// create a new RED fragment - show it
ft = getFragmentManager().beginTransaction();
redFragment = FragmentRed.newInstance("new-red");
ft.replace(R.id.main_holder_red, redFragment);
ft.commit();
```

Przykład 1 – Dynamiczne tworzenie fragmentów



Przykład ilustruje działanie wzorca typu master-detail. Składa się z 3 klas:

- **MainActivity** (bazowa),
- **FragmentRed** (detail)
- **FragmentBlue** (master)

Fragment **niebieski** prezentuje listę elementów. Gdy użytkownik wybierze jeden z nich, wiadomość jest wysyłana do głównej aktywności **MainActivity** która przekazuje ją dalej do czerwonego fragmentu (po prawej stronie).

Fragment **czerwony** przytacza dane szczegółowe dotyczące wybranej pozycji z listy.

Interfejsy **FragmentCallbacks** i **MainCallbacks** definiują metody używane do komunikacji między fragmentami a aktywnością i odwrotnie.

Przykład 1 – Dynamiczne tworzenie fragmentów

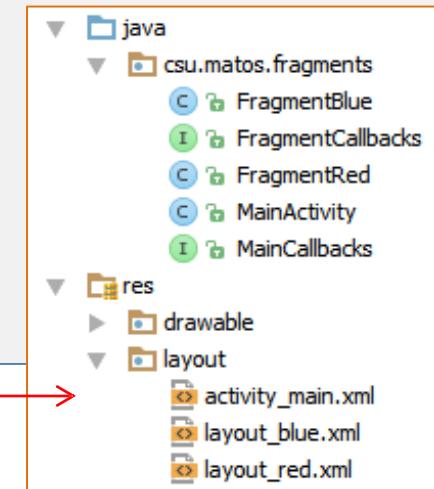
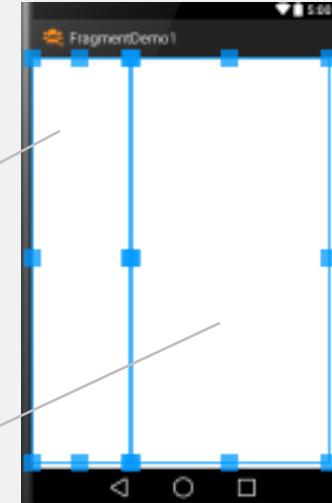
UKŁAD XML: activity_main.xml

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

    <FrameLayout
        android:id="@+id/main_holder_blue"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="vertical" />

    <FrameLayout
        android:id="@+id/main_holder_red"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="2"
        android:orientation="vertical" />

</LinearLayout>
```



Przykład 1 – Dynamiczne tworzenie fragmentów

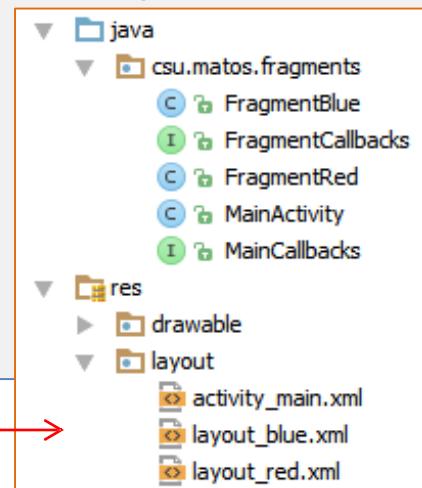
UKŁAD XML: layout_blue.xml

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

    <TextView
        android:id="@+id/textView1Blue"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Blue Layout..."
        android:textColor="#ffffffff"
        android:background="#ff0000ff"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <ListView
        android:id="@+id/listView1Blue"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>

</LinearLayout>
```



Przykład 1 – Dynamiczne tworzenie fragmentów

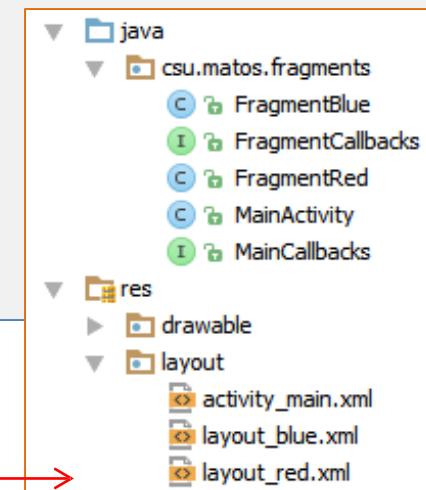
UKŁAD XML: layout_red.xml

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

    <TextView
        android:id="@+id/textView1Red"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffff0000"
        android:text="Red Layout..." 
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <Button
        android:id="@+id/button1Red"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:ems="10"
        android:text="Click me!" />

</LinearLayout>
```



Przykład 1 Klasa MainActivity

1 z 3

```
// -----
// GOAL: This example shows an Activity that includes two fragments.
// Fragments inflate layouts and then get attached to their corresponding
// layouts in the UI. The example includes two interfaces MainCallbacks
// and FragmentCallbacks. They implement inter-process communication from
// Main-to-fragments and from Fragments-to-Main.
// -----
public class MainActivity extends Activity implements MainCallbacks {

    FragmentTransaction ft;
    FragmentRed redFragment;
    FragmentBlue blueFragment;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // create a new BLUE fragment - show it
        ft = getFragmentManager().beginTransaction();
        blueFragment = FragmentBlue.newInstance("first-blue");
        ft.replace(R.id.main_holder_blue, blueFragment);
        ft.commit();

        // create a new RED fragment - show it
        ft = getFragmentManager().beginTransaction();
        redFragment = FragmentRed.newInstance("first-red");
        ft.replace(R.id.main_holder_red, redFragment);
        ft.commit();

    }
}
```



```
// MainCallback implementation (receiving messages coming from Fragments)
@Override
public void onMsgFromFragToMain(String sender, String strValue) {
    // show message arriving to MainActivity
    Toast.makeText(getApplicationContext(),
        " MAIN GOT>> " + sender + "\n" + strValue, Toast.LENGTH_LONG)
        .show();

    if (sender.equals("RED-FRAG")) {
        // TODO: if needed, do here something on behalf of the RED fragment
    }

    if (sender.equals("BLUE-FRAG")) {
        try {
            // forward blue-data to redFragment using its callback method
            redFragment.onMsgFromMainToFragment("\nSender: " + sender
                + "\nMsg: " + strValue);

        } catch (Exception e) {
            Log.e("ERROR", "onStrFromFragToMain " + e.getMessage());
        }
    }
}
```

Komentarze

1. Każdy fragment tworzony jest w bezpiecznym bloku transakcyjnym:
beginTransaction ... commit.
2. Wywołanie metody **newInstance** powoduje przekazanie do konstruktora fragmentu wszystkich wymaganych informacji.
3. Po stworzeniu, nowy fragment **zastępuje** cokolwiek jest aktualnie wyświetlane w określonym uprzednio miejscu GUI.
4. Metoda **onMsgFromFragToMain** implementuje interfejs MainCallbacks. Akceptuje on wiadomości wysyłane asynchronicznie do głównej aktywności przez czerwony lub niebieski fragment.
5. W naszym przykładzie, numer wybranego wiersza z niebieskiego fragmentu jest przekazywany do czerwonego fragmentu wykorzystując metodę **onMsgFromMainToFragment**.

Przykład 1 - FragmentBlue

1 z 3

```
public class FragmentBlue extends Fragment {  
    // this fragment shows a ListView  
    MainActivity main;  
    Context context = null;  
    String message = "";  
  
    // data to fill-up the ListView  
    private String items[] = { "Text-on-Line-00", "Text-on-Line-01",  
        "Text-on-Line-02", "Text-on-Line-03", "Text-on-Line-04",  
        "Text-on-Line-05", "Text-on-Line-06", "Text-on-Line-07",  
        "Text-on-Line-08", "Text-on-Line-09", "Text-on-Line-10", };  
  
    // convenient constructor(accept arguments, copy them to a bundle, binds bundle to fragment)  
    public static FragmentBlue newInstance(String strArg) {  
        FragmentBlue fragment = new FragmentBlue();  
        Bundle args = new Bundle();  
        args.putString("strArg1", strArg);  
        fragment.setArguments(args);  
        return fragment;  
    }  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        try {  
            context = getActivity(); // use this reference to invoke main callbacks  
            main = (MainActivity) getActivity();  
        } catch (IllegalStateException e) {  
            throw new IllegalStateException(  
                "MainActivity must implement callbacks");  
        }  
    }  
}
```

Przykład 1 - FragmentBlue

2 z 3

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {

    // inflate res/layout_blue.xml to make GUI holding a TextView and a ListView
    LinearLayout layout_blue = (LinearLayout) inflater.inflate(R.layout.layout_blue, null);
    // plumbing - get a reference to textView and listView
    final TextView txtBlue = (TextView) layout_blue.findViewById(R.id.textView1Blue);
    ListView listView = (ListView) layout_blue.findViewById(R.id.listView1Blue);
    listView.setBackgroundColor(Color.parseColor("#ffccddff"));
    // define a simple adapter to fill rows of the listView
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(context,
                                                                android.R.layout.simple_list_item_1, items);
    listView.setAdapter(adapter);
    // show listView from the top
    listView.setSelection(0);
    listView.smoothScrollToPosition(0);
    // react to click events on listView's rows
    listView.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
            // inform enclosing MainActivity of the row's position just selected
            main.onMsgFromFragToMain("BLUE-FRAG", "Blue selected row=" + position);
            txtBlue.setText("Blue selected row=" + position);
        }
    });
    // do this for each row (ViewHolder-Pattern could be used for better performance!)
    return layout_blue;
}

// onCreateView
}// class
```

Komentarze

1. Metoda `Class.newInstance(...)` stanowi praktyczną implementację wzorca projektowego – prostej fabryki – wykorzystywanego do tworzenia nowych instancji.
2. Tworzenie nowego fragmentu rozpoczyna się od stworzenia nowego obiektu typu `Bundle`, który przechowuje argumenty w formie par `<klucz, wartość>`. Obiekt jest wówczas wiązany z fragmentem poprzez metodę `.setArguments(...)`. Ostatecznie nowa instancja fragmentu jest zwracana.
3. Metoda **onCreate** weryfikuje, że główna aktywność implementuje interfejs definiujący metody określający sposób komunikacji aktywności z fragmentem.
4. Fragmenty wykonują większość pracy w metodzie **onCreateView**. W tym przykładzie tworzone są instancje obiektów na podstawie układu `layout_blue.xml` oraz definiowane są reakcje na zdarzenia.
5. Prosty `ArrayAdapter` wykorzystywany jest by wypełnić `ListView`.
6. Gdy użytkownik kliknie pozycję na liście jest ona wysyłana do metody **onMsgFromFragToMain** aktywności głównej.

Przykład 1 – FragmentRed

1 z 3

```
public class FragmentRed extends Fragment implements FragmentCallbacks {  
    MainActivity main;  
    TextView txtRed;  
    Button btnRedClock;  
  
    public static FragmentRed newInstance(String strArg1) {  
        FragmentRed fragment = new FragmentRed();  
        Bundle bundle = new Bundle();  
        bundle.putString("arg1", strArg1);  
        fragment.setArguments(bundle);  
        return fragment;  
    } // newInstance  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // Activities containing this fragment must implement interface: MainCallbacks  
        if (!(getActivity() instanceof MainCallbacks)) {  
            throw new IllegalStateException( " Activity must implement MainCallbacks");  
        }  
        main = (MainActivity) getActivity(); // use this reference to invoke main  
        callbacks  
    }  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                            Bundle savedInstanceState) {  
        // inflate res/layout_red.xml which includes a textView and a button  
        LinearLayout view_layout_red = (LinearLayout) inflater.inflate(  
                R.layout.layout_red, null);  
        // plumbing - get a reference to widgets in the inflated layout  
        txtRed = (TextView) view_layout_red.findViewById(R.id.textView1Red);  
    }  
}
```

Przykład 1 – FragmentRed

2 z 3

```
// show string argument supplied by constructor (if any!)
try {
    Bundle arguments = getArguments();
    String redMessage = arguments.getString("arg1", "");
    txtRed.setText(redMessage);
} catch (Exception e) {
    Log.e("RED BUNDLE ERROR - ", "" + e.getMessage());
}
// clicking the button changes the time displayed and sends a copy to MainActivity
btnRedClock = (Button) view_layout_red.findViewById(R.id.button1Red);
btnRedClock.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        String redMessage = "Red clock:\n" + new Date().toString();
        txtRed.setText(redMessage);
        main.onMsgFromFragToMain("RED-FRAG", redMessage);
    }
});
return view_layout_red;
}

@Override
public void onMsgFromMainToFragment(String strValue) {
    // receiving a message from MainActivity (it may happen at any point in time)
    txtRed.setText("THIS MESSAGE COMES FROM MAIN:" + strValue);
}

}// FragmentRed
```

Komentarze

1. Prezentowany proces jest bardzo podobny do założeń poczynionych dla klasy FragmentBlue.
2. Jak poprzednio, metoda *newInstance* jest wywoływana by stworzyć instancję fragmentu.
3. Klasa FragmentRed wykorzystuje metodę *onCreate* by zweryfikować, że główna aktywność implementuje metody potrzebne do wysyłania do niego komunikatów.
4. Obiekt typu FragmentRed otrzymuje komunikaty z głównej aktywności w sposób asynchroniczny dzięki nasłuchiwaczu **onMsgFromMainToFragment**.
5. Po otrzymaniu wiadomości przekazanej przez obiekt FragmentBlue, możliwa jest prezentacja szczegółowych informacji związanych z wyborem dokonanym przez użytkownika.

Przykład 1 – Wywołania zwrotne

```
// method(s) to pass messages from fragments to MainActivity

public interface MainCallbacks {

    public void onMsgFromFragToMain (String sender, String strValue);

}
```

```
// method(s) to pass messages from MainActivity to Fragments

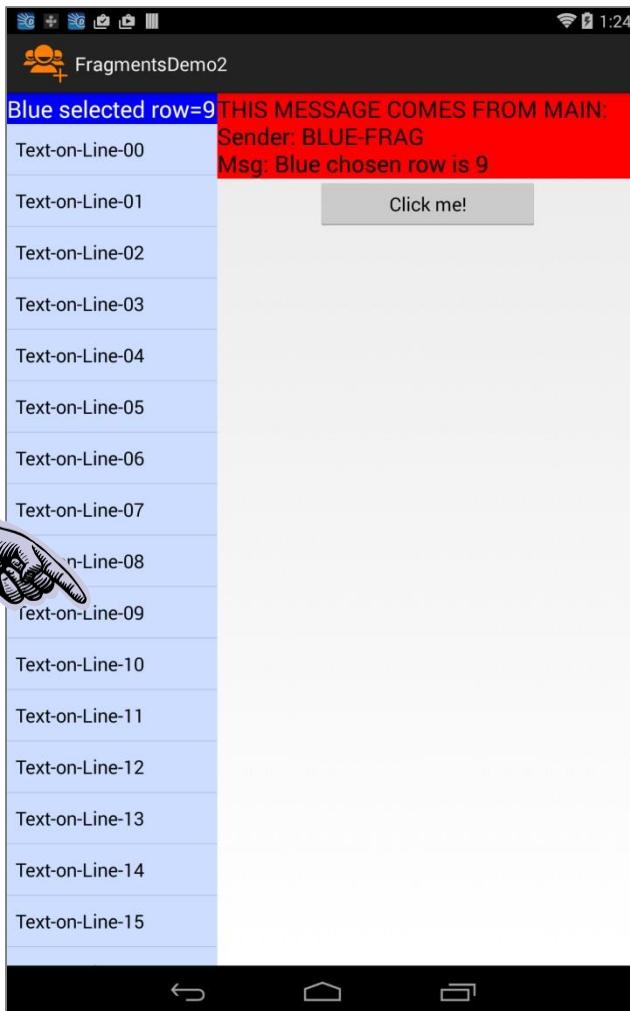
public interface FragmentCallbacks {

    public void onMsgFromMainToFragment(String strValue);

}
```

Przedstawione interfejsy implementują formalny wymóg do stworzenia **komunikacji wewnętrz procesów** między fragmentami i aktywnością bazową.

Przykład 2 – Statyczne wiązanie



Przykład jest analogiczny do poprzedniego. W skład jego struktury wchodzą 3 klasy:

- **MainActivity** (bazowa),
- **FragmentRed** (master),
- **FragmentBlue** (detail)

Główna różnica między przykładem pierwszym i drugim polega na sposobie tworzenia interfejsu użytkownika – tym razem fragmenty powiązane są w sposób statyczny.

Na następnych stronach przedstawiono nowy układ *activity_main.xml* oraz klasę głównej aktywności **MainActivity**. Pozostałe moduły pozostają bez zmian.

Przykład 2 – Statyczne wiązanie

- Statyczne wiązanie jest prostsze i wymaga mniej programowania niż w przypadku dynamicznego wiązania.
- To podejście rekomendowane jest dla aplikacji, w ramach których fragmenty nie ulegają zmianie.
- Statycznie dołączone fragmenty nie mogą zostać usunięte (jednakże inne fragmenty mogą zostać dodane do interfejsu).
- Układ XML aktywności wykorzystuje znacznik **<fragment>** by zaznaczyć pozycję oraz wielkość obszaru w ramach którego fragment może zostać osadzony.
- Fragmenty mogą być identyfikowane poprzez nadanie im unikalnej nazwy korzystając z przedstawionych atrybutów: (jeżeli żadna nazwa nie zostanie fragmentowi nadana, rolę identyfikatora pełni ID kontenera w ramach którego fragment jest osadzony)
 1. **android:name="AppPackageName.FragmentClassName"**
 2. **android:id="@+id/uniqueName"**
 3. **android:tag="tag"**

Przykład 2 – Statyczne wiązanie

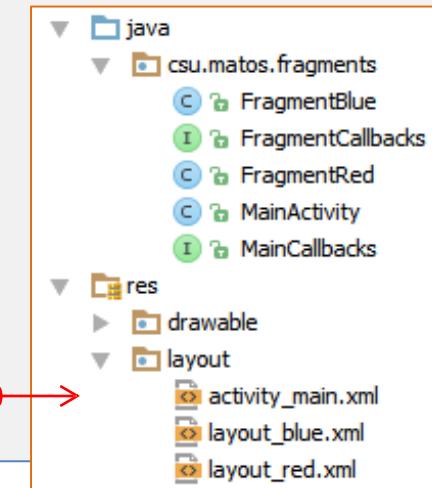
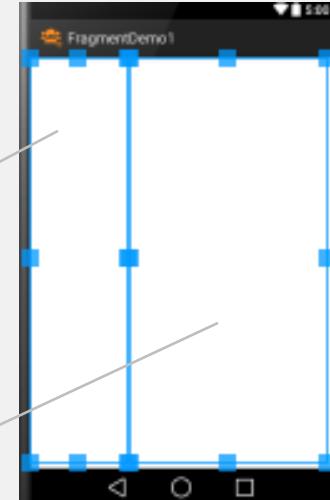
Układ XML: activity_main.xml

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

    <fragment
        android:id="@+id/main_holder_blue"
        android:name="csu.matos.fragments.FragmentBlue"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1" />

    <fragment
        android:id="@+id/main_holder_red"
        android:name="csu.matos.fragments.FragmentRed"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="2" />

</LinearLayout>
```



Przykład 2 – Klasa MainActivity

1 z 3

```
public class MainActivity extends Activity implements MainCallbacks {  
    FragmentRed redFragment;  
    FragmentBlue blueFragment ;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        // NOTHING to do, fragments will be automatically created and added to the GUI  
    }  
  
    @Override  
    public void onAttachFragment(Fragment fragment) {  
        super.onAttachFragment(fragment);  
        // get a reference to each fragment attached to the GUI  
        if (fragment.getClass() == FragmentRed.class ){  
            redFragment = (FragmentRed) fragment;  
  
        }  
        if (fragment.getClass() == FragmentBlue.class ){  
            blueFragment = (FragmentBlue) fragment;  
  
        }  
    }  
}
```



Przykład 2 – Klasa MainActivity

2 z 3

```
@Override
public void onMsgFromFragToMain(String sender, String strValue) {
    Toast.makeText(getApplicationContext(), " MAIN GOT MSG >> " + sender
        + "\n" + strValue, Toast.LENGTH_LONG).show();

    if (sender.equals("RED-FRAG")){
        //TODO: do here something smart on behalf of BLUE fragment
    }

    if (sender.equals("BLUE-FRAG")) {
        redFragment.onMsgFromActivity("\nSender: " + sender + "\nMsg: " + strValue);

    }
}

}//onMsgFromFragToMain

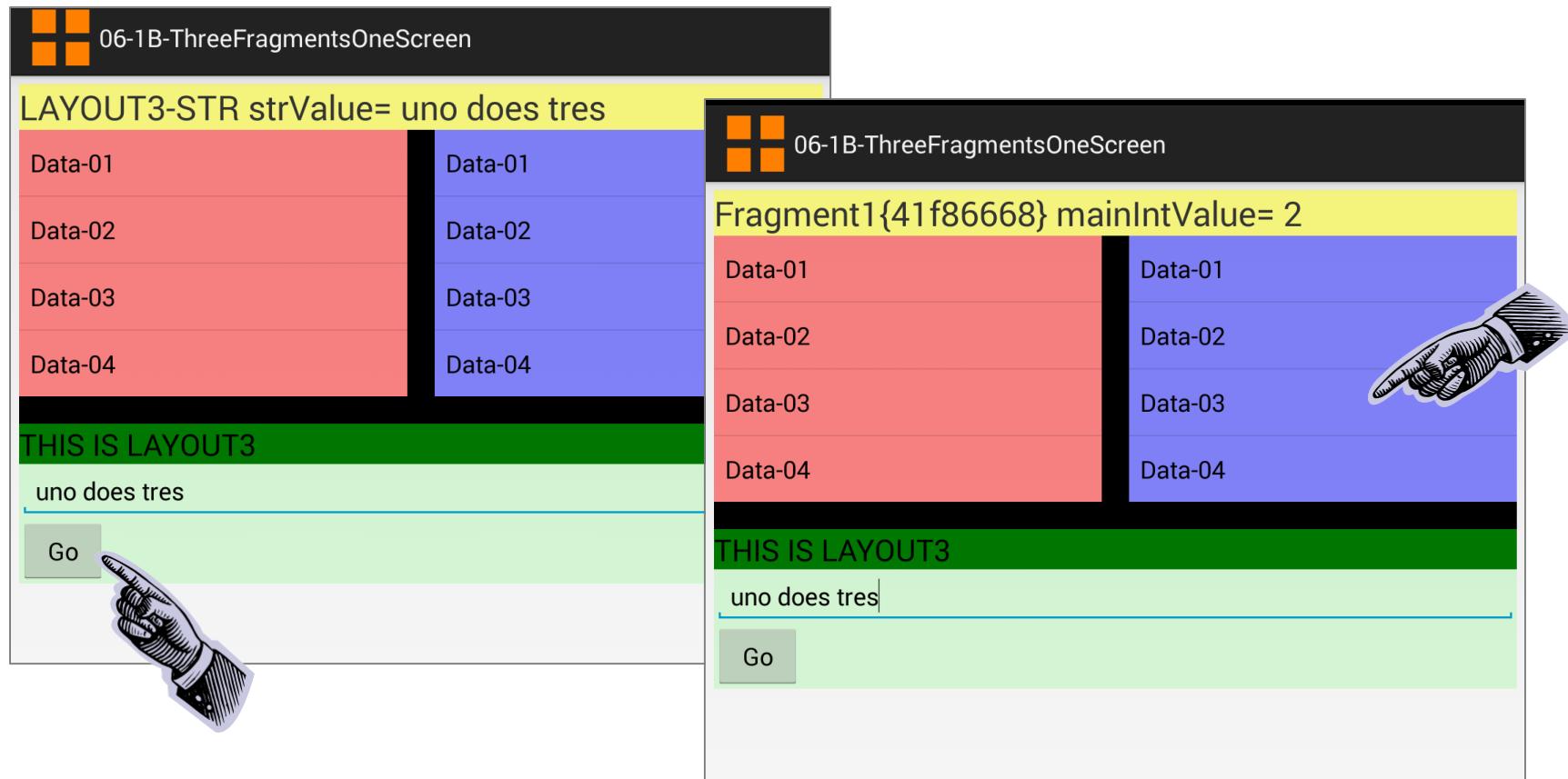
}
```

Komentarze

1. W tym przykładzie metoda **onCreate** nie jest wykorzystywana. Ponadto metoda **onCreateView** nie jest nawet wywoływana. Klauzula `android:name="csu.matos.fragments.FragmentXYZ"` definiuje jaki fragment ma zostać dodany do aktywności.
2. Gdy fragment jest dołączany do ekranu, metoda **onAttachFragment** jest wywoływana. Pozwala to na zapamiętanie referencji do *czerwonego i niebieskiego* fragmentu.
3. Wiadomości wysłane przez blueFragment do głównej aktywności są wyłapywane w nasłuchiwaczu **onMsgFromFragToMain**. Jak w poprzednim przykładzie, wiadomości od blueFragment są przekazywane do redFragment.

Przykład 3 – Wiele niezależnych fragmentów

- Jest to niewielka modyfikacja Przykładu 1. Główna aktywność reprezentuje ekran pokazujący trzy niezależne fragmenty.
- Wszystkie fragmenty są **widoczne i aktywne**, zapewniając wiele możliwości interakcji z użytkownikiem. Dwie instancje prezentują listę elementów, natomiast dolny widok zawiera pole tekstowe i przycisk.



UKŁAD XML: activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/linearLayoutMain"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="6dp" >

    <TextView
        android:id="@+id/txtMsgMain"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#77ffff00"
        android:textSize="25sp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <LinearLayout
            android:id="@+id/home1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#77ff0000"
            android:orientation="vertical" />
    
```

Tylko główna aktywność jest prezentowana, ponieważ reszta kodu jest niemalże tożsama z przykładem 1.

XML LAYOUT: activity_main.xml

```
<View
    android:layout_width="20dp"
    android:layout_height="match_parent"
    android:background="#ff000000" />

<LinearLayout
    android:id="@+id/home2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:background="#770000ff"
    android:orientation="vertical" />
</LinearLayout>

<View
    android:layout_width="match_parent"
    android:layout_height="20dp"
    android:background="#ff000000" />

<LinearLayout
    android:id="@+id/home3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" />
</LinearLayout>
```

Przykład 3 Klasa MainActivity - fragment

```
public class MainActivity extends Activity implements MainCallbacks {  
  
    FragmentTransaction ft;  
    FragmentRed redFragment;  
    FragmentBlue blueFragment;  
    FragmentBlue blueFragment2  
;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        // create a new BLUE fragment - show it  
        ft = getFragmentManager().beginTransaction();  
        blueFragment = FragmentBlue.newInstance("blue");  
        ft.replace(R.id.home1, blueFragment); ft.commit();  
  
        // create a new BLUE fragment - show it  
        ft = getFragmentManager().beginTransaction();  
        blueFragment2 = FragmentBlue.newInstance("red");  
        ft.replace(R.id.home2, blueFragment2); ft.commit();  
  
        // create a new RED fragment - show it  
        ft = getFragmentManager().beginTransaction();  
        redFragment = FragmentRed.newInstance("green");  
        ft.replace(R.id.home3, redFragment); ft.commit();  
  
    }  

```



Przykład 4 – Zachowywanie stanu fragmentów

Dobrą praktyką jest redefinicja metody **onSaveInstanceState** by przed opuszczeniem fragmentu zapisać jego stan w zarządzalnej systemowo kolekcji *outState*.

Później można sprawdzić czy został zapisany poprzedni stan komponentu oraz odtworzyć go w jednej z metod: **onCreate** , **onCreateView** , **onViewCreated**, lub **onViewStateRestored**. Jest to tzw. *goracy-start*.

W przypadku tzw. *zimnego-startu* referencja tej kolekcji ustawiona jest na **null**. W przeciwnieństwie do aktywności, fragmenty nie mają metody **onRestoreInstanceState**.

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    ...  
    stateData = getArguments().getString("arg1", "cold-start");  
  
    if (savedInstanceState != null){  
        stateData = savedInstanceState.getString("arg1", "warm-default");  
    }  
}//onCreate  
...  
@Override  
public void onSaveInstanceState(Bundle outState) {  
    ...  
    outState.putString("arg1", stateData);  
    super.onSaveInstanceState(outState);
```

Operacje na fragmentach

Istnieje wiele operacji, które wpływają na widoczność i przypisanie fragmentów do danej aktywności. Te operacje muszą być wykonane w ramach transakcji (patrz obiekt **FragmentTransaction**).

add()

Dodaje fragment do aktywności. Jeżeli aktywność jest restartowana, uprzednio dodane fragmenty [poprzez metodę add()] mogą zostać ponownie wykorzystane (nie ma potrzeby ponownego tworzenia obiektu reprezentującego fragment).

remove()

Usuwa fragment z aktywności. Fragment jest niszczony (chyba, że został dodany do tzw. BackStack).

replace()

Fragment jest niszczony i zostaje zastąpiony przez inną instancję fragmentu.

show() / hide()

Pokazuje uprzednio ukryty fragment (nie usunięty).

attach() / detach()

Dołącza uprzednio odłączony z aktywności fragment. Odłączone fragmenty nie są widoczne, ale nie zostają również od razu zniszczone.

Operacje na fragmentach

Przemyśl poniższy fragment kodu, który powoduje ukrycie i pokazanie fragmentów. Jaka jest między nimi różnica?

```
FragmentTransaction ft = getFragmentManager().beginTransaction();

redFragment = FragmentRed.newInstance(intValue);
ft.add(R.id.main_holder, redFragment, "RED-TAG");

ft.hide(redFragment);
ft.show(redFragment);

ft.detach(redFragment);
ft.attach(redFragment);

ft.commit();
```

Odtwarzanie stanu wykorzystując BackStack

System Android wprowadził specjalny stos by pomóc fragmentom w zapamiętaniu poprzedniego stanu, gdy użytkownik nawiguje po ekranach aplikacji.

Stos ten nazywa się **BackStack** i umożliwia wykonanie operacji push/pop podczas transakcji (**FragmentTransactions**). Jest to analogiczne zachowanie do stosu aktywności.

Należy pamiętać, że *wszystkie urządzenia mobilne z Androidem* posiadają przycisk **Wstecz**. Jego użycie powoduje cofnięcie do poprzedniego ekranu (aktywności).

Po co używać BackStack?

Umożliwia on ponowne wykorzystanie fragmentu (zamiast jego ponowne stworzenie od zera) a jego stan jest przywracany w sposób transparentny (bez potrzeby stosowania obiektów typu Bundle). Upraszczając to tworzenie aplikacji z wykorzystaniem fragmentów.

Odtwarzanie stanu wykorzystując BackStack

Typowa sekwencja by stworzyć fragment i dodać do BackStack wygląda tak:

```
FragmentTransaction ft = getFragmentManager().beginTransaction();
Fragment redFragment = FragmentRed.newInstance(intParameterValue);
ft.replace(R.id.main_holder, redFragment, "RED-FRAG");

ft.addToBackStack("RED_UI");
ft.commit();
```

W tym przykładzie, podczas transakcji (*ft*) dodawany jest obiekt *redFragment* do głównej aktywności. Fragment wykorzystuje opcjonalny alias “RED-FRAG” do łatwiejszej identyfikacji.

Zanim transakcja zostanie wykonana, instrukcja

```
ft.addToBackStack("RED_UI");
```

dodaje referencje transakcji do stosu BackStack pod aliasem “RED_UI”. Później możliwe jest przeszukanie BackStack pod kątem wartości aliasu. Gdy zostanie on znaleziony i wykonana zostanie operacja pop, aplikacja wraca do stanu po transakcji.

Odtwarzanie stanu wykorzystując BackStack

Nawigacja

Wydobywanie danych z BackStack odbywa się na wiele sposobów:

- Poprzez wciśnięcie przycisku **Wstecz** by cofnąć się do poprzednio prezentowanego interfejsu użytkownika.
- Poprzez wywołanie metody **.popBackStackImmediate(...)** by przywrócić określony stan aplikacji.

```
// Remove current fragment's UI and show its previous screen
try {
    FragmentTransaction ft = getFragmentManager().beginTransaction();

    android.app.FragmentManager fragmentManager = getFragmentManager();

    ①→ int bsCount = fragmentManager.getBackStackEntryCount();
    String tag = fragmentManager.getBackStackEntryAt(bsCount-1).getName();
    ②→ int id = fragmentManager.getBackStackEntryAt(bsCount-1).getId();
    Log.e("PREVIOUS Fragment: ", "" + tag + " " + id);

    ③→ fragmentManager.popBackStackImmediate(id, 1); //supply: id or tag

    ft.commit();

} catch (Exception e) {
    Log.e("REMOVE>>> ", e.getMessage() );
}
```

Odtwarzanie stanu wykorzystując BackStack

Nawigacja

W poprzednim przykładzie symulowane jest zachowanie kliknięcia przycisku wstecz – powrót do poprzedniego stanu aplikacji:

1. Rozmiar stosu BackStack jest wyznaczany (`getBackStackEntryCount`)
2. Pobierany jest pierwszy element stosu. Pobierany jest jego alias (tag) oraz unikalny identyfikator numeryczny:
`fragmentManager.getBackStackEntryAt(bsCount-1).getId()`.
3. Metoda `.popBackStack(id, 1)` usuwa elementy ze stosu BackStack poczynając od pierwszego, dopóki znajdzie wpis z pasującym **id**. Widok aplikacji jest aktualizowany, pokazując stan zgodny z transakcją zapisaną na stosie BackStack.

Odtwarzanie stanu wykorzystując BackStack

Nawigacja

Poniższy kod czyści BackStack. Wszystkie wpisy dodane poprzez wywołanie metody `ft.addToBackStack(...)` są usuwane. Widok aplikacji jest aktualizowany, poprzez usunięcie wszystkich fragmentów, których referencja znajduje się w stosie BackStack.

```
try {
    FragmentTransaction ft = getFragmentManager().beginTransaction();

    android.app.FragmentManager fragmentManager = getFragmentManager();

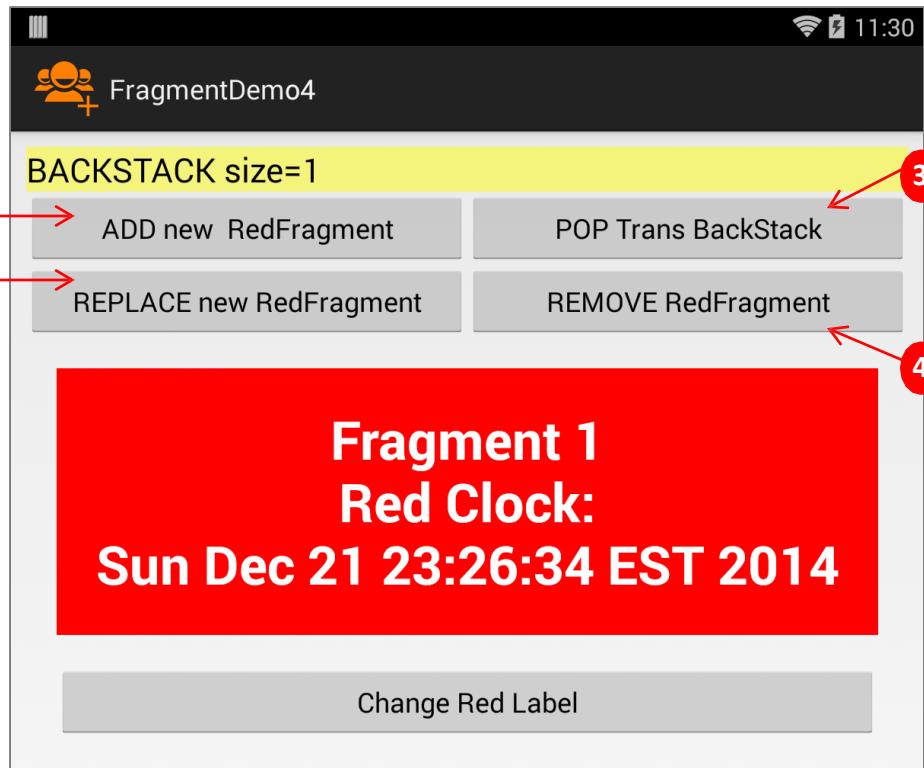
    fragmentManager.popBackStackImmediate(null,
                                         FragmentManager.POP_BACK_STACK_INCLUSIVE);

    ft.commit();

} catch (Exception e) {
    Log.e("CLEAR-STACK>>> ", e.getMessage() );
}
```

Przykład 5 – Wykorzystanie BackStack

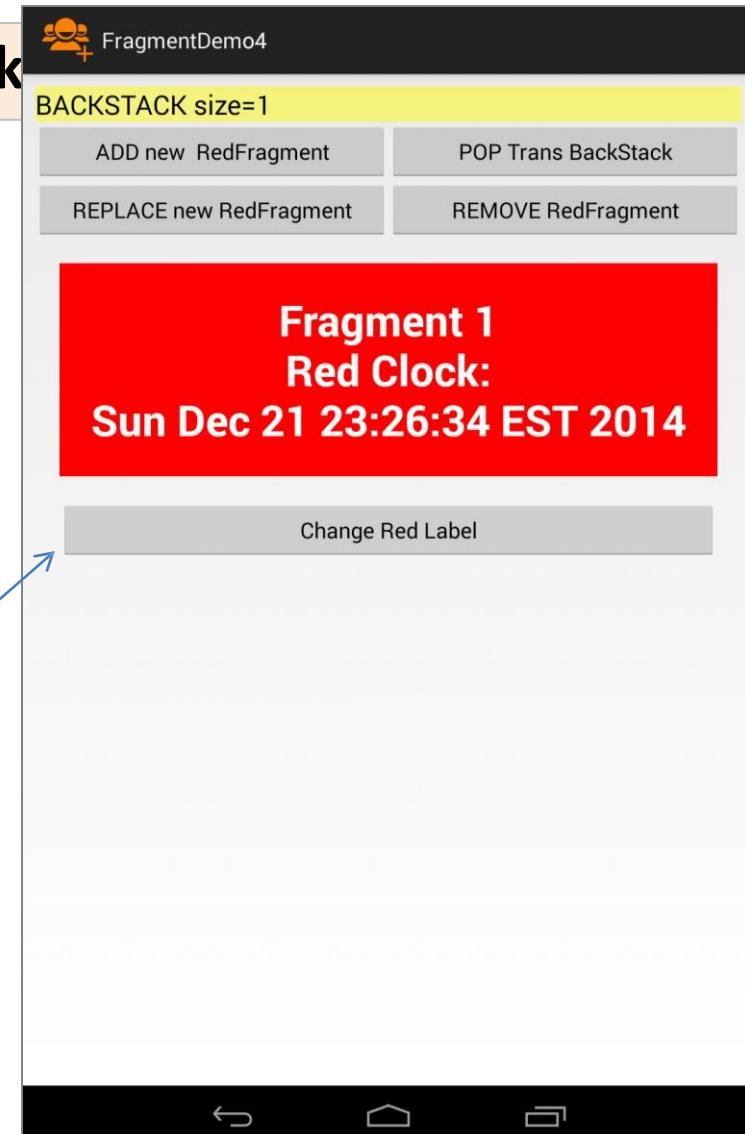
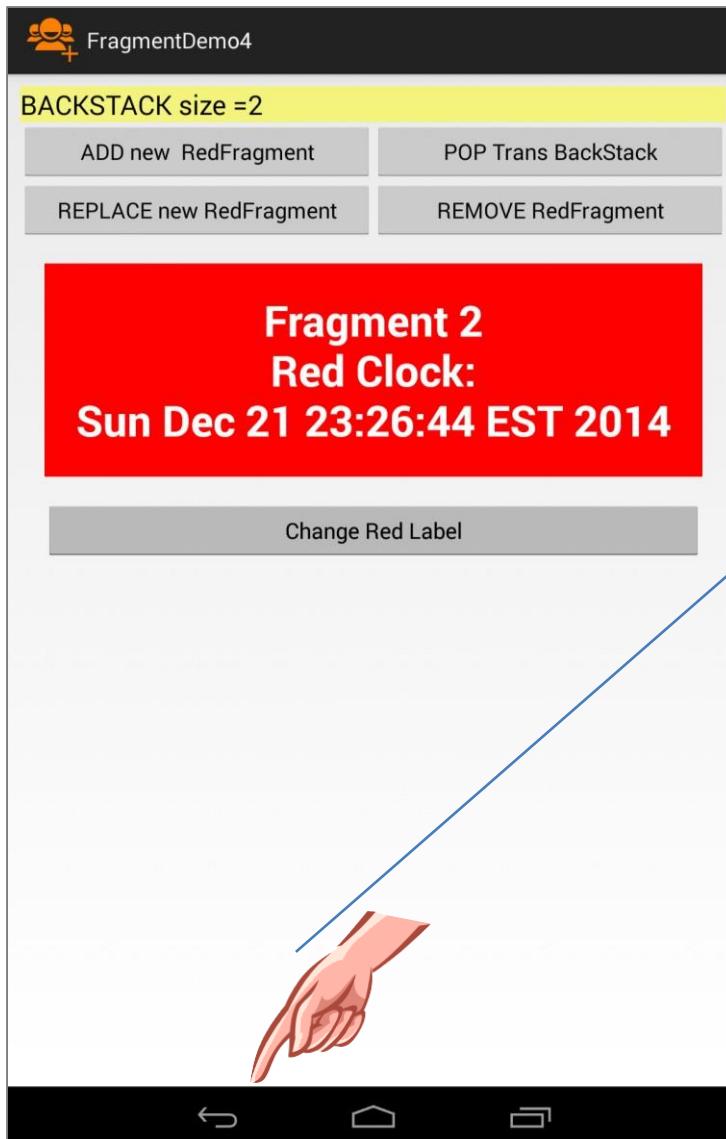
1 of x



1. Nowy fragment *redFragment* jest tworzony oraz dodawany do aktywności poprzez metodę **add()**. Informacje o tej transakcji trafiają do BackStack.
2. Tak jak poprzednia opcja, ale fragment dodawany jest do aktywności z użyciem metody **replace()** (stary widok jest niszczyony). Transakcja jest dodawana do BackStack.
3. Wydobycie elementu z BackStack skutkuje zmianą widoku aplikacji i przywróceniem poprzedniego stanu fragmentu. Wielkość BackStack jest zmniejszana o jeden.
4. Przycisk “Remove” aktywuje wyszukiwanie korzystając z metody *findFragmentByTag*. Wyszukiwanie to uwzględnia w pierwszej kolejności te fragmenty, które są aktualnie dodane do aktywności; jeżeli nie ma takich fragmentów, cały stos jest przeszukiwany. Użytkownikowi prezentowany jest poprzedni stan aplikacji.

Przykład 5 – Wykorzystanie BackStack

1. Tworzony jest czerwony fragment, a informacja o tej transakcji zapisywana jest w BackStack.



2. Poprzez kliknięcie na przycisk **Wstecz** aplikacja cofa się do poprzedniego stanu.

Przykład 5. Układ: activity_main.xml

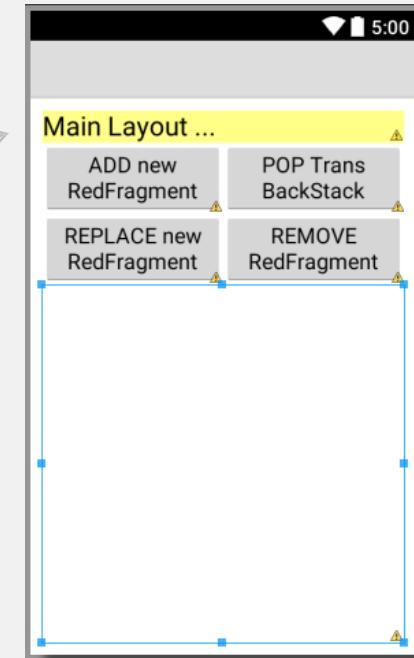
1 z 3

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:baselineAligned="false"
    android:orientation="vertical"
    android:padding="10dp" >

    <TextView
        android:id="@+id/textView1Main"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#77ffff00"
        android:text="Main Layout ..."
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:baselineAligned="false"
        android:orientation="horizontal" >

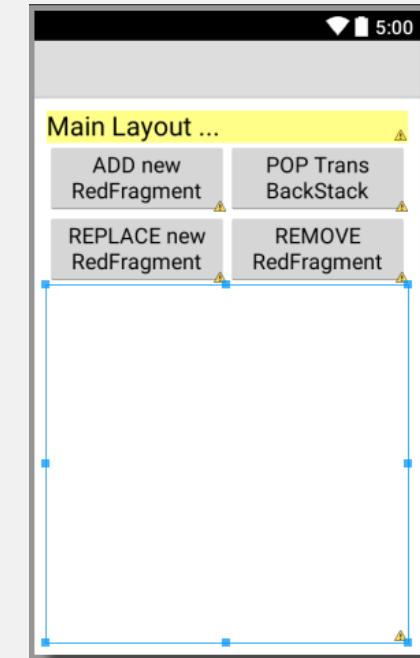
        <Button
            android:id="@+id/button1MainShowRed"
            android:layout_width="150dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="ADD new RedFragment" />
    
```



Przykład 5. Układ: activity_main.xml

2 z 3

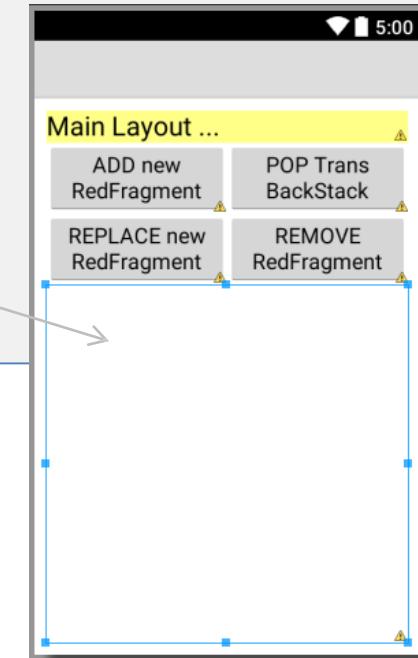
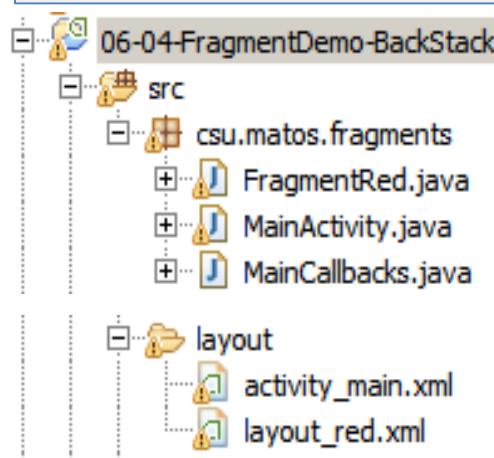
```
<Button  
    android:id="@+id/button2MainPop"  
    android:layout_width="150dp"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="POP Trans BackStack" />  
  
</LinearLayout>  
  
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:baselineAligned="false"  
    android:orientation="horizontal" >  
  
    <Button  
        android:id="@+id/button4MainReplace"  
        android:layout_width="150dp"  
        android:layout_height="wrap_content"  
        android:layout_weight="1"  
        android:text="REPLACE new RedFragment" />  
  
    <Button  
        android:id="@+id/button3MainRemove"  
        android:layout_width="150dp"  
        android:layout_height="wrap_content"  
        android:layout_weight="1"  
        android:text="REMOVE RedFragment" />  
  
</LinearLayout>
```



Przykład 5. Układ: activity_main.xml

3 z 3

```
<FrameLayout  
    android:id="@+id/main_holder"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_weight="2"  
    android:orientation="vertical" />  
  
</LinearLayout>
```



Wywołania zwrotne

```
package csu.matos.fragments;  
// method(s) to pass messages from fragments to MainActivity  
  
public interface MainCallbacks {  
    public void onMsgFromFragToMain ( String sender, String  
        strValue);  
}
```

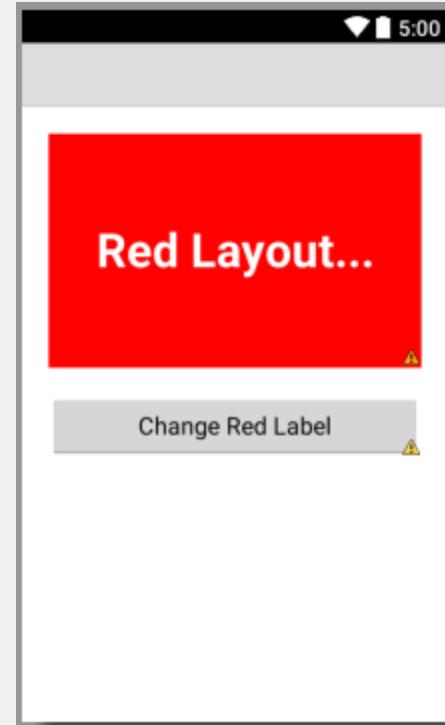
Przykład 5. Układ: layout_red.xml

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

    <TextView
        android:id="@+id/textView1Red"
        android:layout_width="match_parent"
        android:layout_height="175dp"
        android:layout_margin="20dp"
        android:background="#ffff0000"
        android:gravity="center"
        android:text="Red Layout..."
        android:textColor="@android:color/white"
        android:textSize="35sp"
        android:textStyle="bold" />

    <Button
        android:id="@+id/button1Red"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:text="Change Red Label" />

</LinearLayout>
```



Przykład 5. Klasa MainActivity

1 z 4

```
public class MainActivity extends Activity implements MainCallbacks, OnClickListener {  
  
    FragmentTransaction ft;  
    FragmentRed redFragment;  
    TextView txtMsg;  
    Button btnAddRedFragment;  
    Button btnReplaceRedFragment;  
    Button btnPop;  
    Button btnRemove;  
    int serialCounter = 0; //used to enumerate fragments  
    String redMessage;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        txtMsg = (TextView) findViewById(R.id.textView1Main);  
        btnAddRedFragment = (Button) findViewById(R.id.button1MainShowRed);  
        btnReplaceRedFragment = (Button) findViewById(R.id.button4MainReplace);  
  
        btnPop = (Button) findViewById(R.id.button2MainPop);  
        btnRemove = (Button) findViewById(R.id.button3MainRemove);  
        btnAddRedFragment.setOnClickListener(this);  
        btnReplaceRedFragment.setOnClickListener(this);  
        btnPop.setOnClickListener(this);  
        btnRemove.setOnClickListener(this);  
    }  
}
```

Przykład 5. Klasa MainActivity

2 z 4

```
// CallBack (receiving messages coming from Fragments)
@Override
public void onMsgFromFragToMain(String sender, String strValue) {
    // show message arriving to MainActivity
    txtMsg.setText( sender + ">" + strValue );
}

// -----
public void onClick(View v) {

    if(v.getId() == btnAddRedFragment.getId() ){
        addRedFragment(++serialCounter);
    }

    if(v.getId() == btnReplaceRedFragment.getId() ){
        replaceRedFragment(++serialCounter);
    }

    if(v.getId() == btnPop.getId() ){
        FragmentManager fragmentManager = getFragmentManager();
        int counter = fragmentManager.getBackStackEntryCount();
        txtMsg.setText("BACKSTACK old size=" + counter);

        if(counter>0) {
            // VERSION 1 [ popBackStack could be used as opposite of addBackStack() ]
            // pop takes a Transaction from the BackStack and a view is also deleted
            fragmentManager.popBackStackImmediate();
            txtMsg.append("\nBACKSTACK new size=" + fragmentManager.getBackStackEntryCount() );
        }
    } //Pop
}
```

Przykład 5. Klasa MainActivity

3 z 4

```
if(v.getId() == btnRemove.getId() ){
    FragmentManager fragmentManager = getFragmentManager();
    int counter = fragmentManager.getBackStackEntryCount();
    txtMsg.setText("BACKSTACK old size=" + counter);

    // VERSION 2 -----
    // Removes an existing fragment from the fragmentTransaction.
    // If it was added to a container, its view is also removed from that
    // container. The BackStack may remain the same!
    Fragment f1 = fragmentManager.findFragmentByTag("RED-TAG");
    fragmentManager.beginTransaction().remove(f1).commit();
    txtMsg.append("\nBACKSTACK new size=" + fragmentManager.getBackStackEntryCount() );

    // VERSION 3 -----
    // Fragment f1 = fragmentManager.findFragmentById(R.id.main_holder);
    // fragmentManager.beginTransaction().remove(f1).commit();
    // txtMsg.append("\nBACKSTACK new size=" + fragmentManager.getBackStackEntryCount()
    // );

} // Remove

} // onClick

@Override
public void onBackPressed() {
    super.onBackPressed();
    int counter = getFragmentManager().getBackStackEntryCount();
} txtMsg.setText("BACKSTACK size=" + counter);
```

Przykład 5. Klasa MainActivity

4 z 4

```
public void addRedFragment(int intValue) {  
    // create a new RED fragment, add fragment to the transaction  
    FragmentTransaction ft = getFragmentManager().beginTransaction();  
    redFragment = FragmentRed.newInstance(intValue);  
    ft.add(R.id.main_holder, redFragment, "RED-TAG");  
    ft.addToBackStack("MYSTACK1");  
    ft.commit();  
  
    // complete any pending insertions in the BackStack, then report its size  
    getFragmentManager().executePendingTransactions();  
    txtMsg.setText("BACKSTACK size = " + getFragmentManager().getBackStackEntryCount() );  
}  
  
public void replaceRedFragment(int intValue) {  
    // create a new RED fragment, replace fragments in the transaction  
    FragmentTransaction ft = getFragmentManager().beginTransaction();  
    redFragment = FragmentRed.newInstance(intValue);  
    ft.replace(R.id.main_holder, redFragment, "RED-TAG");  
    ft.addToBackStack("MYSTACK1");  
    ft.commit();  
  
    // complete any pending insertions in the BackStack, then report its size  
    getFragmentManager().executePendingTransactions();  
    txtMsg.setText("BACKSTACK size = " + getFragmentManager().getBackStackEntryCount() );  
}
```

Przykład 5. Fragment FragmentRed

1 z 2

```
public class FragmentRed extends Fragment {  
    MainActivity main;  
    TextView txtRed;  
    Button btnRedClock;  
    int fragmentId;  
    String selectedRedText = "";  
  
    public static FragmentRed newInstance(int fragmentId) {  
        FragmentRed fragment = new FragmentRed();  
        Bundle bundle = new Bundle();  
        bundle.putInt("fragmentId", fragmentId);  
        fragment.setArguments(bundle);  
        return fragment;  
    } // newInstance  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        // Activities containing this fragment must implement MainCallbacks  
        if (!(getActivity() instanceof MainCallbacks)) {  
            throw new IllegalStateException(  
                ">>> Activity must implement MainCallbacks");  
        }  
        main = (MainActivity) getActivity();  
        fragmentId = getArguments().getInt("fragmentId", -1);  
    }  
}
```

Przykład 5. Fragment FragmentRed

2 z 2

```
@Override
public View onCreateView( LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    LinearLayout view_layout_red = (LinearLayout) inflater.inflate(
        R.layout.Layout_red, null);

    txtRed = (TextView) view_layout_red.findViewById(R.id.textView1Red);
    txtRed.setText( "Fragment " + fragmentId );

    btnRedClock = (Button) view_layout_red.findViewById(R.id.button1Red);
    btnRedClock.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {
            selectedRedText = "\nRed Clock:\n" + new Date().toString();
            txtRed.append(selectedRedText);
            // main.onMsgFromFragToMain("RED-FRAG", selectedRedText );
        }
    });
    return view_layout_red;
}

}// FragmentRed
```



Fragmenty

Pytania ?

Programowanie urządzeń mobilnych

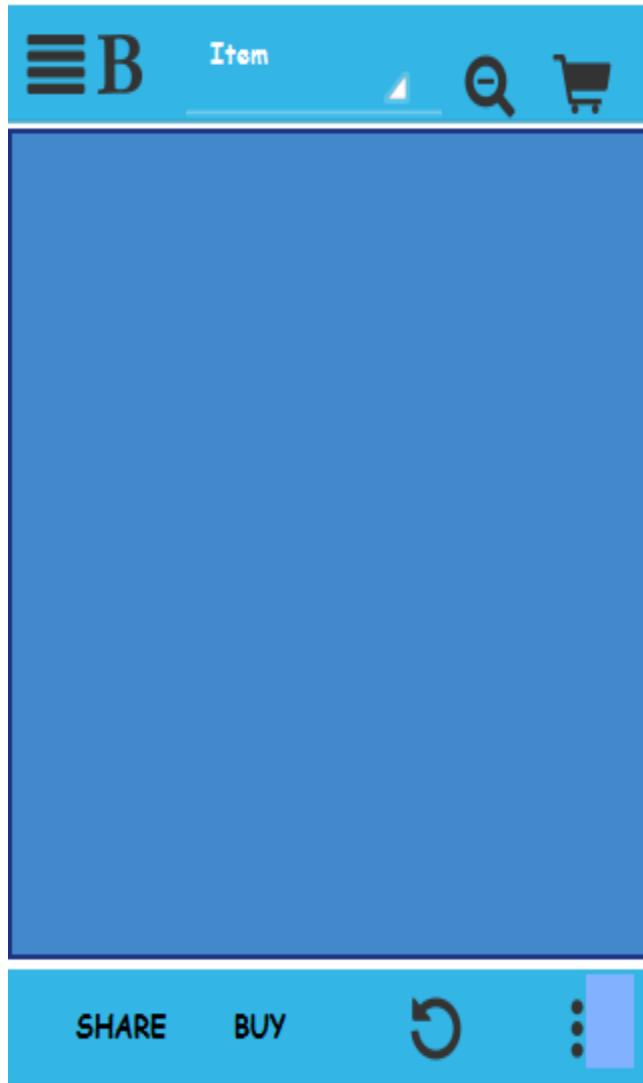
Wykorzystanie menu i ActionBar

Tłumaczenie i adaptacja materiałów: dr Tomasz Xięski.
Na podstawie prezentacji udostępnionych przez Victor Matos, Cleveland State University.

Portions of this page are reproduced from work created and [shared by Google and](#) used according to terms

Tworzenie GUI

Wykorzystanie ActionBar



Nagłówek

ActionBar zajmuje miejsce na górze ekranu. Wyjątkiem jest sytuacja, w której brakuje miejsca do umieszczenia wszystkich elementów. Wówczas zawartość rozdzielana jest na dwa paski: jeden u góry, jeden na dole.

Główna część aplikacji

Stopka:

Paski narzędziowe również mogą zostać umieszczone na dole ekranu.

Tworzenie GUI

Wykorzystanie menu

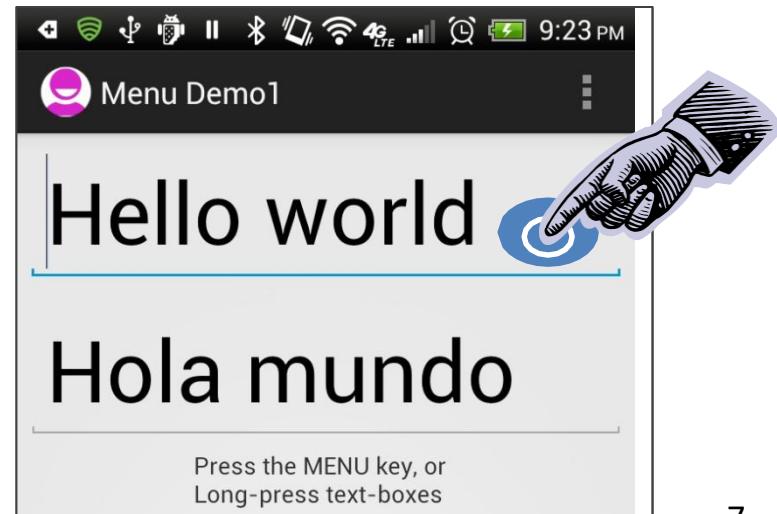
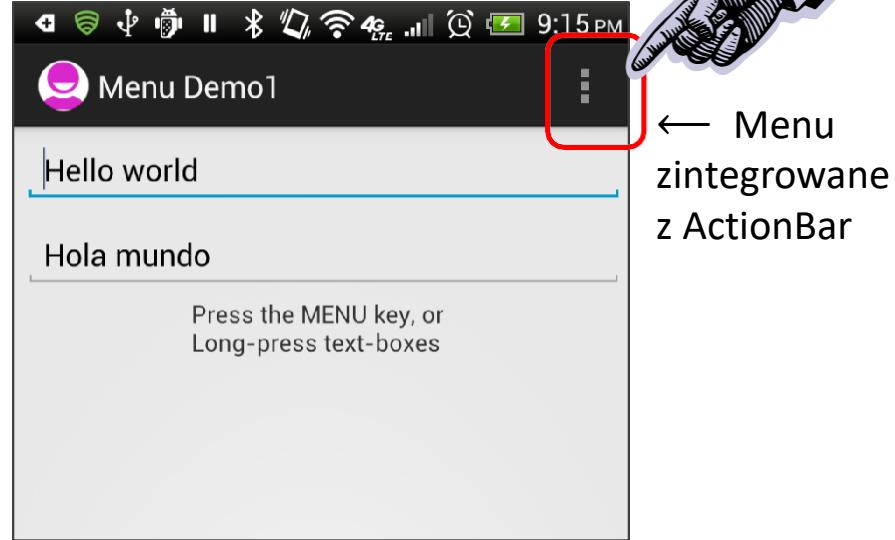
- Menu jest elementem często wykorzystywany podczas tworzenia interfejsu użytkownika w Androidzie.
- Dobre menu zapewnia prosty i jednolity interfejs dodający więcej możliwości do aplikacji, bez zajmowania znacząco większej przestrzeni roboczej.
- Menus może być ściśle powiązane z aktualnie prezentowanym widokiem. Każdy ekran może mieć swój własny zestaw opcji.
- Menu może być powiązane z dowolną liczbą widżetów umieszczonych na danej aktywności.
- **Aktualne rekomendacje w sprawie interfejsu użytkownika łączą wykorzystanie menu wraz z paskiem w nagłówku (ActionBar).**

Wykorzystanie menu

Typy Menu

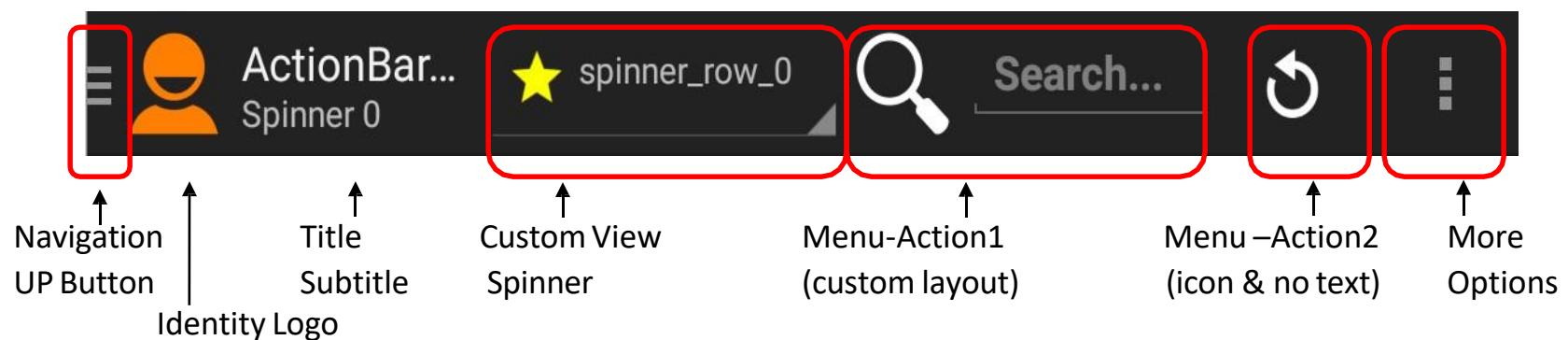
Android wspiera dwa typy menu:
Menu główne oraz **Menu kontekstowe**.

1. Globalne **menu główne** (*options menu*) jest uaktywniane poprzez przyciśnięcie fizycznego bądź wirtualnego przycisku **Menu**. Dla każdego ekranu dostępne jest tylko jedno menu główne.
2. **Menu kontekstowe** (*context menu*) uaktywniane jest poprzez dłuższe przytrzymanie palca na widżecie powiązanym z danym menu. Każdy widżet może mieć swoje menu kontekstowe.



Wykorzystanie ActionBar

- Komponent **ActionBar** został wprowadzony wraz z SDK 3.0 i jest obecny w wielu aplikacjach mobilnych. Jest przedstawiony jako pasek narzędziowy znajdujący się na górze każdego ekranu i zwykle posiada stałe miejsce dla całej aplikacji.
- Zwykle składa się z następujących elementów:
 - Nawigacji – Przycisk wyżej (symbolizowany jako ikona strzałki)
 - Logo,
 - Tytułu i podtytułu,
 - Opcjonalnych widżetów użytkownika,
 - Kafelek (przycisków posiadających tekst i/lub grafikę),
 - Przycisku menu głównego
 - Zakładek (status *deprecated od SDK4.4*)



Wykorzystanie ActionBar

ActionBar jest bardzo istotnym elementem ponieważ:

- Działa jako **skondensowany panel przełączający** pozwalający uzyskać łatwy dostęp do głównych funkcjonalności aplikacji.
- Zawiera przycisk wywołujący menu główne ‘:’, którego funkcjonalność może się dostosować do każdego ekranu.
- Zwykle pojawia się wyłącznie na górze ekranu, budując wrażenie **spójnego wyglądu** gdzie "*wszystkie najważniejsze przyciski znajdują się w jednym miejscu*".
- Może być wykorzystywany do implementacji **różnych typów nawigacji**:
 - (a) jako główny przycisk wyświetlający dodatkową listę opcji nad aktualnie wyświetlonym widokiem,
 - (b) jako zestaw zakładek do najważniejszych części aplikacji,
 - jako **przycisk wyżej**, który może być wykorzystany do cofania się o jedną pozycję w hierarchii stosu aktywności bądź do dowolnego innego miejsca hierarchii.

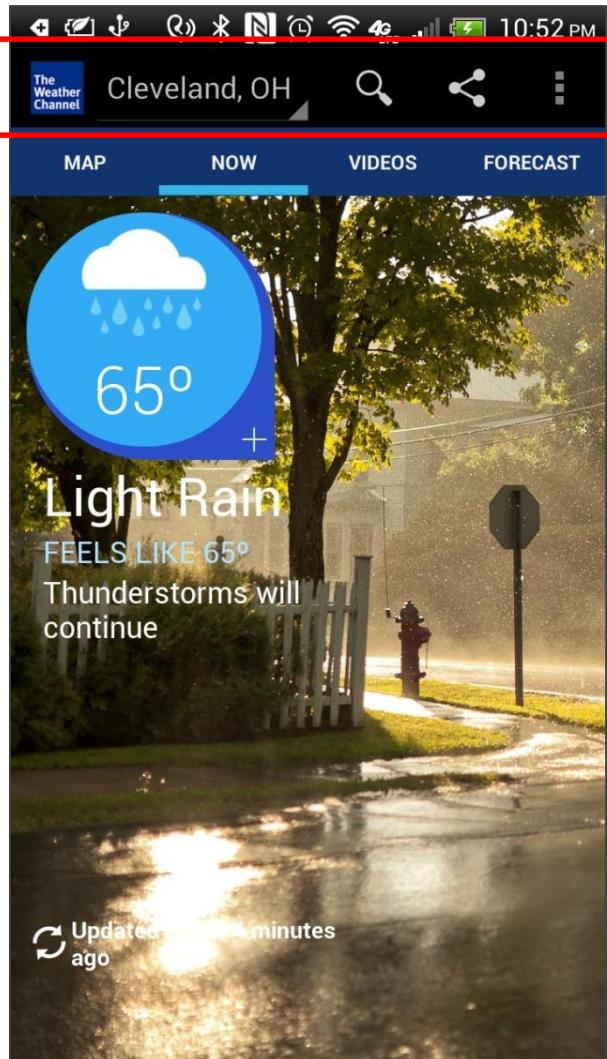
Przykład aplikacji wykorzystujących ActionBar

	Nazwa aplikacji	Liczba ściągnięć	Ocena	Kategoria
1	Google Search	1B	4.4	Tools
2	Gmail	1B	4.3	Communication
3	Google Maps	1B	4.3	Travel & Local
4	YouTube	1B	4.1	Media & Video
5	Facebook	1B	4.0	Social
6	WhatsApp	500M	4.4	Communications
7	Instagram	100M	4.5	Social
8	Pandora	100M	4.4	Music & Audio
9	Netflix	100M	4.4	Entertainment
10	Adobe Reader	100M	4.3	Productivity
11	Skype	100M	4.1	Communications
12	Twitter	100M	4.1	Social
13	eBay	50M	4.3	Shopping
14	Weather Channel	50M	4.2	Weather
15	Kindle	50M	4.1	Books & References
16	Wikipedia	10M	4.4	Books & References
17	Zillow	10M	4.4	Lifestyle
18	ESPN SportCenter	10M	4.2	Sports
19	BBC News	10M	4.2	News & Magazines
20	Amazon (Tablets)	10M	4.0	Shopping
21	Expedia	10M	4.0	Travel & Local

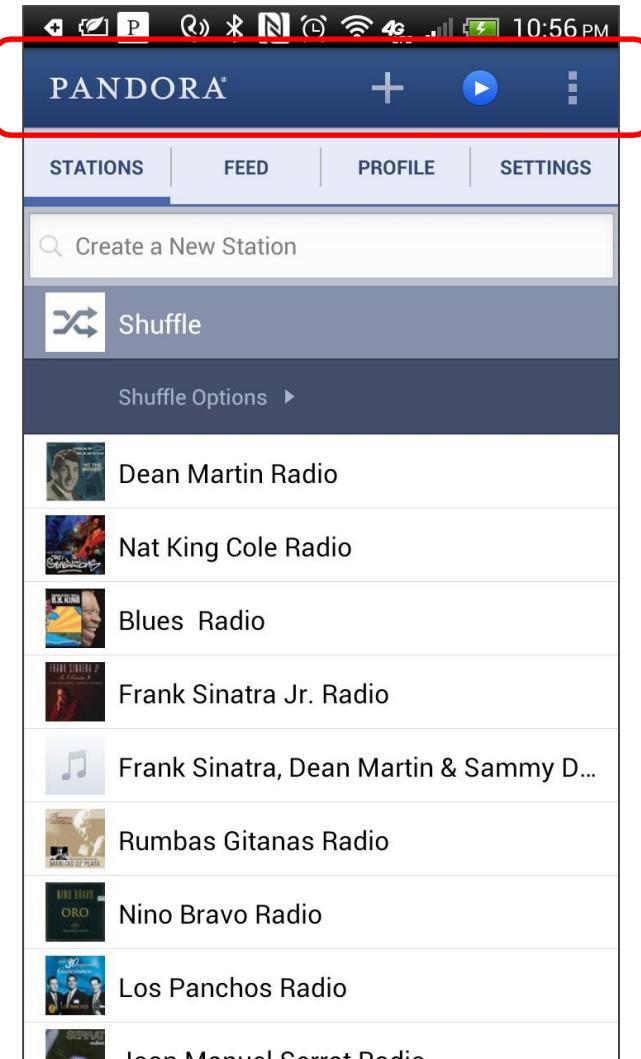
Źródło:

Sklep play. Dostęp: 16.02.2015. Link: <https://play.google.com/store?hl=en>

Przykład aplikacji wykorzystujących ActionBar



ActionBar



Dwie zupełnie różne
aplikacje
reprezentujące dość
podobny widok i
sposób nawigacji.

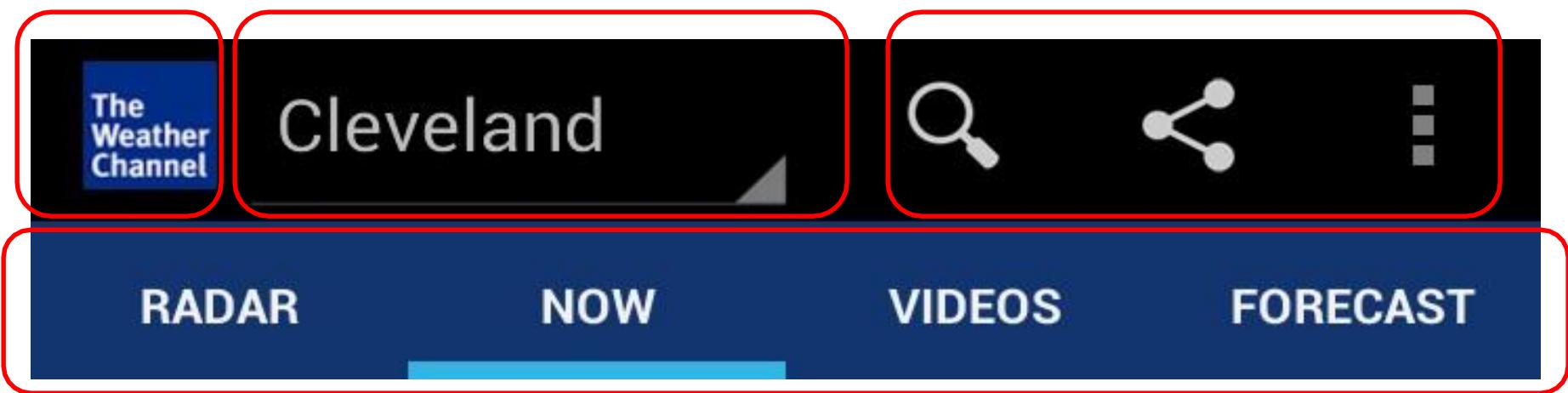
Identyfikacja elementów składowych ActionBar'a

Przykładowa interpretacja użytych komponentów dla aplikacji pogodowej:

Logo

ListView

Menu (Znajdź i Udostępnij) oraz pozostałe



PageViewer

Aktualnie wybrana zakładka

Kontrolka PageViewer stanowi alternatywę dla bezpośredniego użycia ActionBar do grupowania zakładek.

Architectura ActionBar

Klikalne *kafelki* udostępniane przez ActionBar zwykle są definiowane w pliku XML rezydującym w katalogu **res/menu**. Na jego podstawie tworzony jest obiekt odpowiedzialny za wyświetlenie menu głównego.

Standardowo każda dostępna opcja menu jest wizualizowana jako lista tekstowa aktywowana przez kliknięcie wirtualnego przycisku :. Jednakże wybrane pozycje mogą być wyświetlane osobno jako przycisku (z tekstem i/lub grafiką) będące częścią komponentu ActionBar.

Lista opcji menu jest zazwyczaj stała przez cały cykl życia aplikacji, jednakże może być w sposób dynamiczny wyłączana czy zmieniana.

Dwie metody, których zadaniem jest manipulowanie zestawem opcji dostępnych w ramach komponentu ActionBar to:

onCreateOptionsMenu(...)

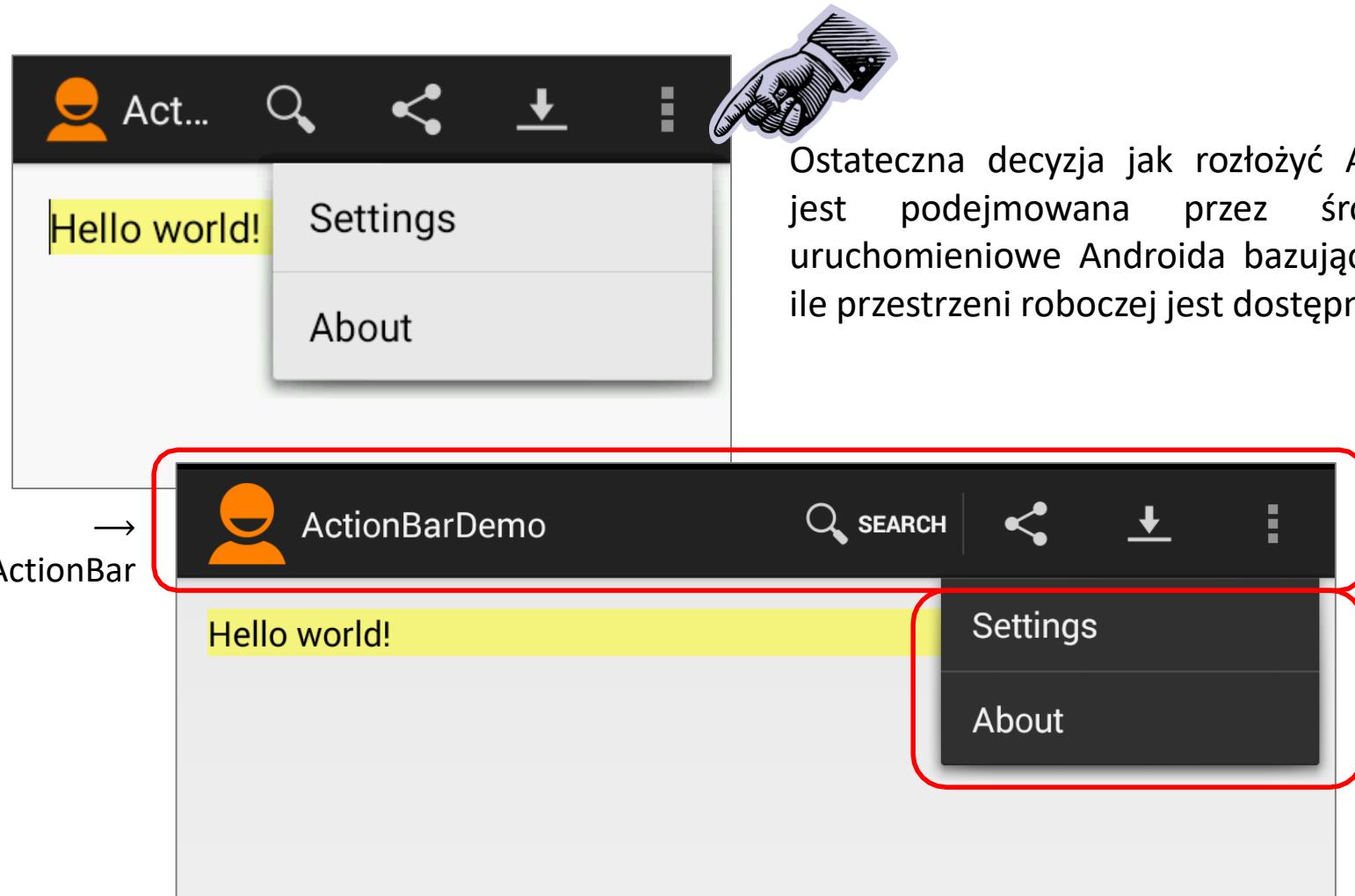
Tworzy menu na podstawie specyfikacji XML

onOptionsItemSelected(...)

Reaguje na zdarzenie kliknięcia poszczególnej opcji

Przykład 1 – Proste wykorzystanie ActionBar

Aplikacja bazuje na szablonie typu Basic Activity. Należy zmodyfikować plik **res/menu/main.xml** by wygenerować odpowiednie opcje menu. Zdjęcia pochodzą z tabletu oraz telefonu komórkowego.



Przykład 1 – Proste wykorzystanie ActionBar

UKŁAD res/menu/main.xml.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="csu.matos.MainActivity" >

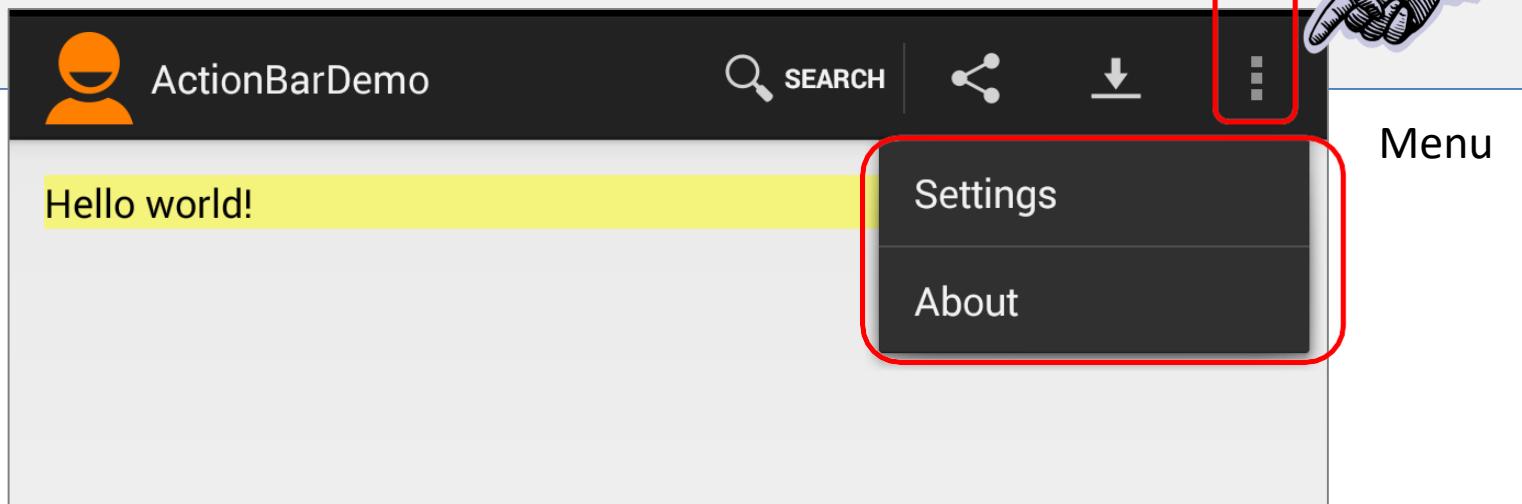
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always/withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
```



Przykład 1 – Proste wykorzystanie ActionBar

UKŁAD res/menu/main.xml.

```
<item  
    android:id="@+id/action_settings  
    android:orderInCategory="180"  
    android:showAsAction="never"  
    android:title="Settings"/>  
  
<item  
    android:id="@+id/action_about  
    android:orderInCategory="200"  
    android:showAsAction="never"  
    android:title="About"/>  
  
</menu>
```

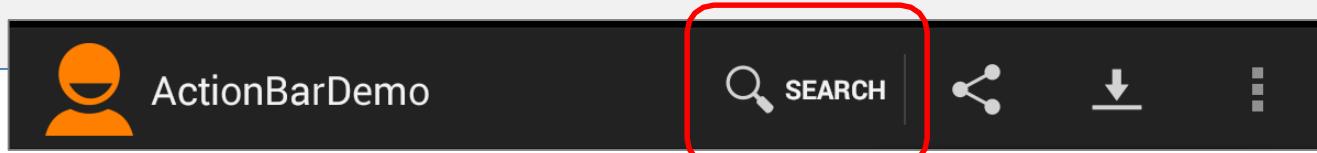


Menu

Przykład 1 – Proste wykorzystanie ActionBar

Każda pozycja menu <item> reprezentuje poszczególny kafelek:

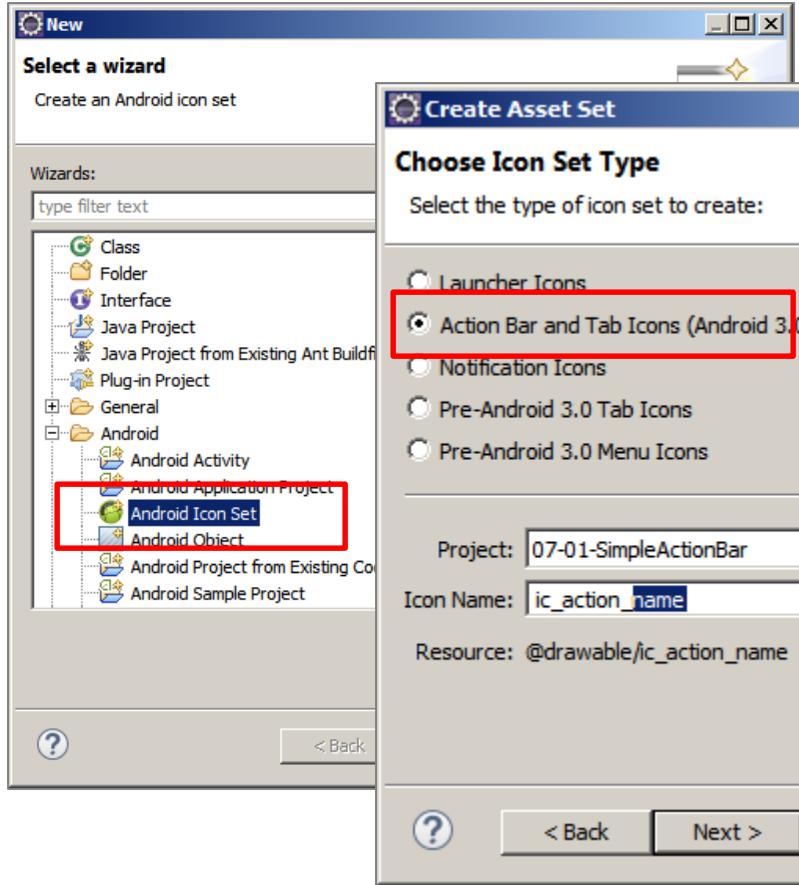
```
<item  
    android:id="@+id/action_search"  
    android:icon="@drawable/ic_action_search"  
    android:orderInCategory="120"  
    android:showAsAction="always/withText"  
    android:title="Search"/>
```



android:id	Identyfikator akcji ID (@+id/action_search), wymagany by widzieć która pozycja została wybrana
android:icon	Opcjonalna ikona powiązana z daną pozycją. Więcej informacji: http://developer.android.com/design/style/iconography.html
:orderInCategory	Relatywna pozycja tytułu na ActionBar (100, 120, 140, ...)
:showAsAction	Za pomocą tych klauzul możliwa jest kontrola położenia kafelek: “never”, “ifRoom”, “always”, “withText”, and “collapseActionView”.
:title	Opcjonalny tekst ('SEARCH') opisujący kafelek

Przykład 1 – Proste wykorzystanie ActionBar

Ikony



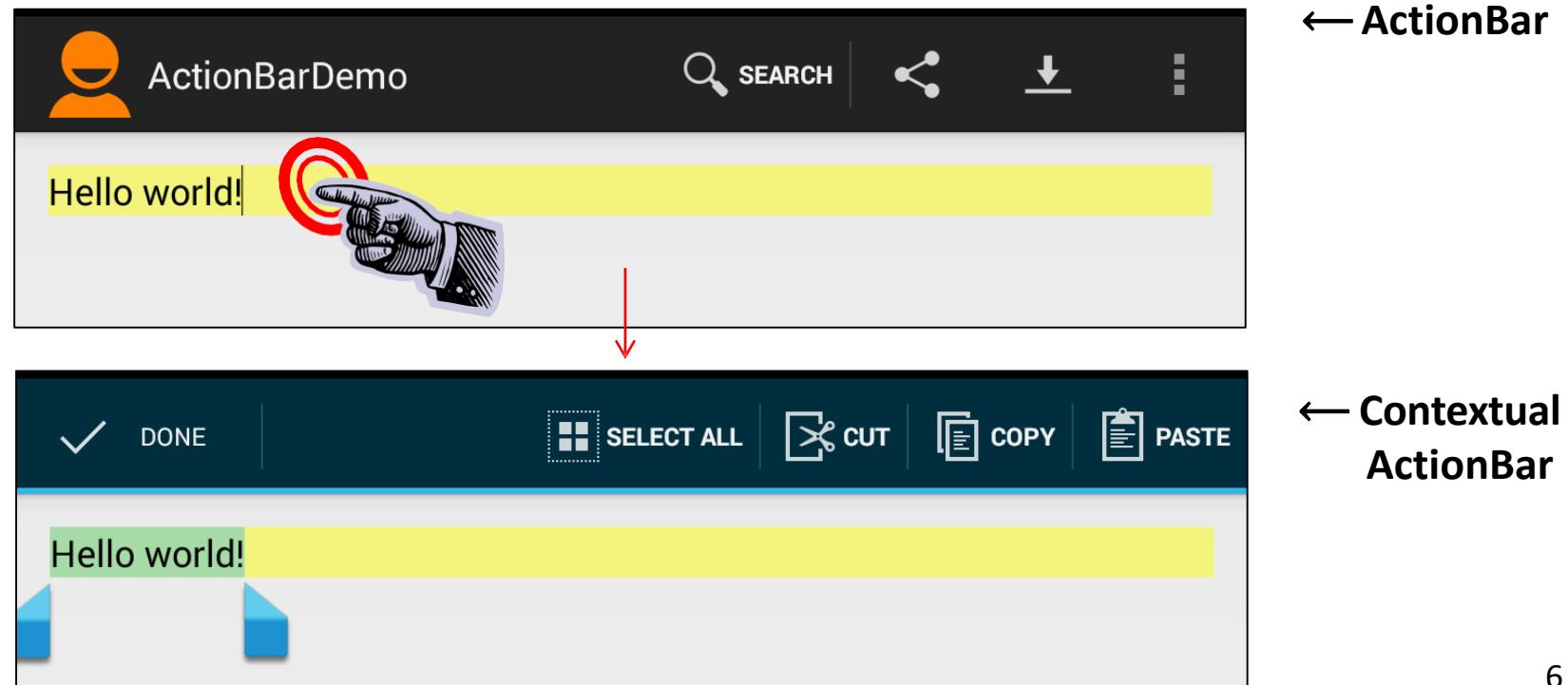
Kolekcja ikon dla
ActionBar

Przykład 1 – Proste wykorzystanie ActionBar

OBSERWACJA: Zmień plik `activity_main.xml` – zmień typ komponentu “HelloWorld” z **TextView** na **EditText**. Uruchom aplikację i przytrzymaj bądź kliknij dwukrotnie na polu tekstowym. ActionBar zmienia się w tzw. **Contextual ActionBar** (CAB) podobnie jak na zaprezentowanym obrazku.

ActionBar zależny od kontekstu wprowadza szereg nowych możliwości interakcji z użytkownikiem.

Sposób tworzenia tego typu komponentów zostanie przedstawiony później.



Przykład 1 – Proste wykorzystanie ActionBar

ActivityMain.java

```
public class MainActivity extends Activity {  
    EditText txtMsg;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        txtMsg = (EditText)findViewById(R.id.txtMsg);  
    }  
  
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        // Inflate the menu; add items to the action bar  
        getMenuInflater().inflate(R.menu.main, menu);  
        return true;  
    }  
  
    @Override  
    public boolean onOptionsItemSelected(MenuItem item) {  
        // user clicked a menu-item from ActionBar  
        int id = item.getItemId();  
  
        3 → if (id == R.id.action_search) {  
            txtMsg.setText("Search...");  
            // perform SEARCH operations...  
            return true;  
        }  
    }  
}
```

Przykład 1 – Proste wykorzystanie ActionBar

ActivityMain.java

```
3 → else if (id == R.id.action_share) {  
    txtMsg.setText("Share...");  
    // perform SHARE operations...  
    return true;  
}  
else if (id == R.id.action_download) {  
    txtMsg.setText("Download...");  
    // perform DOWNLOAD operations...  
    return true;  
}  
else if (id == R.id.action_about) {  
    txtMsg.setText("About...");  
    // perform ABOUT operations...  
    return true;  
}  
else if (id == R.id.action_settings) {  
    txtMsg.setText("Settings...");  
    // perform SETTING operations...  
    return true;  
}  
  
    return false;  
}
```

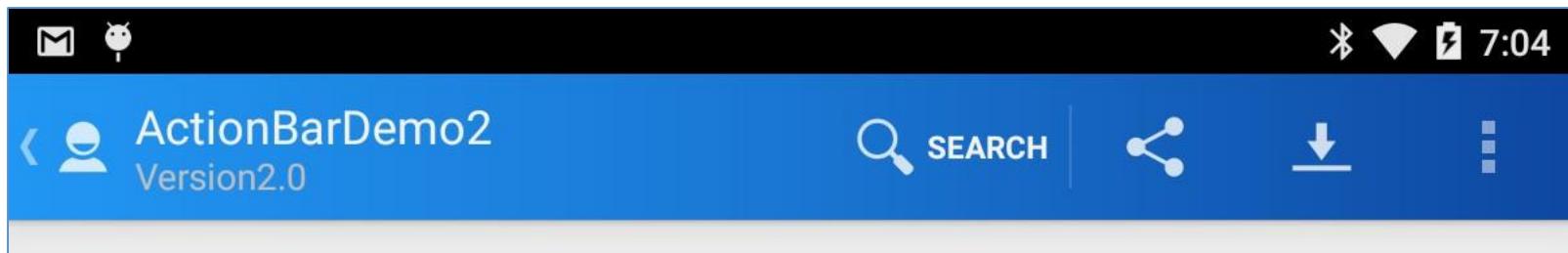
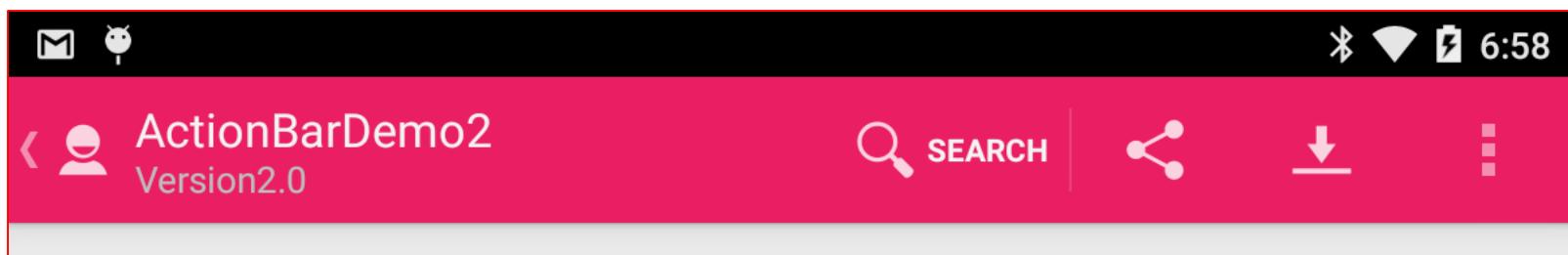
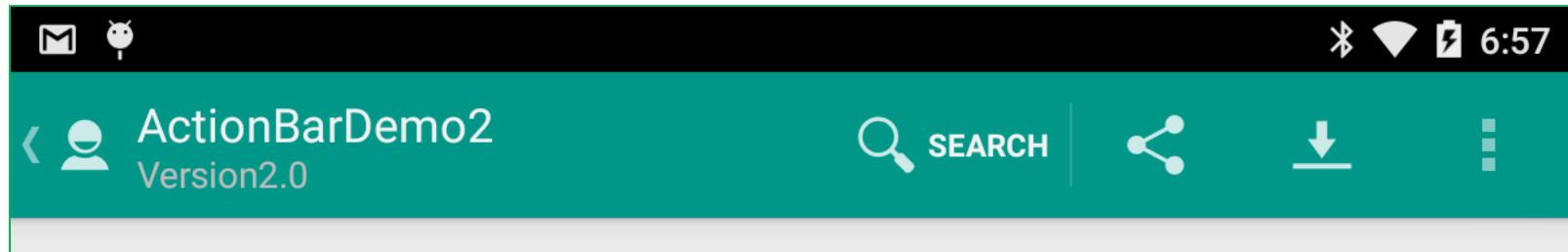
Przykład 1 – Proste wykorzystanie ActionBar

Komentarz: ActivityMain.java

1. Metoda `onCreateOptionsMenu()` jest wywoływana by przygotować menu główne. Plik xml `res/menu/main.xml` zawierający specyfikację ActionBar, na podstawie której tworzone są obiekty (z wykorzystaniem MenuInflater). Niektóre elementy prezentowane będą w postaci kafelek na ActionBarze (posiadających Ikonę/Tekst) natomiast pozostałe opcje będą wyświetlane po wciśnięciu przycisku menu.
2. Gdy użytkownik kliknie na określone miejsce ActionBar, jego identyfikator jest przekazywany do metody `onOptionsItemSelected()`. Tam następuje obsługa danego zdarzenia. Następnie zwracana jest wartość `true` by zasygnalizować, że zdarzenie zostało w pełni obsłużone.

Przykład 2 – Modyfikacja ActionBar

Przykład jest modyfikacją poprzedniego. ActionBar został zmodyfikowany uwzględniając inny kolor tła, logo oraz strzałki wyżej. Kolory wybrano z: <http://www.google.com/design/spec/style/color.html#color-color-palette>



Przykład 2 – Modyfikacja ActionBar

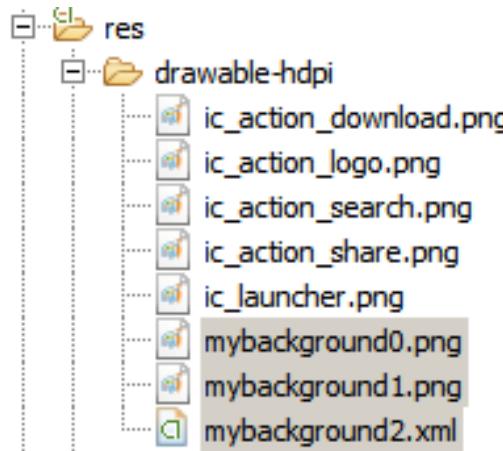
ActionBar zmieniony został programowo jako:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    txtMsg = (EditText)findViewById(R.id.txtMsg);  
  
    // setup ActionBar  
    1   actionBar = getActionBar();  
  
    2   actionBar.setTitle("ActionBarDemo2");  
    actionBar.setSubtitle("Version2.0");  
    actionBar.setLogo(R.drawable.ic_action_logo);  
  
    // choose one type of background  
    3   actionBar.setBackgroundDrawable(getResources().getDrawable(R.drawable.mybackground0));  
    actionBar.setBackgroundDrawable(getResources().getDrawable(R.drawable.mybackground1));  
    actionBar.setBackgroundDrawable(getResources().getDrawable(R.drawable.mybackground2));  
  
    4   actionBar.setDisplayShowCustomEnabled(true);      // allow custom views to be shown  
    actionBar.setDisplayHomeAsUpEnabled(true);          // show 'UP' affordance < button  
    actionBar.setDisplayShowHomeEnabled(true);           // allow app icon - logo to be shown  
    actionBar.setHomeButtonEnabled(true);               // needed for API14 or greater  
  
}
```

Przykład 2 – Modyfikacja ActionBar

Komentarz:

1. Wywołanie metody `getActionBar()` zwraca referencję do komponentu ActionBar. Poprzez referencję mamy dostęp do komponentów tam zawartych.
2. W tym przykładzie nowy tytuł i podtytuł jest przypisany do komponentu ActionBar. W przypadku skomplikowanych aplikacji z dużym zestawem ekranów jest to wręcz obowiązkowa czynność by ułatwić użytkownikowi nawigacje.
3. Logo może zostać zmienione poprzez metodę `.setLogo(drawable)`. Podobnie tło ActionBar może zostać zmienione na dowolny obrazek. Zwykle tego typu zasoby znajdują się w katalogu **res/drawable**.



mybackground0.png



mybackground1.png



Przykład 2 – Modyfikacja ActionBar

Komentarz:



3. Można również zdefiniować gradient jako tło dla ActionBar

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"  
       android:shape="rectangle"  
  
       <gradient  
           android:angle="0"  
           android:centerColor="#1976D2"  
           android:centerX="50%"  
           android:endColor="#0D47A1"  
           android:startColor="#2196F3"  
           android:type="linear" />  
  
</shape>
```

4. Można również zdefiniować widoczność niektórych elementów:

```
// set ActionBar options  
actionBar.setDisplayShowCustomEnabled(true); // allow custom views to be shown  
actionBar.setDisplayHomeAsUpEnabled(true); // show 'UP' affordance < button  
actionBar.setDisplayShowHomeEnabled(true); // allow app icon - logo to be shown  
actionBar.setHomeButtonEnabled(true); // needed for API.14 or greater
```

Przykład 2 – Modyfikacja ActionBar

Komentarz:

4. Zwykle w celu ukrycia lub aktywacji pewnych opcji wykorzystywana jest pojedyncza instrukcja:

```
// set ActionBar options  
  
actionBar.setDisplayOptions( ActionBar.DISPLAY_SHOW_TITLE  
                           | ActionBar.DISPLAY_SHOW_HOME  
                           | ActionBar.DISPLAY_HOME_AS_UP  
                           | ActionBar.DISPLAY_SHOW_CUSTOM );
```

Dodatkowe zasoby:

Gradienty <http://angrytools.com/gradient/>

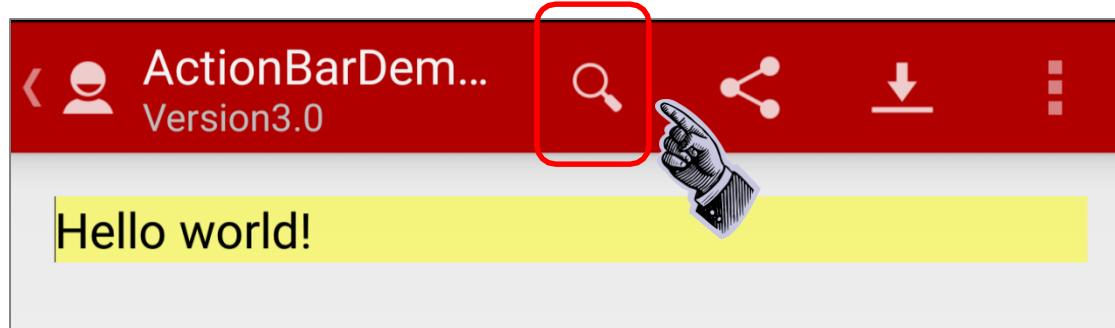
Kolory <http://www.google.com/design/spec/style/color.html#color-color-palette>

Grafiki <http://developer.android.com/guide/topics/resources/drawable-resource.html>

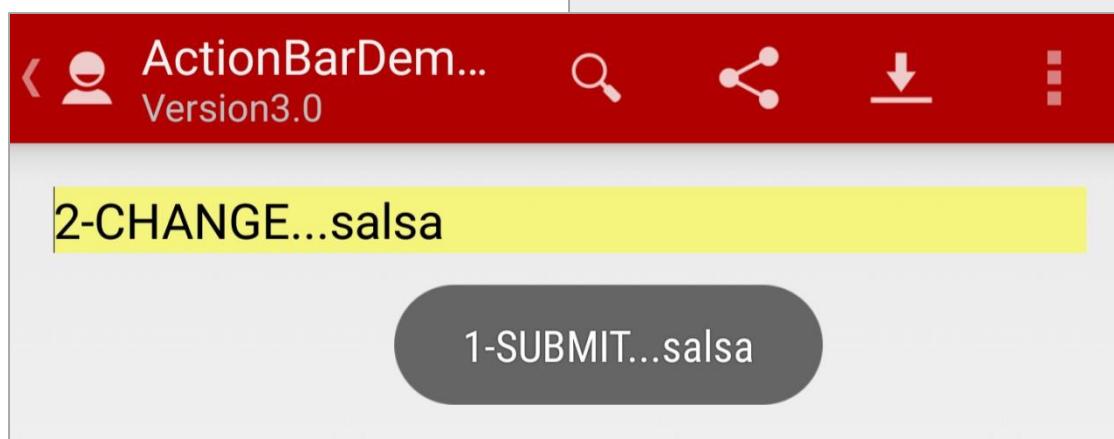
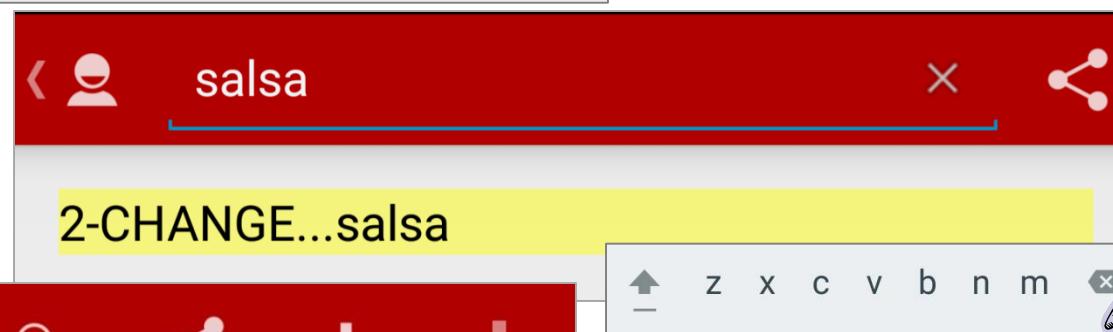
Przykład 3 – Kafelek do wyszukiwania



Jest to wariant Przykładu 1. Opcja wyszukiwania została zaimplementowana przez widżet **SearchView** jako opcja menu. Gdy użytkownik kliknie na opcję wyszukiwania tekst wpisany w polu wyszukiwania jest wykorzystywany do uaktywniania tego procesu.



Naciśnij **x** by wyczyścić



Naciśnij **search icon** by zamknąć i rozpoczęć wyszukiwanie

Przykład 3 – Kafelek do wyszukiwania

1 z 2



Pole typu **SearchView** jest zdefiniowane w pliku XML podobnie jak w zaprezentowanym przykładzie:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
      xmlns:app="http://schemas.android.com/apk/res-auto"  
      xmlns:tools="http://schemas.android.com/tools"  
      tools:context="csu.matos.MainActivity" >
```

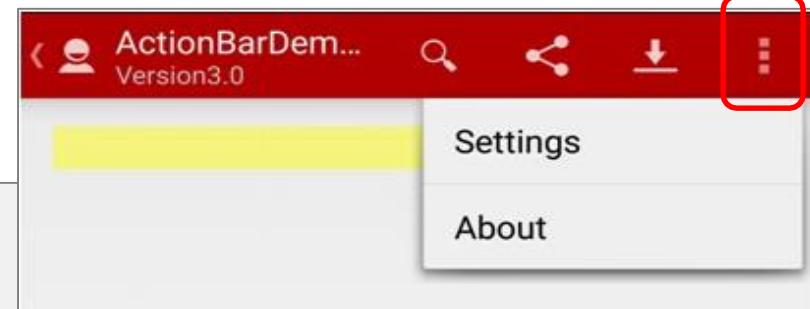
1 →

```
    <item  
        android:id="@+id/action_search"  
        android:actionViewClass="android.widget.SearchView"  
        android:orderInCategory="120"  
        android:showAsAction="always/withText"  
        android:title="Search"/>
```

```
    <item  
        android:id="@+id/action_share"  
        android:icon="@drawable/ic_action_share"  
        android:orderInCategory="140"  
        android:showAsAction="always"  
        android:title="Share"/>
```

```
    <item  
        android:id="@+id/action_download"  
        android:icon="@drawable/ic_action_download"  
        android:orderInCategory="160"  
        android:showAsAction="always"  
        android:title="Download"/>
```





```
<item  
    android:id="@+id/action_settings"  
    android:orderInCategory="180"  
    android:showAsAction="never"  
    android:title="Settings"/>  
  
<item  
    android:id="@+id/action_about"  
    android:orderInCategory="200"  
    android:showAsAction="never"  
    android:title="About"/>  
  
</menu>
```

Komentarz:

- Element `action_search` nie zawiera klauzuli `:icon` – zamiast wykorzystuje klauzule `android:actionViewClass="android.widget.SearchView"` by ustawić widok który rozwinięty umożliwia użytkownikowi wprowadzenie kwerendy oraz rozpoczęcie procesu wyszukiwania.



```
public class MainActivity extends Activity {  
    EditText txtMsg;  
    ActionBar actionBar;  
    SearchView txtSearchValue;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        txtMsg = (EditText) findViewById(R.id.txtMsg);  
  
        // setup the ActionBar  
        actionBar = getActionBar();  
        actionBar.setTitle("ActionBarDemo3");  
        actionBar.setSubtitle("Version3.0");  
        actionBar.setLogo(R.drawable.ic_action_logo);  
        actionBar.setBackgroundDrawable(getResources().getDrawable(  
                R.drawable.mybackground1));  
  
        actionBar.setDisplayShowCustomEnabled(true); // allow custom views to be shown  
        actionBar.setDisplayHomeAsUpEnabled(true); // show 'UP' affordance < button  
        actionBar.setDisplayShowHomeEnabled(true); // allow app icon - logo to be shown  
        actionBar.setHomeButtonEnabled(true); // needed for API14 or greater  
    }  
}
```

Kod wzorowany na przykładzie 2.



```
1 @Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the options menu to adds items from menu/main.xml into the ActionBar
    getMenuInflater().inflate(R.menu.main, menu);
    // get access to the collapsible SearchView
    txtSearchValue = (SearchView) menu.findItem(R.id.action_search)
                    .getActionView();
    // set searchView listener (look for text changes, and submit event)
    txtSearchValue.setOnQueryTextListener(new OnQueryTextListener() {

        2 @Override
        public boolean onQueryTextSubmit(String query) {
            Toast.makeText(getApplicationContext(), "1-SUBMIT..." + query,
                Toast.LENGTH_SHORT).show();
            // recreate the 'original' ActionBar (collapse the SearchBox)
            invalidateOptionsMenu();
            // clear searchView text
            txtSearchValue.setQuery("", false);
            return false;
        }

        3 @Override
        public boolean onQueryTextChange(String newText) {
            // accept input one character at the time
            txtMsg.append("\n2-CHANGE..." + newText);
            return false;
        }
    });
    return true;
}
```



```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle ActionBar item clicks here.
    // NOTE: Observe that SEARCH menuItem is NOT processed in this
    // method (it has its own listener set by onCreateOptionsMenu)

    int id = item.getItemId();

    if (id == android.R.id.home) {
        txtMsg.setText("Home...");
        return true;
    } else if (id == R.id.action_share) {
        txtMsg.setText("Share...");

        return true;
    } else if (id == R.id.action_download) {
        txtMsg.setText("Download...");
        return true;
    } else if (id == R.id.action_about) {
        txtMsg.setText("About...");
        return true;
    } else if (id == R.id.action_settings) {
        txtMsg.setText("Settings...");
        return true;
    }

    return false;
} //onOptionsItemSelected

} //MainActivity
```

4



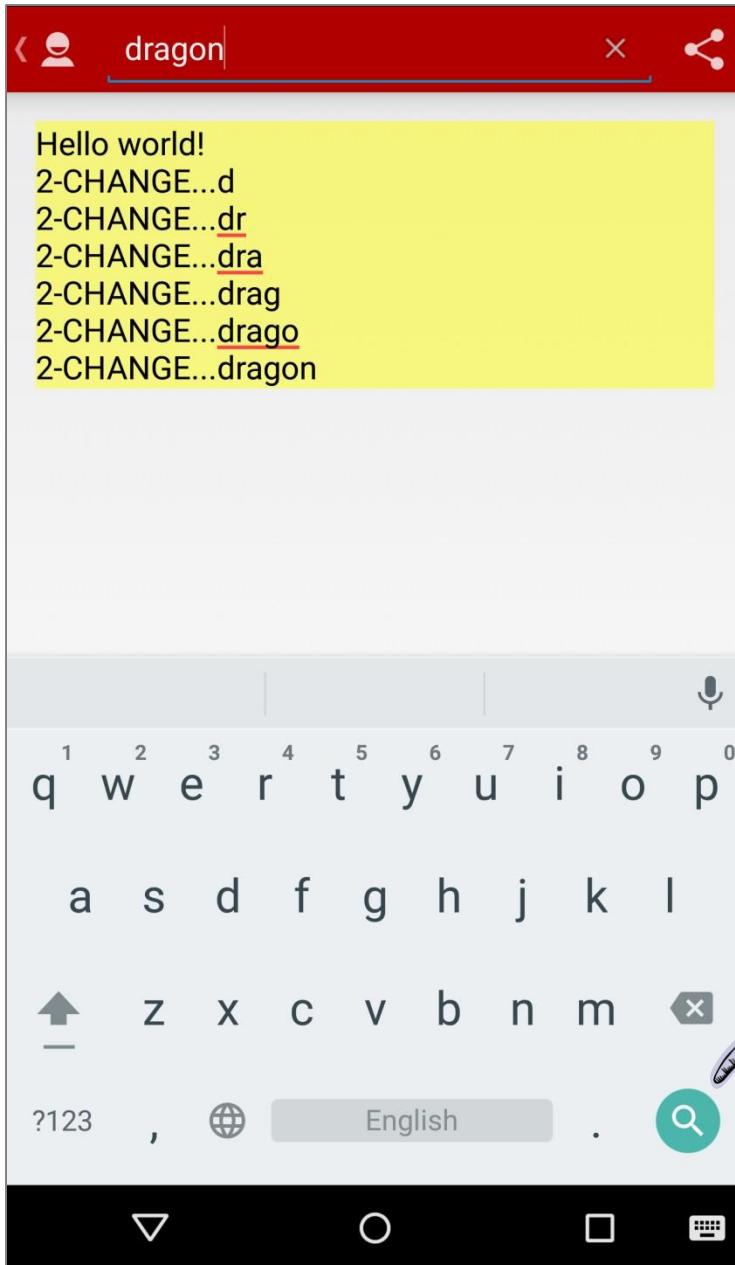
Komentarz

1. Podczas wywołania metody **onCreateOptionsMenu()** elementy definiowane w pliku zasobów *menu/main.xml* są dodane do ActionBar. Instrukcja

```
txtSearchValue = (SearchView) menu.findItem(R.id.action_search).getActionView();
```

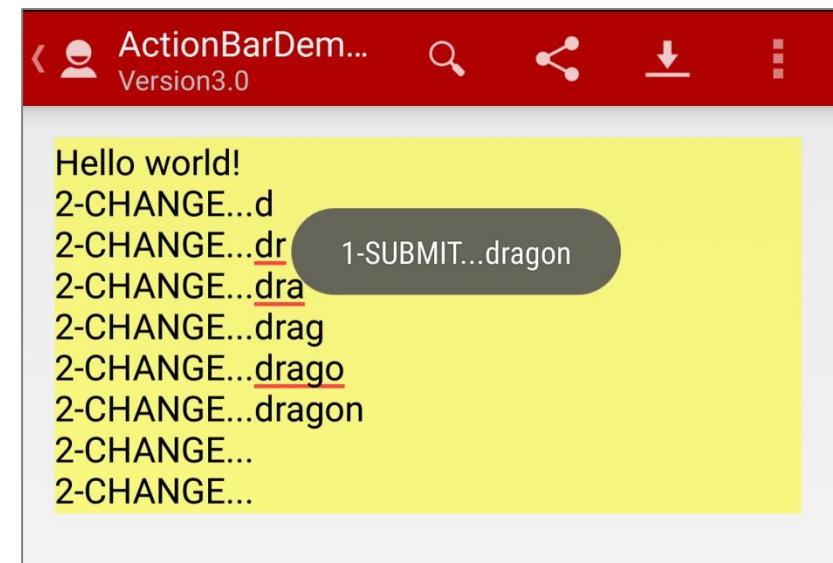
daje dostęp do elementu SearchView, reprezentowany jako ikona lupy.
2. Następny krok polega na zdefiniowaniu **QueryTextListener** powiązany z komponentem SearchView. Nasłuchiwanie ma dwie metody. Pierwsza – **onQueryTextSubmit()** – wywoływana jest kiedy użytkownik kliknie na ikonę wyszukiwania. W tym momencie, tekst zawarty wewnątrz SearchView może zostać wysłany do zdefiniowanego przez użytkownika *dostawcy wyszukiwania*. Instrukcja *invalidateOptionsMenu()* zamyka pasek wyszukiwania i odświeża widokActionBara. Instrukcja *.setQuery("", false)* wykorzystywana jest by wyczyścić tekst nowo utworzonego komponentu SearchView (który nie jest jeszcze widoczny).
3. Druga metoda – **onQueryTextChange** – implementuje **mechanizm text-watcher** wywoływany kiedy nowy znak wprowadzony jest do SearchView. Metoda ta może być wykorzystana by pokazać użytkownikowi sugestie ciągów do wyszukiwania.

Przykład 3 – Kafelek do wyszukiwania

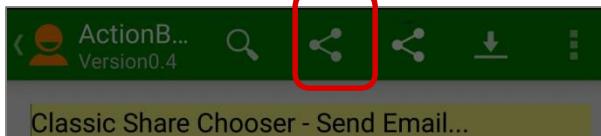


Rezultat działania przykładu 3. Nasłuchiwanie reaguje po każdorazowym wpisaniu nowego znaku.

Komunikat typu toast pojawia się po zaakceptowaniu ciągu do wyszukiwania.



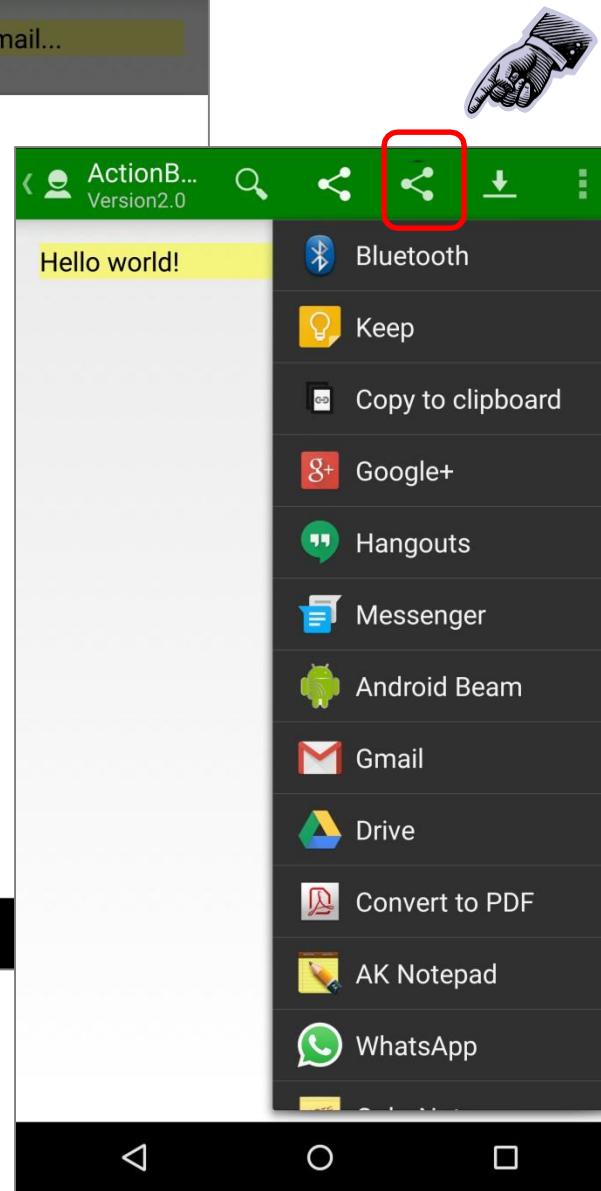
Przykład 4 – Przycisk do udostępniania



Send Email Using...

- Skype
- Messenger
- Facebook
- WhatsApp
- Gmail
- Twitter
- Google+
- AK Notepad
- Android Beam

(a) Chooser Share



(b) Easy Share

W tym przykładzie aplikacja zostanie wzbogacona o możliwość udostępniania danych.

Użytkownik może wybrać jakie dane oraz przez jaki kanał komunikacji można wysłać.

Przykład prezentuje dwa warianty:

(a) klasyczny **dialog wyboru** z którego użytkownik wybiera sposób komunikacji

(b) z wykorzystaniem kafelka „**łatwego udostępniania**” bazującym na widżecie *SharedActionProvider* dostępnym od API-14.

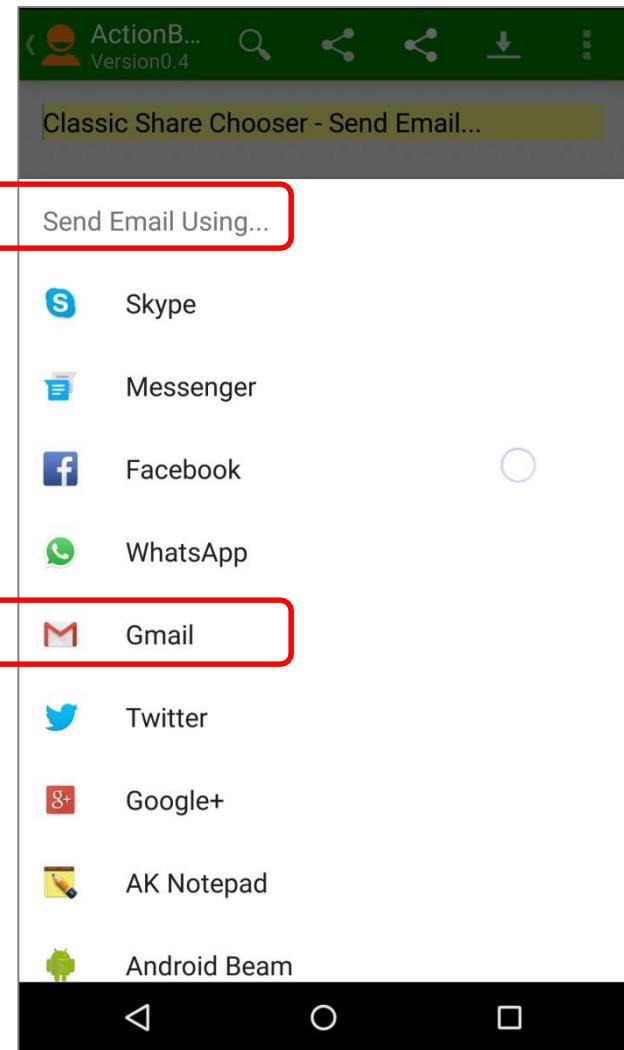
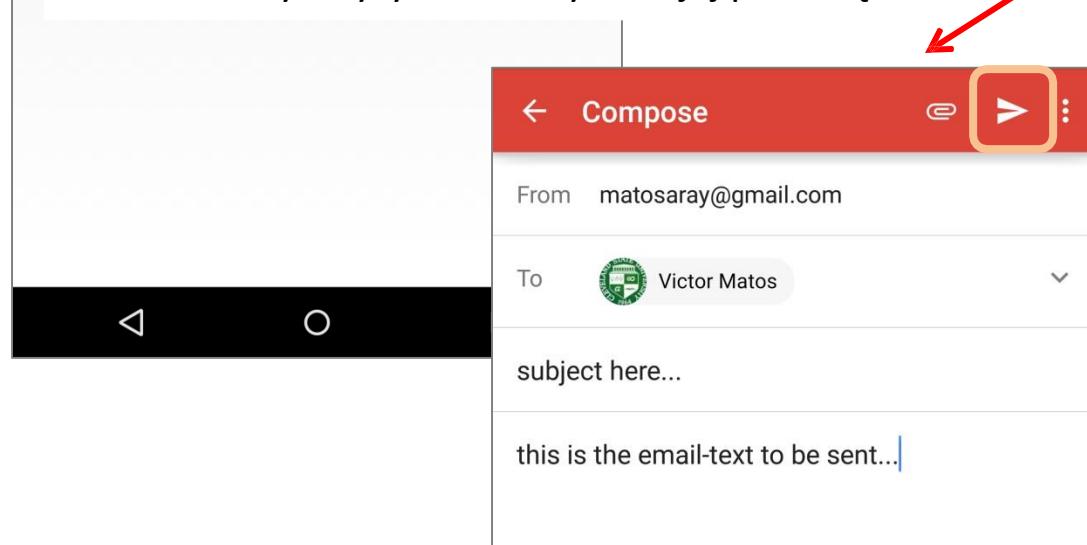
Przykład 4 – Przycisk do udostępniania



Instrukcja

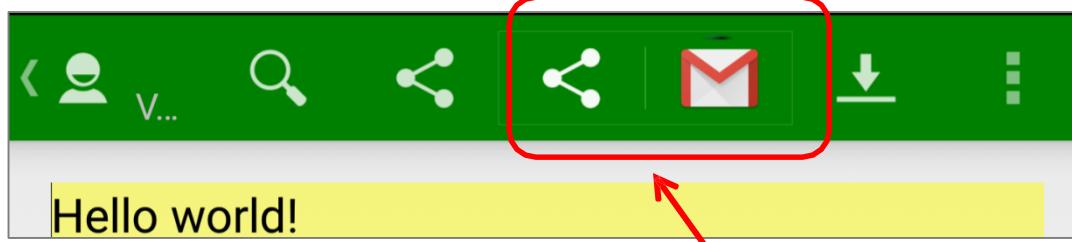
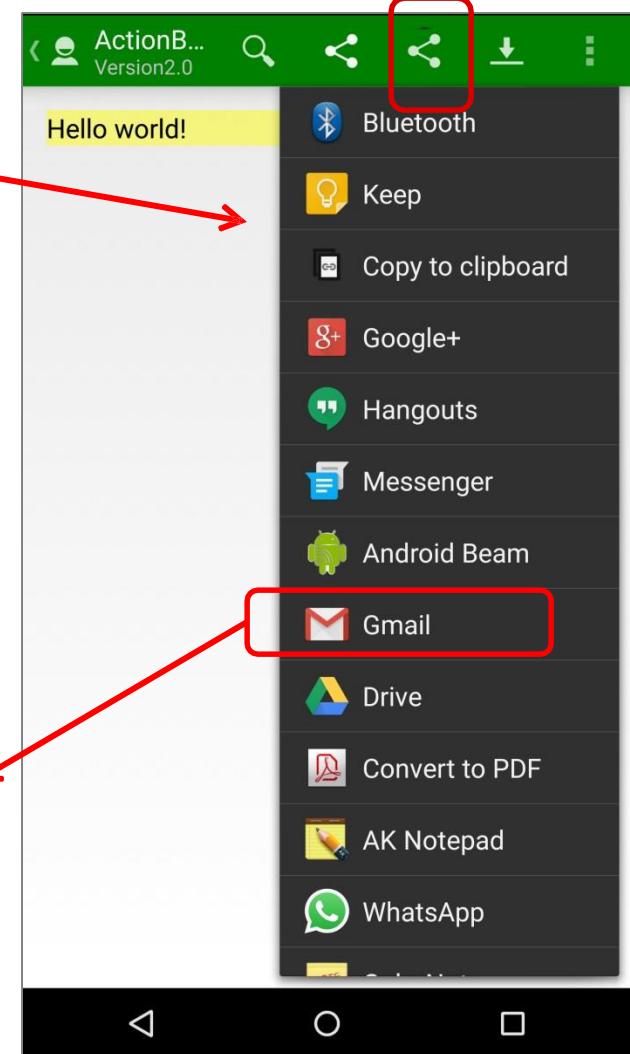
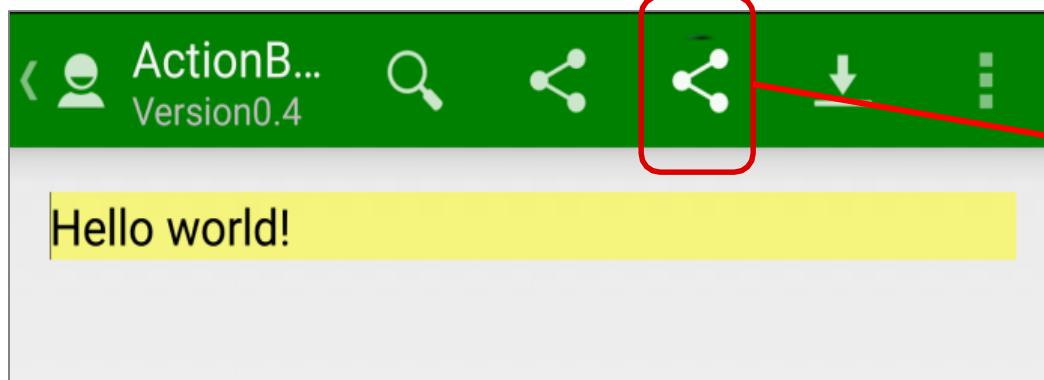
```
startActivity(Intent.createChooser(  
    emailIntent(),  
    "Send EMAIL Using..."));
```

- Wyświetla listę wszystkich aplikacji obsługujących intencję ACTION_SENDTO
- Użytkownik wybiera aplikację z listy by dokończyć wysyłanie danych za jej pomocą.



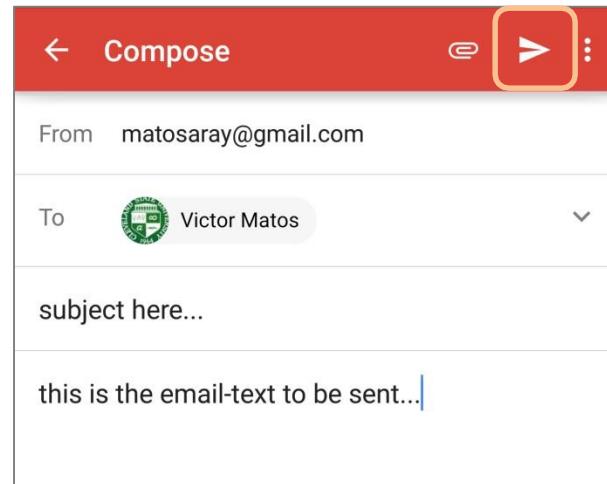
(a) Klasyczny dialog

Przykład 4 – Przycisk do udostępniania



Po wyborze dostawcy
ActionBar ulega
zmianie.

Wybrana opcja
dodawana jest jako
domyślna.

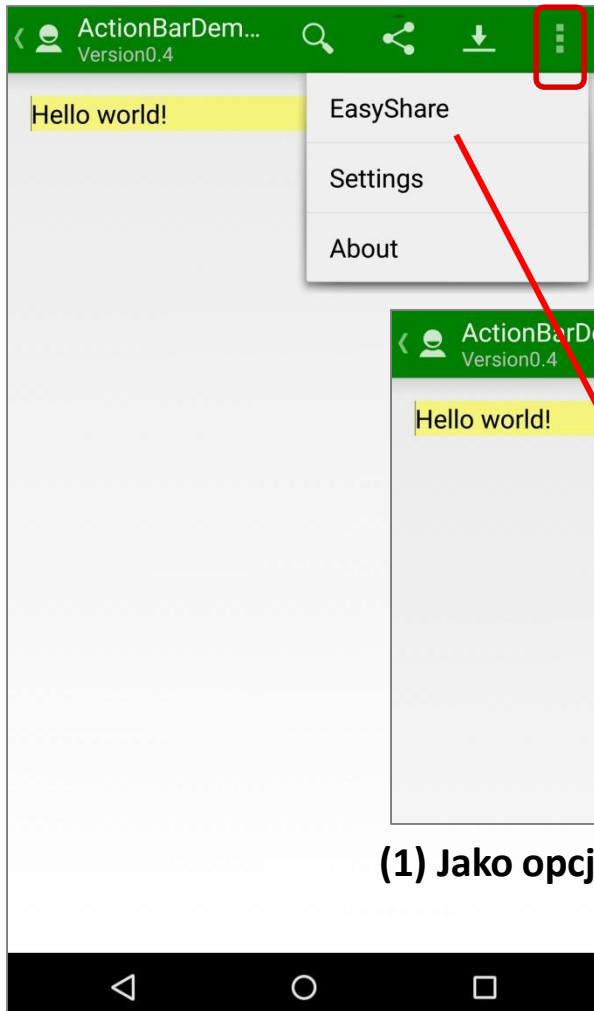


(b) Łatwe udostępnianie

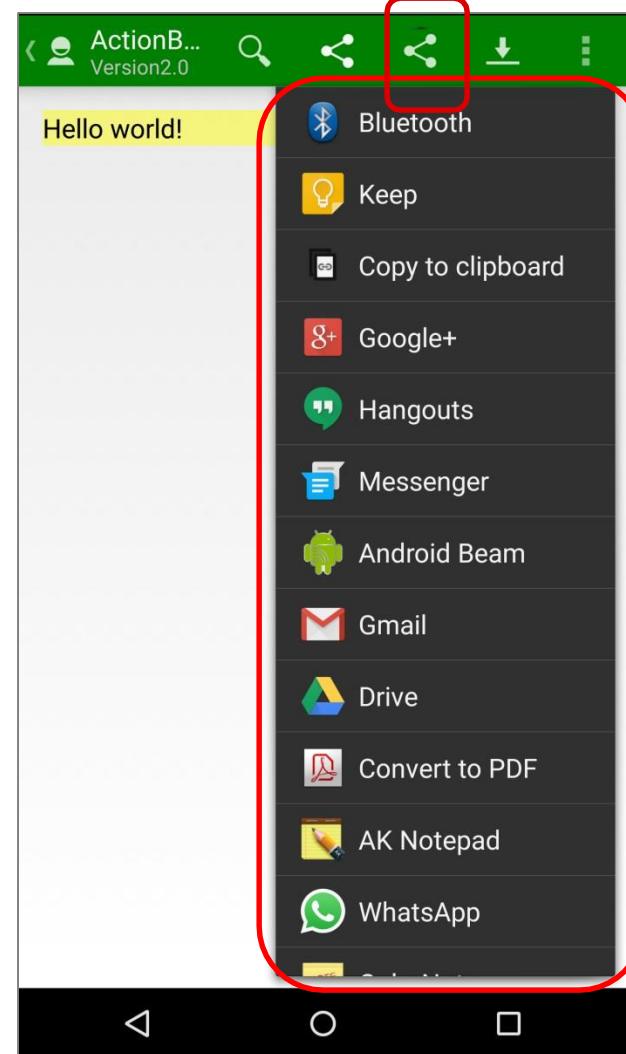
Przykład 4 – Przycisk do udostępniania



Widok łatwego udostępniania może być wyświetlany zarówno na komponentie ActionBar jak i w menu głównym.



(1) Jako opcja menu głównego



(2) Jako część ActionBar'a



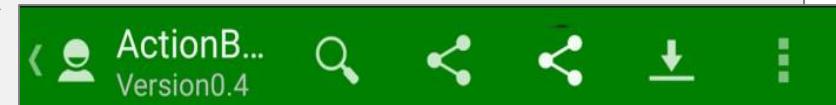
Plik zasobów MENU uwzględniający widżet ShareActionProvider:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="csu.matos.MainActivity" >

    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always/withText"
        android:title="Search"/>

    <item
        android:id="@+id/action_share_chooser"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="ShareChooser" />

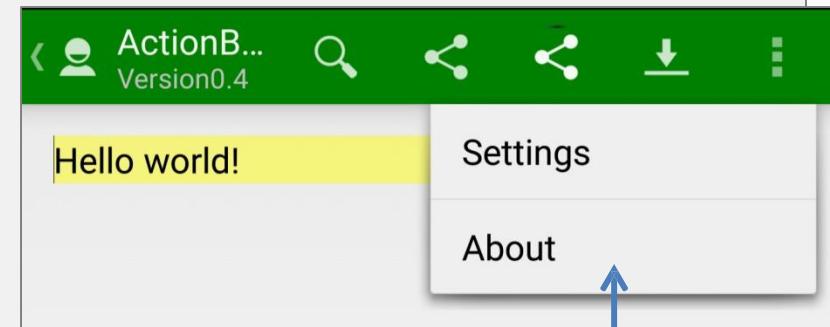
    <item
        android:id="@+id/action_share_easy"
        android:orderInCategory="145"
        android:showAsAction="always"
        android:title="EasyShare"
        android:actionProviderClass="android.widget.ShareActionProvider" />
```





Plik zasobów MENU uwzględniający widżet ShareActionProvider:

```
<item  
    android:id="@+id/action_download"  
    android:icon="@drawable/ic_action_download"  
    android:orderInCategory="160"  
    android:showAsAction="always"  
    android:title="Download"/>  
  
<item  
    android:id="@+id/action_settings"  
    android:orderInCategory="180"  
    android:showAsAction="never"  
    android:title="Settings"/>  
  
<item  
    android:id="@+id/action_about"  
    android:orderInCategory="200"  
    android:showAsAction="never"  
    android:title="About"/>  
  
</menu>
```





```
public class MainActivity extends Activity {  
    EditText txtMsg;  
    ActionBar actionBar;  
    ShareActionProvider shareActionProvider;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        txtMsg = (EditText)findViewById(R.id.txtMsg);  
  
        // setup ActionBar  
        actionBar = getActionBar();  
  
        actionBar.setTitle("ActionBarDemo4");  
        actionBar.setSubtitle("Version0.4");  
        actionBar.setLogo(R.drawable.ic_action_logo);  
        actionBar.setBackgroundDrawable(getResources()  
            .getDrawable(R.drawable.mybackground0));  
  
        actionBar.setDisplayShowCustomEnabled(true); // allow custom views to be shown  
        actionBar.setDisplayHomeAsUpEnabled(true); // show 'UP' affordance < button  
        actionBar.setDisplayShowHomeEnabled(true); // allow app icon - logo to be shown  
        actionBar.setHomeButtonEnabled(true); // needed for API14 or greater  
    }  
}
```

Kod wzorowany na
Przykładzie 2.



```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
  
    // Inflate the menu, add ShareChooser & EasyShare tiles  
    getMenuInflater().inflate(R.menu.main, menu);  
  
    1    // Locate MenuItem holding ShareActionProvider  
    MenuItem easySharedItem = menu.findItem(R.id.action_share_easy);  
  
    // prepare EASY SHARE action tile:  
    // Fetch and store a ShareActionProvider for future usage  
    // an intent assembles the email(or SMS), you need only to select carrier  
    shareActionProvider = (ShareActionProvider) easySharedItem.getActionProvider();  
  
    // prepare an EMAIL  
    shareActionProvider.setShareIntent( emailIntent() );  
  
    // prepare an SMS - try this later...  
    // shareActionProvider.setShareIntent( smsIntent() );  
  
    return super.onCreateOptionsMenu(menu);  
}
```

Przykład 4 – Przycisk do udostępniania

3 z 4



2

```
// return a SHARED intent to deliver an email
private Intent emailIntent() {

    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.setType("text/plain");
    intent.putExtra(Intent.EXTRA_EMAIL, new String[] { "v.matos@csuohio.edu" });
    intent.putExtra(Intent.EXTRA_SUBJECT, "subject here...");  
intent.putExtra(Intent.EXTRA_TEXT, "this is the email-text to be sent...");

    return intent;
}
```

3

```
// return a SHARED intent to deliver an SMS text-message
private Intent smsIntent() {

    Intent intent = new Intent(Intent.ACTION_VIEW);
    String yourNumber="+ 1 216 555-4321";
    intent.setData( Uri.parse("sms:" + yourNumber) );
    intent.putExtra("sms_body", "Here goes my msg");

    return intent;
}
```

Przykład 4 – Przycisk do udostępniania

4 z 4



```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. Observe EasyShare is NOT handled here!
    int id = item.getItemId();
    if (id == R.id.action_search) {
        txtMsg.setText("Search...");
        return true;
    }
    else if (id == R.id.action_share_chooser) {
        txtMsg.setText("Classic Share Chooser - Send Email...");
        startActivityForResult( Intent.createChooser( emailIntent(), "Send EMAIL Using..." ) );
        //startActivity(Intent.createChooser(smsIntent(), "Send SMS Using..."));
        return true;
    }
    else if (id == R.id.action_download) {
        txtMsg.setText("Download...");
        return true;
    }
    else if (id == R.id.action_about) {
        txtMsg.setText("About...");
        return true;
    }
    else if (id == R.id.action_settings) {
        txtMsg.setText("Settings...");
        return true;
    }
    return false;
}
}//Activity
```

4



Komentarz

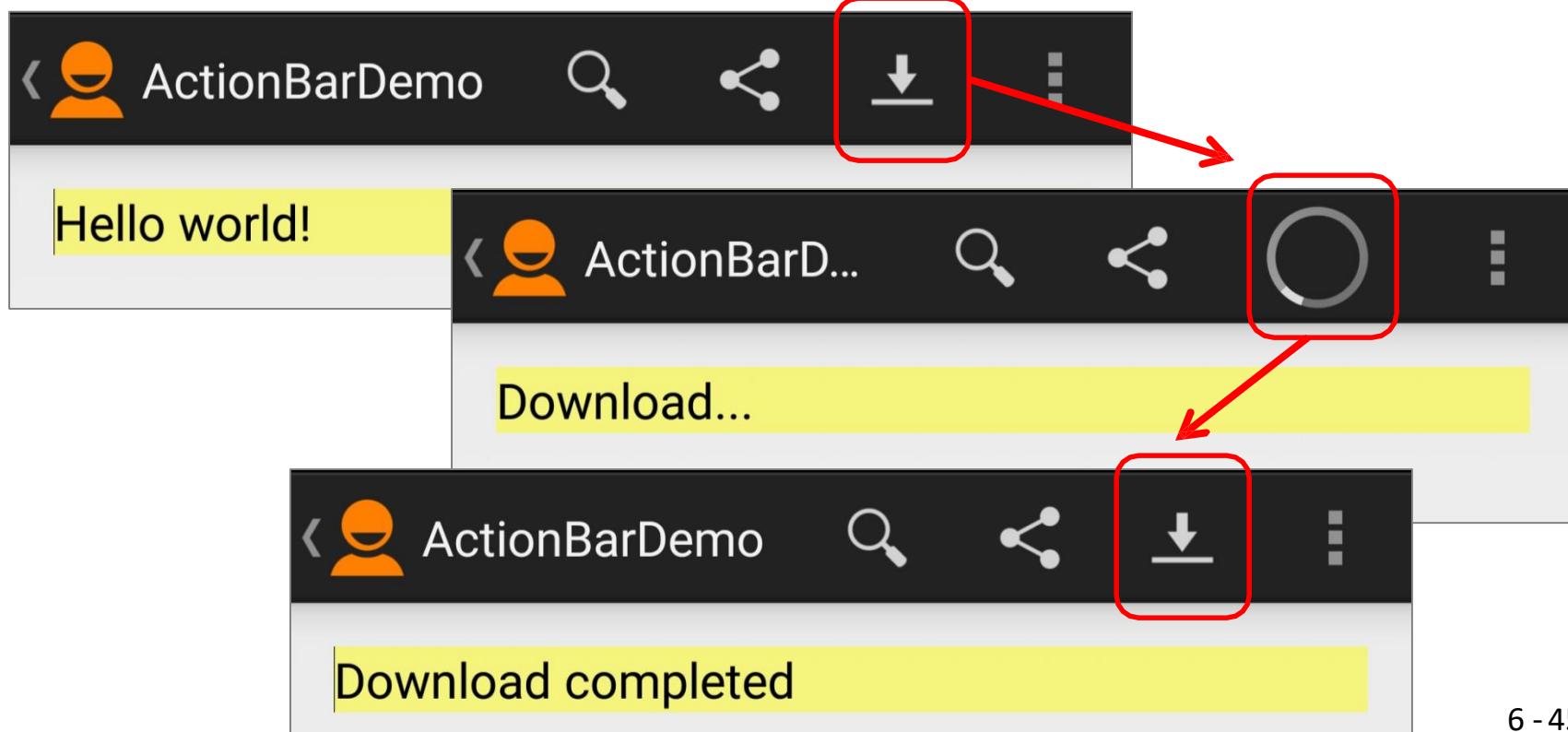
1. Metoda **onCreateOptionsMenu** jest wykorzystywana do stworzenia obiektów na podstawie specyfikacji XML. Element `@+id+action_share_easy` zawierający widżet *ShareAccessProvider* jest powiązany z obiektem *easyShareProvider*. Jest to używane do kontrolowania wersji z łatwym udostępnianiem. Później wspomniany obiekt jest wiązany z intencją (ACTION_VIEW, ACTION_SENDTO) która zostanie użyta by dostarczyć wybrane dane.
2. Metoda **emailIntent()** zwraca intencję w której zostaje stworzona prosta wiadomość mailowa. Wiadomość zawiera typ, odbiorców, temat i treść.
3. Metoda **smsIntent()** zwraca intencję w której zostaje stworzona wiadomość SMS. Wiadomość zawiera typ, odbiorcę oraz treść.
4. Metoda **onOptionsItemSelected()** jest wykorzystywana by rozpoznać wybór użytkownika (gdy wykorzystywany jest wariant z dialogiem). Implikuje to wywołanie metody *Intent.createChooser(...)* by wyświetlić listę aplikacji umożliwiających wysyłanie danych. Wariant z łatwym udostępnianiem nie jest przetwarzany w tej metodzie.

Przykład 5 – Kafelek do ściągania



W tym przykładzie ActionBar wyświetla ikonę ściągania. Po kliknięciu wywołuje **zdefiniowaną przez użytkownika długotrwałą operację** (jak przeszukiwanie bazy danych czy operacje internetowe). Tego typu czynności nie powinny być wykonywane w głównym wątku aplikacji.

Zadanie wykonywane jest w tle. Użytkownik jest informowany o stanie wątku w tle poprzez okrągły pasek postępu. Gdy zadanie zostanie ukończone, aktywność jest o tym informowana przez odpowiednie wywołanie zwrotne.





UKŁAD XML:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="csu.matos.MainActivity" >

    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always/withText"
        android:title="Search"/>

    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>

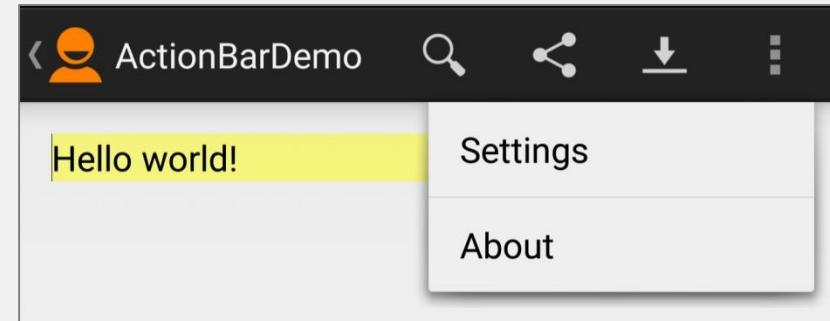
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
```





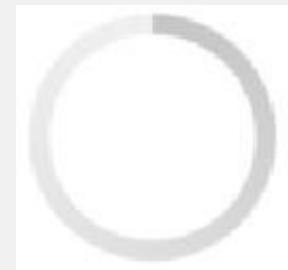
UKŁAD XML:

```
<item  
    android:id="@+id/action_settings"  
    android:orderInCategory="180"  
    android:showAsAction="never"  
    android:title="Settings"/>  
  
<item  
    android:id="@+id/action_about"  
    android:orderInCategory="200"  
    android:showAsAction="never"  
    android:title="About"/>  
  
</menu>
```



Plik **res/layout/custom_view_download** definiujący okrągły pasek postępu:

```
<?xml version="1.0" encoding="utf-8"?>  
  
<ProgressBar  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/customViewActionProgressBar"  
    style="?android:attr/progressBarStyleLarge"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```





MainActivity

```
public class MainActivity extends Activity {

    EditText txtMsg;
    ActionBar actionBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtMsg = (EditText) findViewById(R.id.txtMsg);

        // setup the ActionBar
        actionBar = getActionBar();
        actionBar.setDisplayShowCustomEnabled(true); // allow custom views to be shown
        actionBar.setDisplayHomeAsUpEnabled(true); // show 'UP' affordance < button
        actionBar.setDisplayShowHomeEnabled(true); // allow app icon - logo to be shown
        actionBar.setHomeButtonEnabled(true); // needed for API.14 or greater
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the resource menu file: main.xml
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```



```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle clicking of ActionBar items here.
    int id = item.getItemId();

    if (id == R.id.action_search) {
        txtMsg.setText("Search...");
        return true;
    } else if (id == R.id.action_share) {
        txtMsg.setText("Share...");
        return true;

    } else if (id == R.id.action_download) {
        txtMsg.setText("Download...");
        // Temporarily replace the download action-tile with circular progress bar
        performSlowOperation(item);

        return true;
    } else if (id == R.id.action_about) {
        txtMsg.setText("About...");
        return true;
    } else if (id == R.id.action_settings) {
        txtMsg.setText("Settings...");
        return true;
    }

    return false;
}
```

1 →



```
private void performSlowOperation(MenuItem item) {  
    // temporally replace Download action-icon with a progress-bar  
    final MenuItem downloadActionItem = item;  
    downloadActionItem.setActionView(R.layout.custom_view_download);  
    downloadActionItem.expandActionView();  
    // define an Android-Handler control to receive messages from a  
    // background thread were the slow work is to be done  
    final Handler handler = new Handler() {  
        @Override  
        public void handleMessage(Message msg) {  
            super.handleMessage(msg);  
            txtMsg.setText("Download completed"); // announce completion of slow job  
            downloadActionItem.collapseActionView(); // dismiss progress-bar  
            downloadActionItem.setActionView(null);  
        }  
    };  
    // a parallel Thread runs the slow-job and signals its termination  
    new Thread() {  
        @Override  
        public void run() {  
            // real 'BUSY_WORK' goes here...(we fake 2 seconds)  
            long endTime = SystemClock.uptimeMillis() + 2000; //now + 2 seconds  
            handler.sendMessageAtTime(handler.obtainMessage(), endTime);  
            super.run();  
        }  
    }.start();  
}  
  
}//Activity
```



Komentarz

1. Po kliknięciu przez użytkownika w kafelek ściągania metoda **onOptionsItemSelected()** rozpoznaje zdarzenie oraz wywołuje nowy wątek działający w tle.
2. Wątek rozpoczyna działanie od zamiany ikony do ściągania na okrągły pasek postępu (patrz metody **.setActionView(...)** oraz **expandActionView()**).
3. Tworzony jest obiekt do nasłuchiwania komunikatów wysyłanych przez wątek pracujący w tle. Gdy nadjejdzie wiadomość "zakończono" obiekt usuwa pasek postępu i przywraca oryginalny stan ActionBar'a (patrz metody **.collapseActionView()** oraz **.setActionView(null)**).
4. Osobny wątek symuluje długotrwałą operację trwającą 5 sekund. Dzięki pracy w tle, główny wątek w dalszym ciągu jest responsywny i użytkownik może (podczas oczekiwania na zakończenie zadania) wykonywać również inne czynności.

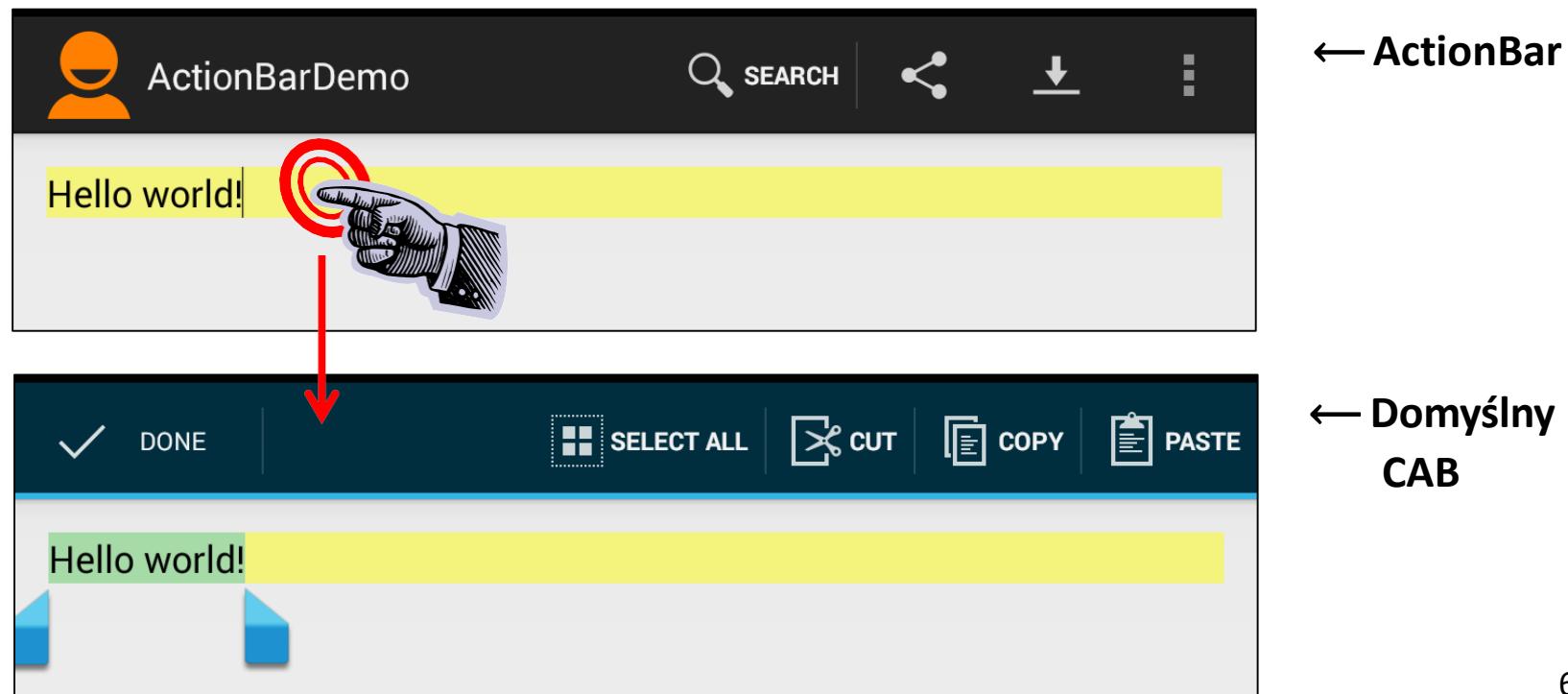
Więcej na temat wielowątkowości znajduje się w osobnej prezentacji.

Przykład 6 – Kontekstowy ActionBar (CAB)



Standardowy ActionBar może zostać rozszerzony o reagowanie na kontekst danego zdarzenia – wywoływanego przez przytrzymanie palca na określonym komponencie. Wówczas ActionBar jest zastępowany przez tzw. **Kontekstowy ActionBar (CAB)**. Jest to niejako alternatywa dla wykorzystania menu kontekstowego, co było popularnym wzorcem w aplikacjach pisanych dla starszych wersji Androida.

Przykład 1 Prezentował najprostszy sposób wykorzystania CAB.

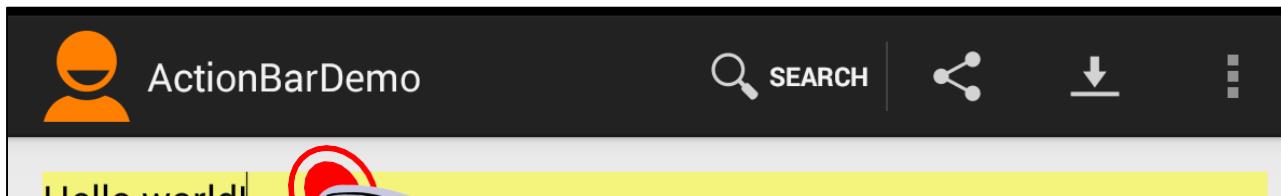


Przykład 6 – Kontekstowy ActionBar (CAB)

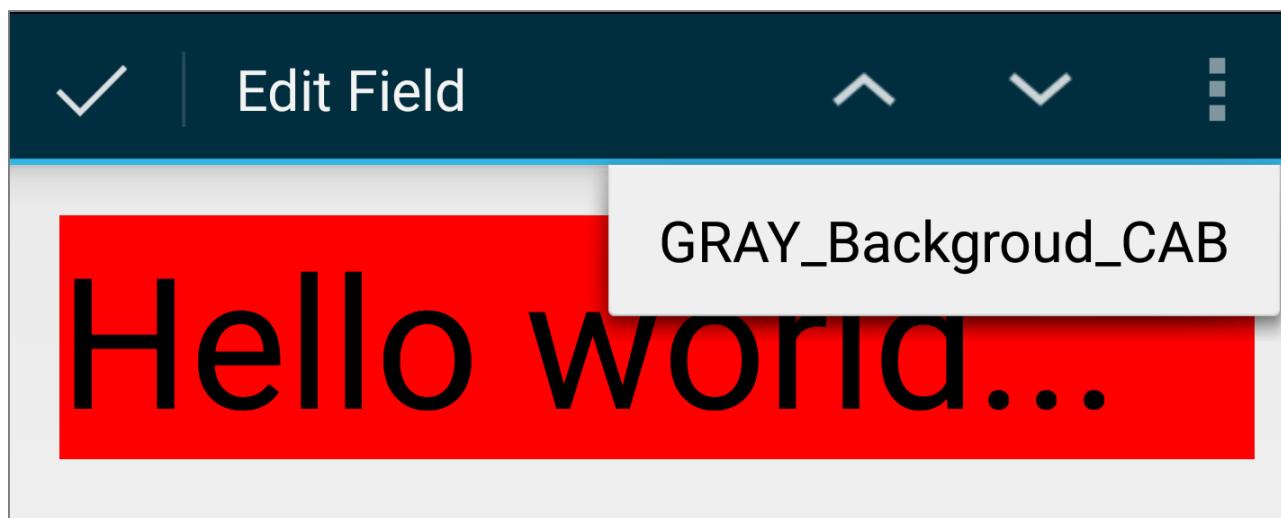


Można oczywiście tworzyć własne CABy. W tym celu programista musi stworzyć **tymczasowe menu** które zostanie nałożone na istniejący ActionBar. Należy również zaimplementować interfejs **ActionMode.Callback** by wskazać jak reagować na wybór konkretnej pozycji z CAB.

CAB jest usuwany w momencie w którym użytkownik odznaczy elementy GUI, naciśnie przycisk WSTECH, lub wybierze opcję *Done* znajdująca się po lewej stronie Actionbara.



← ActionBar

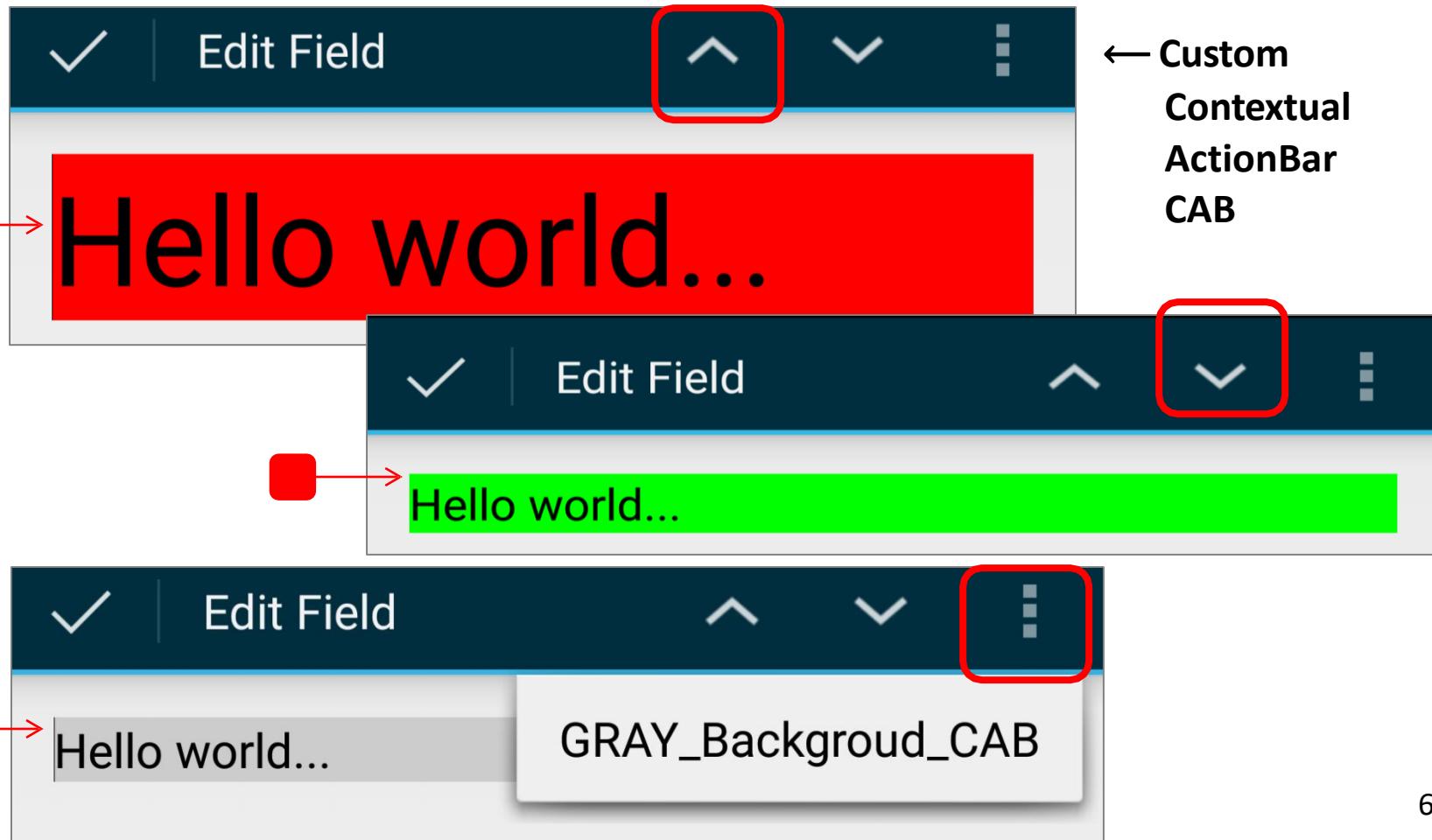


← Własny
CAB

Przykład 6 – Kontekstowy ActionBar (CAB)



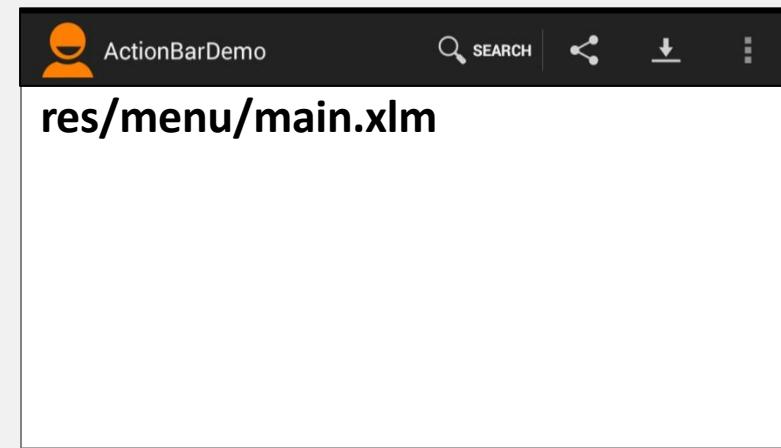
W prezentowanym przykładzie przytrzymanie palca na napisie “Hello world...” aktywuje własny CAB. CAB oferuje dwa kafelki oraz jedną opcję w menu rozwijanym. Pierwszy kafelek (Λ) zwiększa wielkość tekstu oraz zmienia tło na czerwone, drugi (V) zmniejsza rozmiar tekstu i ustawia tło na zielone, natomiast opcja w menu głównym zmienia kolor tła na szary.



Przykład 6 – Kontekstowy ActionBar (CAB)



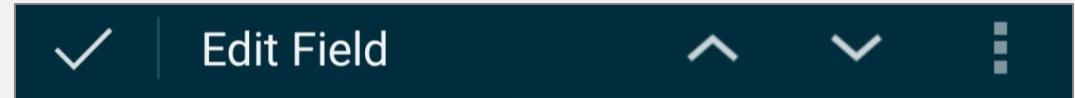
```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="csu.matos.MainActivity" >
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always/withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>
</menu>
```



Przykład 6 – Kontekstowy ActionBar (CAB)



```
<menu xmlns:android="http://schemas.android.com/apk/res/android"  
      xmlns:tools="http://schemas.android.com/tools"  
      tools:context="com.example.x3.MainActivity" >
```



```
<item  
    android:id="@+id/action_gray_background_cab"  
    android:orderInCategory="100"  
    android:showAsAction="never"  
    android:title="GRAY_Backgroud_CAB"/>
```

```
<item  
    android:id="@+id/action_increase"  
    android:orderInCategory="120"  
    android:icon="@drawable/ic_action_increase"  
    android:showAsAction="always/withText"  
    android:title="Increase"/>
```

```
<item  
    android:id="@+id/action_decrease"  
    android:orderInCategory="160"  
    android:icon="@drawable/ic_action_decrease"  
    android:showAsAction="always/withText"  
    android:title="Decrease"/>
```

```
</menu>
```

res/menu/ txtmsg_cab_menu.xml

Kontekstowy układ XML



MainActivity

```
public class MainActivity extends Activity {  
  
    EditText txtMsg;  
    public ActionMode actionmode;  
    private TxtMsgCallbacks txtMsgCallbacks;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        1→ txtMsg = (EditText) findViewById(R.id.txtMsg);  
        txtMsg.setOnLongClickListener(new OnLongClickListener() {  
  
            @Override  
            public boolean onLongClick(View v) {  
  
                2→ if (actionmode == null) {  
                    txtMsgCallbacks = new TxtMsgCallbacks(MainActivity.this, txtMsg);  
                } else {  
                    Toast.makeText(getApplicationContext(), "REUSING",  
                        Toast.LENGTH_LONG).show();  
                }  
                actionmode = startActionMode(txtMsgCallbacks);  
                actionmode.setTitle("Edit Field");  
                return true;  
            }  
        });  
    } //onCreate
```

Przykład 6 – Kontekstowy ActionBar (CAB)

2 z 2



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

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_search) {
        txtMsg.setText("Search...");
        return true;
    } else if (id == R.id.action_share) {
        txtMsg.setText("Share...");
        return true;
    } else if (id == R.id.action_download) {
        txtMsg.setText("Download...");
        return true;
    } else if (id == R.id.action_about) {
        txtMsg.setText("About...");
        return true;
    } else if (id == R.id.action_settings) {
        txtMsg.setText("Settings...");
        return true;
    }

    return false;
} //onOptionsItemSelected

} //MainActivity
```



Komentarze – MainActivity

1. Nasłuchiwaniec **LongClickListener** jest ustawiony dla pola tekstowego (typu `EditText`). Zadaniem nasłuchiwacza jest uaktywnienie własnego, kontekstowego paska `ActionBar`.
2. Gdy zdarzenie typu *LongClick* zostaje wykryte, obiekt **ActionMode** jest tworzony by zarządzać CAB. Tego typu obiekty są przydatne w sytuacjach, w których wymagana jest tymczasowa podmiana GUI i zmiana sposobu interakcji z użytkownikiem. W tym przykładzie obiekt `ActionMode` przykrywa globalny pasek `ActionBar`. Obiekt ten otrzymuje określony napis (“Edit Field”), referencję do widoku `EditText` oraz referencję do kontekstu aplikacji.
3. Dodatkowo obiekt **TxtMsgCallback** jest tworzony (bądź wykorzystywany podobnie jeśli nie jest to pierwszy raz) i zostaje powiązany z komponentem `ActionMode` by obsłużyć zdarzenia związane z interakcją użytkownika.



Klasa TxtMsgCallbacks

```
public class TxtMsgCallbacks implements ActionMode.Callback {  
    // This class handles the actions shown by the custom CAB  
    MainActivity mainContext;  
    TextView txtMsg;  
  
    public TxtMsgCallbacks(MainActivity mainContext, TextView txtMsg) {  
        // this is the EditText view controlled by the CAB  
        this.txtMsg = txtMsg;  
        this.mainContext = mainContext;  
    }  
  
    @Override  
    public boolean onActionItemClicked(ActionMode mode, MenuItem item) {  
        1 → //set value of 10px (regardless of screen density)  
        float tenPixels = TypedValue.applyDimension( TypedValue.COMPLEX_UNIT_DIP, 10,  
                                                    mainContext.getResources().getDisplayMetrics());  
        //detect current text-size  
        float oldSize = txtMsg.getTextSize();  
  
        2 → //increase by 10px text-size with red background  
        if (item.getItemId() == R.id.action_increase) {  
            txtMsg.setTextSize(TypedValue.COMPLEX_UNIT_PX, oldSize + tenPixels);  
            txtMsg.setBackgroundColor(Color.RED);  
  
        3 → //decrease by 10px text-size with green background  
        } else if (item.getItemId() == R.id.action_decrease) {  
            txtMsg.setTextSize(TypedValue.COMPLEX_UNIT_PX, oldSize - tenPixels);  
            txtMsg.setBackgroundColor(Color.GREEN);  
        }  
    }  
}
```



Klasa TxtMsgCallbacks

```
4    //set background to gray
} else if (item.getItemId() == R.id.action_gray_background_cab) {
    txtMsg.setBackgroundColor(Color.LTGRAY);
}

return false;
}

5 // showing other states from the ActionMode life-cycle
@Override
public boolean onCreateActionMode(ActionMode mode, Menu menu) {
    mode.getMenuInflater().inflate(R.menu.txtmsg_cab_menu, menu);
    return true;
}

@Override
public void onDestroyActionMode(ActionMode mode) {
    Toast.makeText(mContext, "Destroy CAB", Toast.LENGTH_LONG).show();
}

@Override
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
    Toast.makeText(mContext, "Prepare CAB", Toast.LENGTH_LONG).show();
    return false;
}
}
```



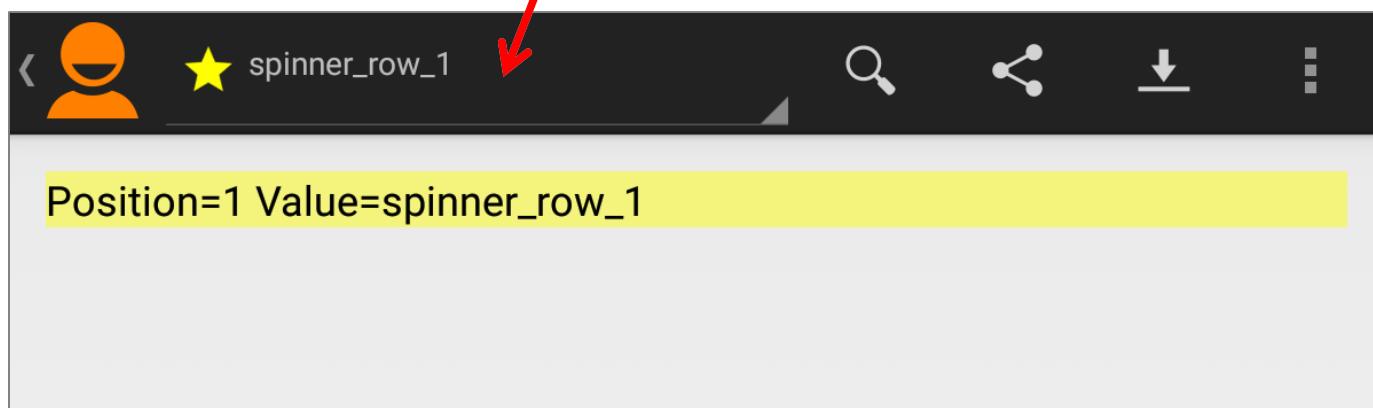
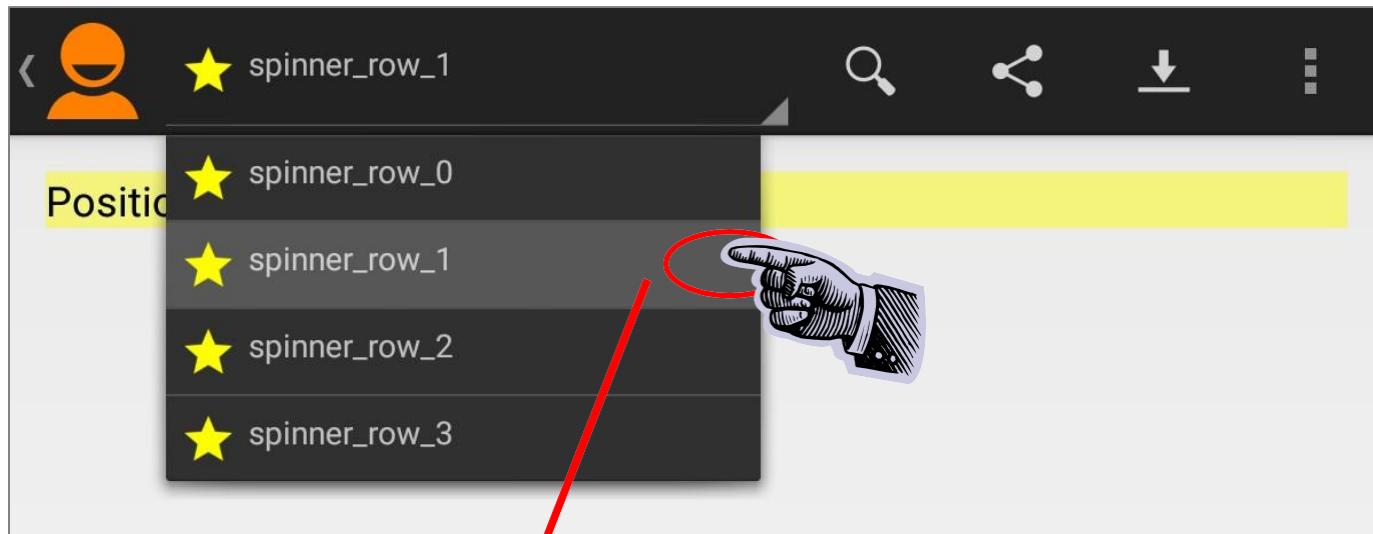
Komentarze – TxtMsgCallbacks

Klasa **ActionMode** jest odpowiedzialna za wizualizację własnego CAB. Klasa pomocnicza **TxtMsgCallbacks** tworzy obiekty na podstawie specyfikacji (**txtmsg_cab_menu.xml**) które zakrywają globalny komponent ActionBar oraz monitoruje kliknięcia przycisków wyświetlanych na CAB.

1. Metoda **onActionItemClicked()** jest wywoływana jeśli użytkownik kliknie wybrany kafelek. Jej pierwszy krok to określenie ile faktycznie zajmuje 10 pikseli na zadanym ekranie.
2. Po wyborze pierwszej akcji wielkość tekstu (txtMsg) jest zwiększana o 10 px, a kolor tła zmieniany jest na czerwony.
3. Po wyborze drugiej akcji wielkość tekstu (txtMsg) jest zmniejszana o 10 px, a kolor tła zmieniany jest na zielony.
4. Wybór trzeciej opcji skutkuje zmianą koloru tła na szary.
5. Komunikat typu Toast jest wyświetlany przy przejściu do innego stanu.

Przykład 7 – Spinner

W tym przykładzie komponent typu **Spinner** jest dodawany do ActionBar. Plik **res/menu/main.xml** jest taki sam jak w pierwszym przykładzie.



Przykład 7 – Spinner

Pierwszym krokiem jest zdefiniowanie układu XML zawierający komponent typu Spinner, który później zostanie umieszczony na pasku Actionbar:

custom_spinner_view_on_actionbar.xml

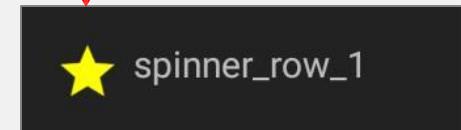
```
<Spinner  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/spinner_data_row"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```



Układ określający formatowanie poszczególnych pozycji spinnera.

custom_spinner_row_icon_caption.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent" android:layout_height="wrap_content"  
    android:orientation="horizontal"      android:padding="6dp" >  
    <ImageView  
        android:id="@+id/imgSpinnerRowIcon"  
        android:layout_width="25dp" android:layout_height="25dp"  
        android:layout_marginRight="5dp"  
        android:src="@drawable/ic_launcher" />  
    <TextView  
        android:id="@+id/txtSpinnerRowCaption"  
        android:layout_width="wrap_content" android:layout_height="wrap_content" />  
</LinearLayout>
```



Przykład 7 – Spinner

W metodzie **onCreate** pole rozwijane (Spinner) jest przypisywane do ActionBara. Ponadto tworzony jest DataAdapter i przypisywany nasłuchiwacz 'ItemSelected'.

```
@Override
protected void onResume() {
    super.onResume();
    actionBar = getActionBar(); // setup the ActionBar
    actionBar.setDisplayShowCustomEnabled(true); // allow custom views to be shown
    actionBar.setDisplayHomeAsUpEnabled(true); // show 'UP' affordance < button
    actionBar.setDisplayShowHomeEnabled(true); // allow app icon - logo to be shown
    actionBar.setHomeButtonEnabled(true); // needed for API.14 or greater

    // move the spinner to the actionBar as a CustomView
    actionBar.setCustomView(R.layout.custom_spinner_view_on_actionbar);

    // create the custom adapter to feed the spinner
    customSpinnerAdapter = new SpinnerCustomAdapter(
        getApplicationContext(),
        SpinnerDummyContent.customSpinnerList);

    // plumbing - get access to the spinner widget shown on the actionBar
    customSpinner = (Spinner)
        actionBar.getCustomView().findViewById(R.id.spinner_data_row);

    // bind spinner and adapter
    customSpinner.setAdapter(customSpinnerAdapter);

    // put a listener to wait for spinner rows to be selected
    customSpinner.setOnItemSelectedListener(this);
    customSpinner.setSelection(selectedSpinnerRow);

} // onResume
```

Przykład 7 – Spinner

1 z 2

SpinnerCustomAdapter. Jest to definicja własnego adaptera by wypełnić poszczególne wiersze Spiniera tekstem oraz grafiką na podstawie: **custom_spinner_row_icon_caption.xml**.

```
public class SpinnerCustomAdapter extends BaseAdapter {

    private ImageView spinnerRowIcon;
    private TextView spinnerRowCaption;
    private ArrayList<SpinnerRow> spinnerRows;
    private Context context;

    public SpinnerCustomAdapter(Context applicationContext,
                                ArrayList<SpinnerRow> customSpinnerList) {
        this.spinnerRows = customSpinnerList;
        this.context = applicationContext;
    }

    @Override
    public int getCount() { return
        spinnerRows.size();
    }

    @Override
    public Object getItem(int index) {
        return spinnerRows.get(index);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }
}
```

Przykład 7 – Spinner

2 z 2

SpinnerCustomAdapter. Jest to definicja własnego adaptera by wypełnić poszczególne wiersze Spinnera tekstem oraz grafiką na podstawie: `custom_spinner_row_icon_caption.xml`.

```
@Override
public View getView(final int position, View convertView, ViewGroup parent) {

    if (convertView == null) {
        LayoutInflater mInflater = (LayoutInflater) context.getSystemService(
                Activity.LAYOUT_INFLATER_SERVICE);
        convertView=mInflater.inflate(R.layout.custom_spinner_row_icon_caption, null);
    }

    spinnerRowIcon = (ImageView) convertView.findViewById(R.id.imgSpinnerRowIcon);
    spinnerRowCaption = (TextView) convertView.findViewById(
            R.id.txtSpinnerRowCaption);

    spinnerRowIcon.setImageResource(spinnerRows.get(position).getIcon());
    spinnerRowCaption.setText(spinnerRows.get(position).getCaption());

    convertView.setId(position);

    return convertView;
}

}
```

Przykład 7 – Spinner

1 z 2

Sztuczne dane źródłowe (**SpinnerDummyContent.java**). Kod służy wygenerowaniu danych pokazowych dla komponentu Spinner.

```
public class SpinnerDummyContent {  
  
    public static ArrayList<SpinnerRow> customSpinnerList = new  
        ArrayList<SpinnerRow>();  
  
    static {  
  
        // preparing spinner data (a set of [caption, icon] objects)  
        customSpinnerList.add( new SpinnerRow("spinner_row_0",  
            R.drawable.ic_spinner_row_icon));  
        customSpinnerList.add( new SpinnerRow("spinner_row_1",  
            R.drawable.ic_spinner_row_icon));  
        customSpinnerList.add( new SpinnerRow("spinner_row_2",  
            R.drawable.ic_spinner_row_icon));  
        customSpinnerList.add( new SpinnerRow("spinner_row_3",  
            R.drawable.ic_spinner_row_icon));  
  
    }  
}
```

Sztuczne dane źródłowe. Kod służy wygenerowaniu danych pokazowych dla komponentu Spinner.

```
public static class SpinnerRow { // each row consists of [caption, icon]
    private String caption;
    private int icon;

    public SpinnerRow(String caption, int icon) {
        this.caption = caption;  this.icon = icon;
    }

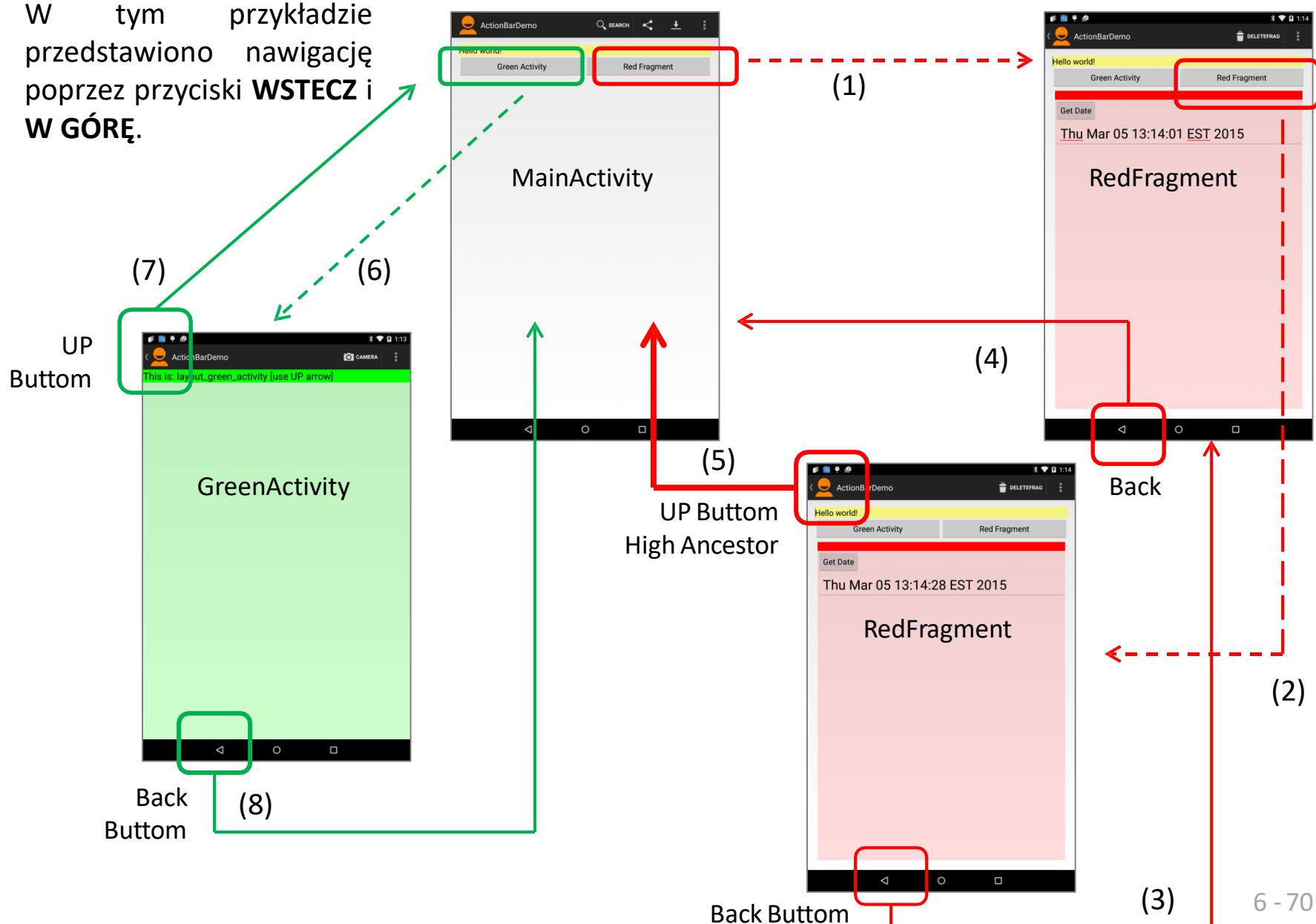
    public String getCaption() {
        return this.caption;
    }

    public int getIcon() {
        return this.icon;
    }

    @Override
    public String toString() {
        return caption;
    }
} //SpinnerRow
```

Przykład 8 – Nawigacja

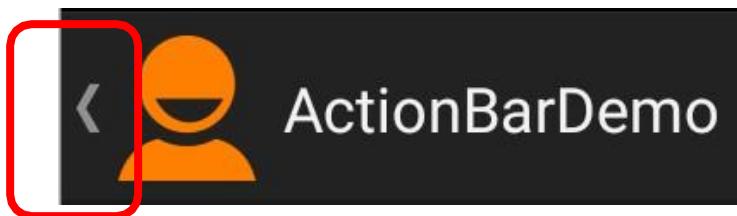
W tym przykładzie przedstawiono nawigację poprzez przyciski **WSTECZ** i **W GÓRĘ**.



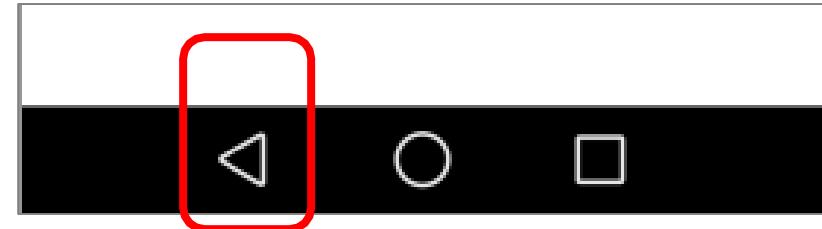
Przykład 8 – Nawigacja

W tym przykładzie zakłada się, że interfejs **MainActivity** może zostać tymczasowo przysłonięty przez inną aktywność (**GreenActivity**) lub może być jedynie częściowo przysłonięty przez fragmenty typu **RedFragments**.

Nawigacja między wymienionymi elementami odbywać się będzie przez wybór klawisza **WSTECZ** lub **W GÓRĘ**.



W GÓRĘ (UP-KEY)



WSTECZ (BACK-KEY)

{Przycisk **W GÓRĘ** używany jest by przekierować widok na dowolny inny element hierarchii. Przycisk **WSTECZ** zapewnia nawigację o jeden element wcześniej zgodnie z ich kolejnością wyświetlania.

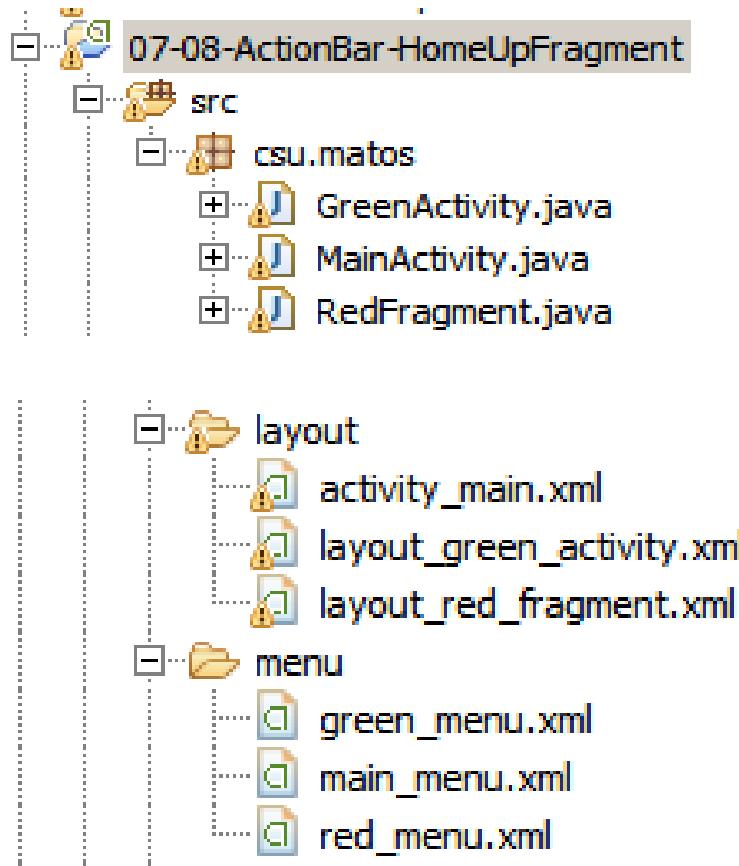
Przykład 8 – Nawigacja

IDEA DZIAŁANIA PRZYKŁADU

1. Po kliknięciu na przycisk ‘Red Fragment’ główna aktywność pokazuje fragment RedFragment (Można wówczas kliknąć przycisk ‘Get Date’ by zmienić stan i dane pokazywane przez fragment).
2. Drugi fragment (redFragment) jest tworzony i umieszczany nad pierwszym.
3. Kliknięcie przycisku WSTE CZ usuwa aktualny widok (drugi fragment) zmieniając stan aplikacji by pokazywała tylko jeden fragment.
4. Kliknięcie przycisku WSTE CZ jeszcze raz spowoduje powrót do głównej aktywności.
5. Kliknięcie przycisku W GÓRĘ powoduje od razu przejście do poprzedniego elementu w hierarchii widoków (w tym przypadku jest to główna aktywność).
6. Wykorzystywana jest intencja by przywołać GreenActivity, która staje się aktywną i widoczną..
7. Plik AndroidManifest posiada wpis mówiący, że MainActivity jest rodzicem dla GreenActivity. Zatem wykorzystanie przycisku W GÓRĘ powoduje przejście do aktywności rodzica (wyżej w hierarchii).

Przykład 8 – Nawigacja

STRUKTURA APLIKACJI



Układy dla MainActivity,
GreenActivity i RedFragment

Każdy z tych komponentów będzie miało swoje menu wizualizowane na ActionBar

Przykład 8 – UKŁAD: activity_main.xml

1 z 2

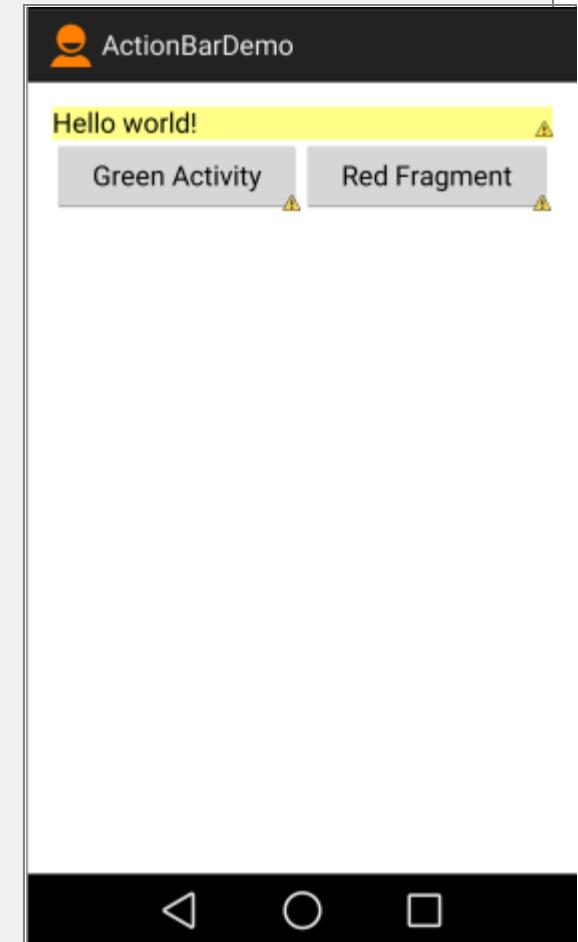
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="csu.matos.MainActivity" >

    <EditText
        android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#77ffff00"
        android:text="@string/hello_world" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id	btnGreenActivity"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Green Activity" />

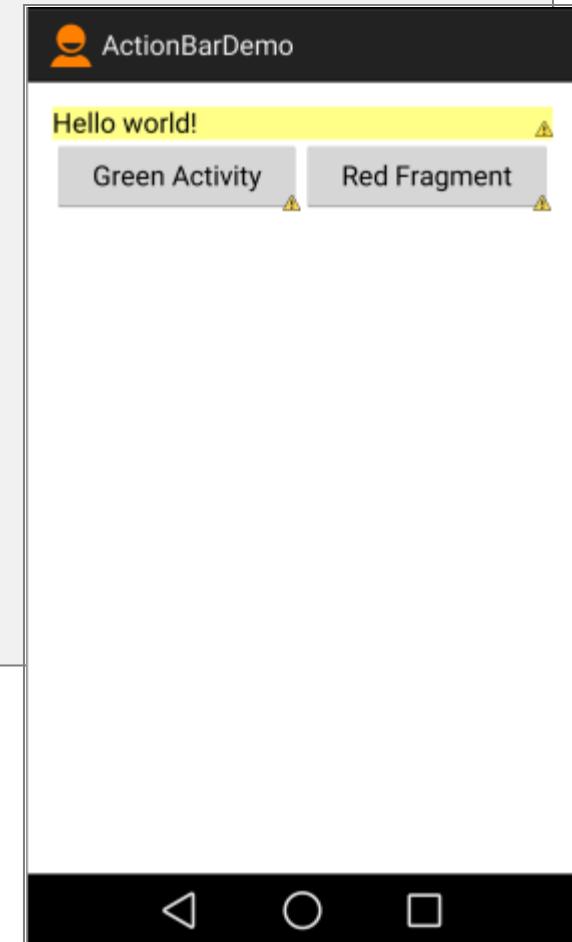
        <Button
            android:id="@+id	btnRedFragment"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Red Fragment" />
    
```



Przykład 8 – UKŁAD: activity_main.xml

2 z 2

```
<Button  
    android:id="@+id/btnRedFragment"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="Red Fragment" />  
</LinearLayout>  
  
<FrameLayout  
    android:id="@+id/frag_container_inside_activity_main"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:layout_margin="6dp"  
    android:layout_weight="2" />  
  
</LinearLayout>
```

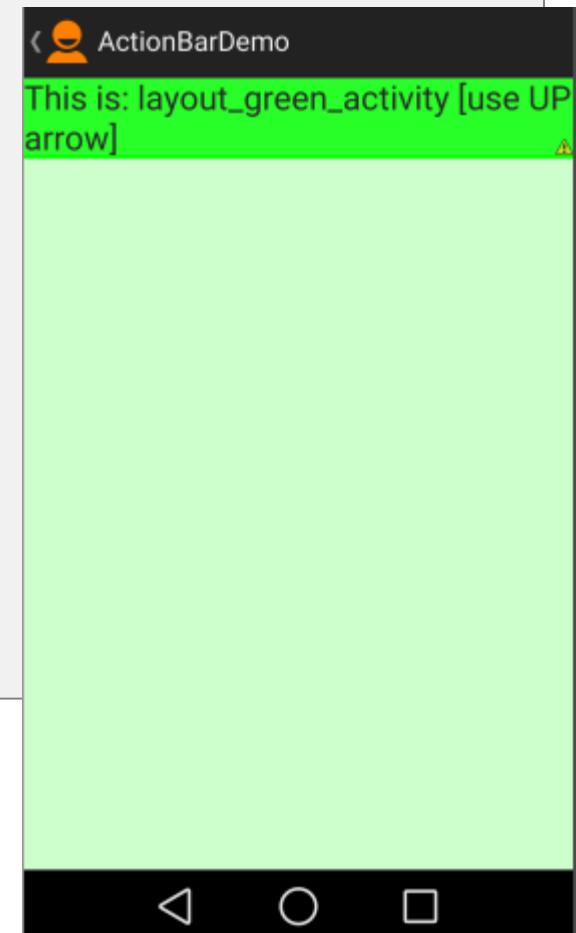


Przykład 8 – UKŁAD: layout_green_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#3300ff00"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textViewGreen1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff00ff00"
        android:text=
            "This is: layout_green_activity [use UP arrow]"
        android:textAppearance=
            "?android:attr/textAppearanceLarge" />

</LinearLayout>
```



Przykład 8 – UKŁAD: layout_red_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#22ff0000"
    android:layout_margin="6dp"
    android:orientation="vertical" >

    <View
        android:background="#ffff0000"
        android:layout_width="match_parent"
        android:layout_height="20dp" />

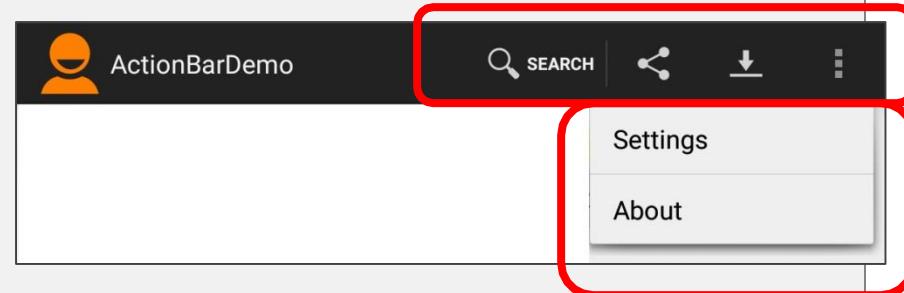
    <Button
        android:id="@+id/btn_date_red_fragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Get Date" />

    <EditText
        android:id="@+id/txtMsgFragmentRed"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:ems="10" />
```



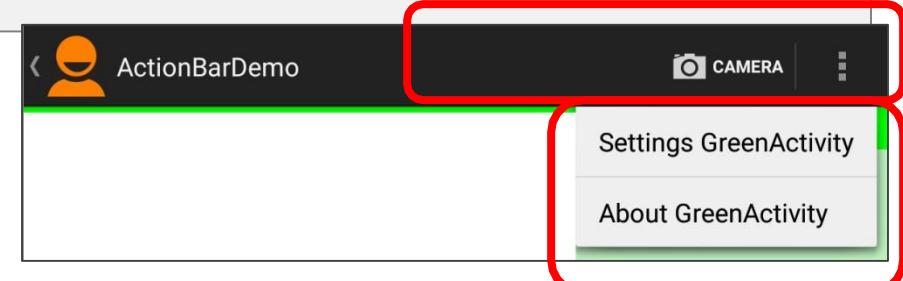
Przykład 8 – MENU: main_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>
</menu>
```



Przykład 8 – MENU: green_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_camera_green_activity"
        android:icon="@drawable/ic_action_camera"
        android:orderInCategory="100"
        android:showAsAction="ifRoom|withText"
        android:title="Camera"/>
    <item
        android:id="@+id/action_settings_green_activity"
        android:icon="@drawable/ic_launcher"
        android:orderInCategory="120"
        android:showAsAction="never"
        android:title="Settings GreenActivity"/>
    <item
        android:id="@+id/action_about_green_activity"
        android:icon="@drawable/ic_launcher"
        android:orderInCategory="140"
        android:showAsAction="never"
        android:title="About GreenActivity"/>
</menu>
```



Przykład 8 – MENU: red_menu.xml

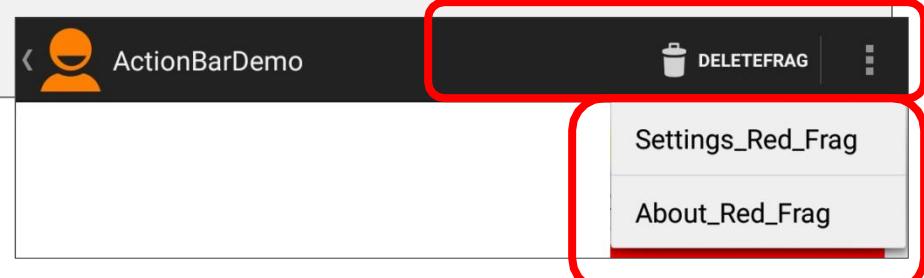
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/action_delete_red_frag"
        android:icon="@drawable/ic_action_delete"
        android:orderInCategory="100"
        android:showAsAction="ifRoom|withText"
        android:title="DeleteFrag"/>

    <item
        android:id="@+id/settings_red_frag"
        android:orderInCategory="120"
        android:showAsAction="never"
        android:title="Settings_Red_Frag"/>

    <item
        android:id="@+id/about_red_frag"
        android:orderInCategory="140"
        android:showAsAction="never"
        android:title="About_Red_Frag"/>

</menu>
```



Przykład 8 – AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="csu.matos" android:versionCode="1" android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="16"
        android:targetSdkVersion="21" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:windowSoftInputMode="stateHidden" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name=".GreenActivity"
            android:label="@string/app_name"
            android:parentActivityName="MainActivity" >
        </activity>

    </application>
</manifest>
```

Przykład 8 – MainActivity.java

1 z 3

```
public class MainActivity extends Activity implements OnClickListener {  
  
    EditText txtMsg;  
    Button btnGreenActivity;  
    Button btnRedFragment;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        txtMsg = (EditText) findViewById(R.id.txtMsg);  
        btnGreenActivity = (Button) findViewById(R.id.btnGreenActivity);  
        btnRedFragment = (Button) findViewById(R.id.btnRedFragment);  
        btnGreenActivity.setOnClickListener(this);  
        btnRedFragment.setOnClickListener(this);  
  
    }  
  
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        // Inflate the menu; add items to the action bar  
        getMenuInflater().inflate(R.menu.main_menu, menu);  
        return true;  
    }  
}
```

Przykład 8 – MainActivity.java

2 z 3

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    1    // Handle action bar item clicks here.
    int id = item.getItemId();

    if (id == R.id.action_search) {
        txtMsg.setText("Search...");
        return true;
    } else if (id == R.id.action_share) {
        txtMsg.setText("Share...");
        return true;
    } else if (id == R.id.action_download) {
        txtMsg.setText("Download...");
        return true;
    } else if (id == R.id.action_about) {
        txtMsg.setText("About...");
        return true;
    } else if (id == R.id.action_settings) {
        txtMsg.setText("Settings...");
        return true;
    }
    return false;
}
```

Przykład 8 – MainActivity.java

3 z 3

```
// "GreenActivity" button used to invoke a supporting Activity called via
// Intent while "RedFragment" button replaces the MainActivity's UI with
// a new instance of a "FragmentRed"
@Override
public void onClick(View v) {

    if (v.getId() == R.id.btnGreenActivity) {
        Intent greenActivityIntent = new Intent(MainActivity.this,
                                                GreenActivity.class);
        // if needed put data items inside the bundle
        Bundle datainfo = new Bundle();
        greenActivityIntent.putExtra("data", datainfo);
        startActivityForResult(greenActivityIntent, 0);
    } else

        if( v.getId() == R.id.btnRedFragment ){
            // create a new RED fragment - show it
            FragmentTransaction ft = getFragmentManager().beginTransaction();
            RedFragment fragmentRed = RedFragment.newInstance("new-red-frag-arg1");
            ft.replace(R.id.frag_container_inside_activity_main,fragmentRed,"red_frag");
            ft.addToBackStack("red_tran"); //allows BackButton pop-navigation
            ft.commit();
        }

    }//onClick

}//MainActivity
```

2

3

Przykład 8 – MainActivity.java

Komentarz

1. W tym przykładzie każda aktywność i fragment posiada indywidualne menu. Aktywność główna tworzy ActionBar na podstawie specyfikacji **main_menu.xml**. Menu zawiera przede wszystkim opcje do ściągania, udostępniania i wyszukiwania.
2. Gdy użytkownik kliknie na przycisk ‘Green Activity’ wysyłana jest intencja by wywołać ekran GreenActivity. Wraz z intencją mogą być przesyłane dane dodatkowe. Metoda ‘*startActivityForResult(...)*’ umożliwia monitorowanie wartości zwróconych przez Green Activity.
3. Kliknięcie na przycisk ‘Red Fragment’ powoduje stworzenie nowej instancji klasy RedFragment. Metoda *.replace(...)* usuwa jakikolwiek poprzedni widok znajdujący się w określonym miejscu MainActivity na ten nowostworzony. Informacje o przeprowadzonej transakcji dodawane są do **BackStack** w celu łatwiejszego przywrócenia poprzedniego stanu aplikacji.

Przykład 8 – GreenActivity.java

1 z 2

```
public class GreenActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.layout_green_activity);  
  
        // enable the home button  
        getActionBar().setDisplayHomeAsUpEnabled(true);  
        getActionBar().setHomeButtonEnabled(true);  
  
    } //onCreate  
  
  
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        getMenuInflater().inflate(R.menu.green_menu, menu);  
        return true;  
    }  
}
```

Przykład 8 – GreenActivity.java

2 z 2

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    switch (item.getItemId()) {

        case android.R.id.home:
            // the user pressed the UP button - where to go now? look for
            // Manifest's entry: android:parentActivityName="MainActivity"
            if (getParentActivityIntent() == null) {
                Log.i("ActivityGreen",
                    "Fix Manifest to indicate the parentActivityName");
                onBackPressed(); //terminate the app

            } else {
                NavUtils.navigateUpFromSameTask(this); //back to parent activity
            }
            return true;

        default:
            return super.onOptionsItemSelected(item);
    }
}
```

1 →

Przykład 8 – GreenActivity.java

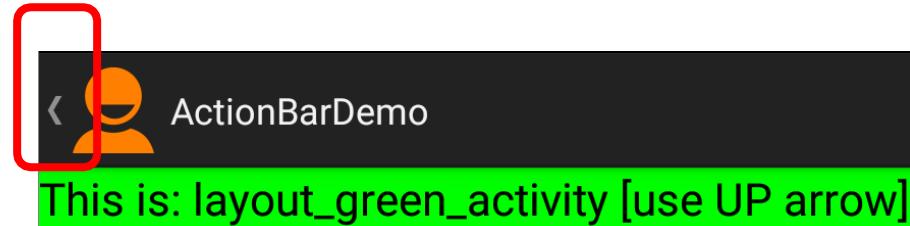
Komentarz

1. Powiązanie hierarchii aktywności odbywa się poprzez odpowiednią zmianę w pliku AndroidManifest.xml:

...

```
<activity
    android:name=".GreenActivity"
    android:label="@string/app_name"
    android:parentActivityName="MainActivity" >
</activity>
```

...



Pozycja menu `android.R.id.home` reprezentuje kliknięcie przycisku W GÓRĘ. Pierwszym krokiem jest sprawdzenie, czy plik manifestu zawiera odwołanie do aktywności nadzędnej. Jeżeli takie odwołanie nie istnieje, wywoływana jest metoda `onBackPressed()` by cofnąć się w hierarchii widoków. W przeciwnym przypadku, wykorzystywana jest klasa **NavUtils** by przejść do wskazanego miejsca w hierarchii.

Przykład 8 – RedFragment.java

1 z 4

```
public class RedFragment extends Fragment {  
  
    TextView txtMsgFragmentRed = null;  
  
    public static RedFragment newInstance(String strArg) {  
        RedFragment fragmentRed = new RedFragment();  
        Bundle args = new Bundle();  
        args.putString("strArg1", strArg);  
        fragmentRed.setArguments(args);  
        return fragmentRed;  
    }  
  
    @Override  
    public void onCreate(Bundle arg0) {  
        super.onCreate(arg0);  
        setHasOptionsMenu(true);  
  
         1 // enable the home button  
        getActivity().getActionBar().setDisplayHomeAsUpEnabled(true);  
        getActivity().getActionBar().setHomeButtonEnabled(true);  
    }  
  
    @Override  
    public View onCreateView(LayoutInflater inflater,  
                            ViewGroup container,  
                            Bundle savedInstanceState) {  
  
        // inflate layout_blue.xml holding a TextView and a ListView  
        LinearLayout redLayout = (LinearLayout) inflater.inflate(  
            R.layout.layout_red_fragment, null );  
    }
```

Przykład 8 – RedFragment.java

2 z 4

```
// plumbing - get a reference to textView and listView
txtMsgFragmentRed = (TextView) redLayout.findViewById(R.id.txtMsgFragmentRed);

final Button button = (Button)redLayout.findViewById(R.id.btn_date_red_fragment);
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        String text = (new Date()).toString();
        txtMsgFragmentRed.setText(text);
    }
});
return redLayout;
}

@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    menu.clear();
    inflater.inflate(R.menu.red_menu, menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here.
    int id = item.getItemId();

    if (id == android.R.id.home) {
        clearBackStack();                  //TRY: jump to MainActivity (HigAncestor)
        //showPreviousRedScreen(); //TRY: same as pressing Back button
    }else {
```

2

Przykład 8 – RedFragment.java

3 z 4

```
//for now -just show the action-id
    txtMsgFragmentRed.setText("ACTION_ID="+id);
}

return super.onOptionsItemSelected(item);
}

private void clearBackStack() {
    try {
        FragmentTransaction ft = getFragmentManager().beginTransaction();

        android.app.FragmentManager fragmentManager = getFragmentManager();

 →
        fragmentManager.popBackStackImmediate(null,
            FragmentManager.POP_BACK_STACK_INCLUSIVE);

        ft.commit();

    } catch (Exception e) {
        Log.e("CLEAR-STACK>>> ", e.getMessage() );
    }
}

//clearBackStack()
```

Przykład 8 – RedFragment.java

4 z 4

```
private void showPreviousRedScreen() {  
    try {  
        FragmentTransaction ft = getFragmentManager().beginTransaction();  
        android.app.FragmentManager fragmentManager = getFragmentManager();  
  
        //determine the size n of the BackStack (0,1,..n-1)  
        int bsCount = fragmentManager.getBackStackEntryCount();  
  
        //see (without removing) the stack's top entry  
        BackStackEntry topEntry = fragmentManager.getBackStackEntryAt(bsCount-1);  
  
        //obtain the numeric ID and name tag of the top entry  
        String tag = topEntry.getName();  
        int id = topEntry.getId();  
        Log.e("RED Top Fragment name: ", "" + tag + " " + id);  
  
        //pop the top entry (until matching id) and reset UI with its state data  
        //fragmentManager.popBackStackImmediate(id, 1);  
        fragmentManager.popBackStackImmediate();  
  
        ft.commit();  
  
    } catch (Exception e) {  
        Log.e("REMOVE>>> ", e.getMessage());  
    }  
  
}//showPreviousRedScreen  
}
```

Komentarz

1. Po przysłonięciu aktywności przez RedFragment możliwe jest wykorzystanie przycisku W GÓRĘ.
2. Prezentowany przykład umożliwia dwa typy nawigacji: historyczną oraz względem przodka.
3. **Nawigacja względem przodka.** Metoda ***clearBackStack()*** jest wywoływana by usunąć wszystkie wpisy w stosie BackStack dotyczące RedFragment. Możliwy jest powrót do „czystej” aktywności.

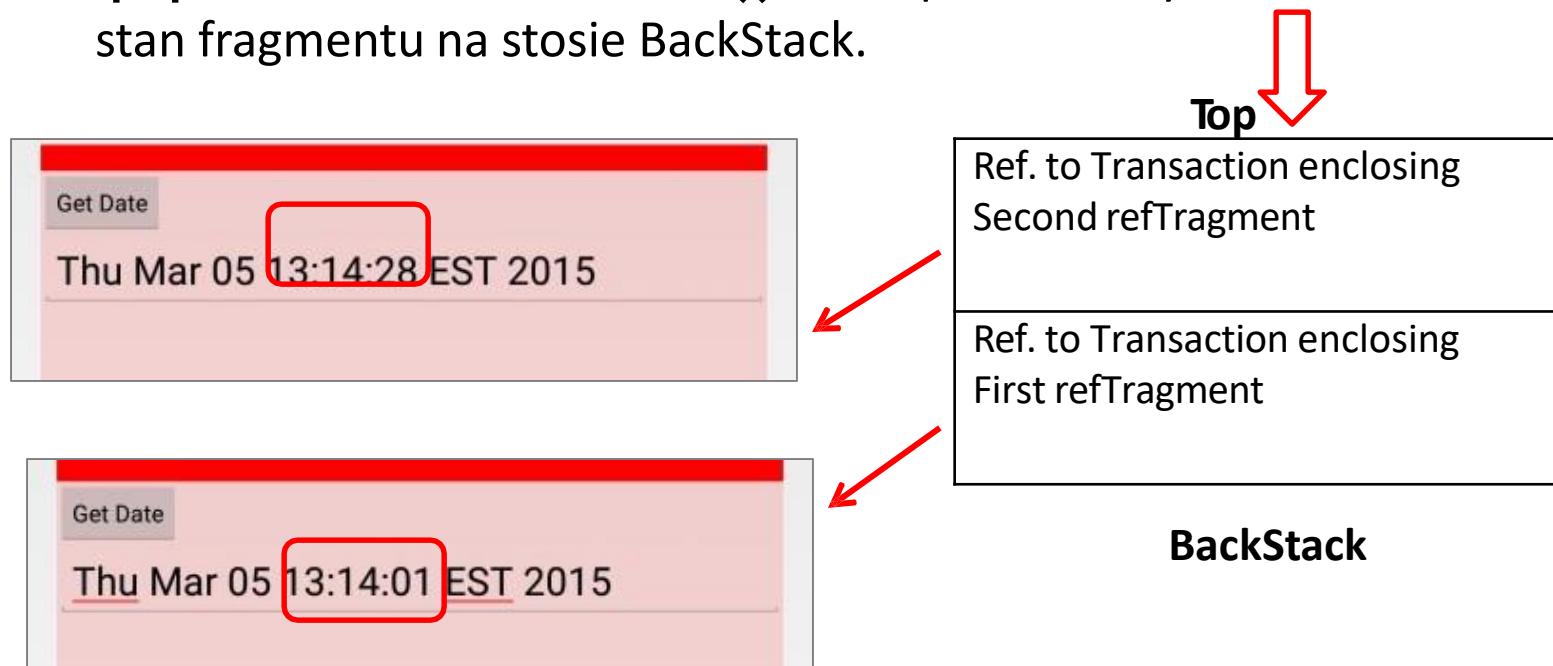
Wywołanie ***.popBackStackImmediate(tag, flag)*** wydobywa wszystkie wpisy z BackStack aż do napotkania określonego tag'a.

Jeżeli ustawiono flagę **POP_BACK_STACK_INCLUSIVE** wszystkie elementy zostaną wydobyte ze stosu aż do trafienia na poszukiwany. W przeciwnym przypadku wszystkie wpisy aż do poszukiwanego (bez niego) będą usunięte.

Komentarz

4. **Nawigacja historyczna.** Metoda własna `showPreviousRedScreen()` pokazuje jak można wymusić zachowanie tożsame jak przy wciśnięciu przycisku W GÓRĘ.

Tranzycja do poprzedzającego obiektu RedFragment (jeżeli jakikolwiek jest dostępny) odbywa się poprzez wywołanie metody `.popBackStackImmediate()` która pobiera i wyświetla ostatni stan fragmentu na stosie BackStack.

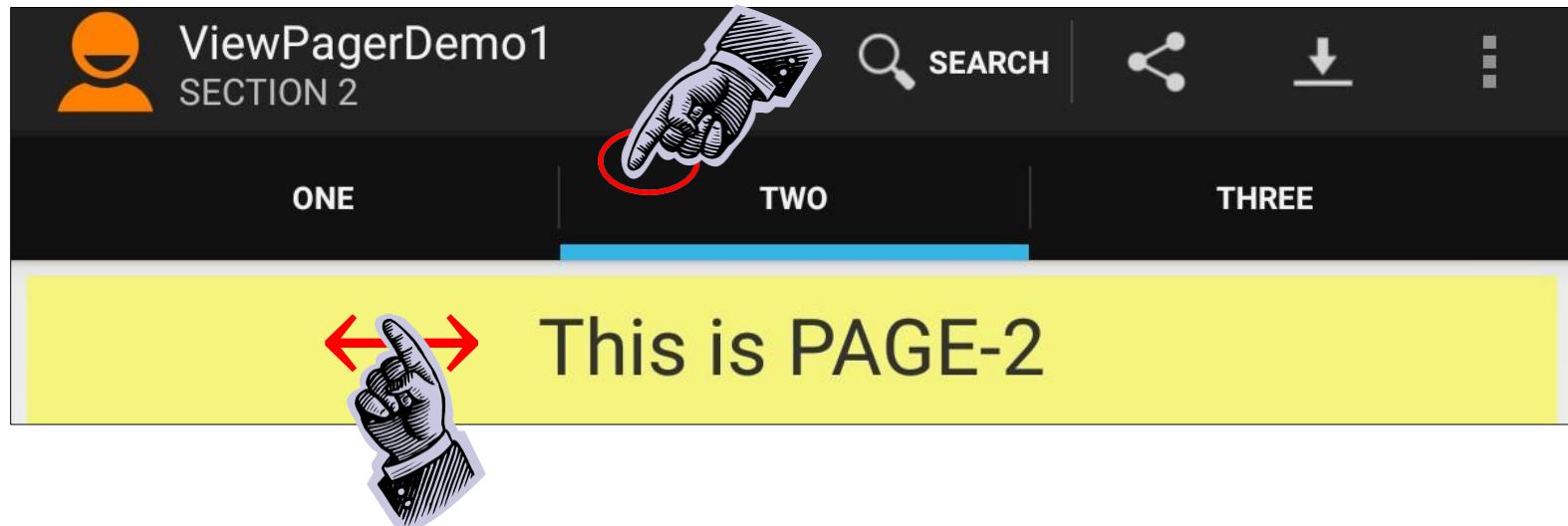


Przykład 9 – Wykorzystanie zakładek: ViewPager

Przesunięcie palcem zakładek jest przykładem tzw. **nawigacji bocznej** w której nowe ekranы pokazywane są na ekranie przez ich przeciągnięcie z końca obszaru roboczego. Widżet **ViewPager** jest jednym z komponentów, który umożliwia praktyczną implementację przesuwania zakładek w poziomie.

Strony zakładek – zwykle reprezentowane przez fragmenty – są generowane przez własny **PagerAdapter**.

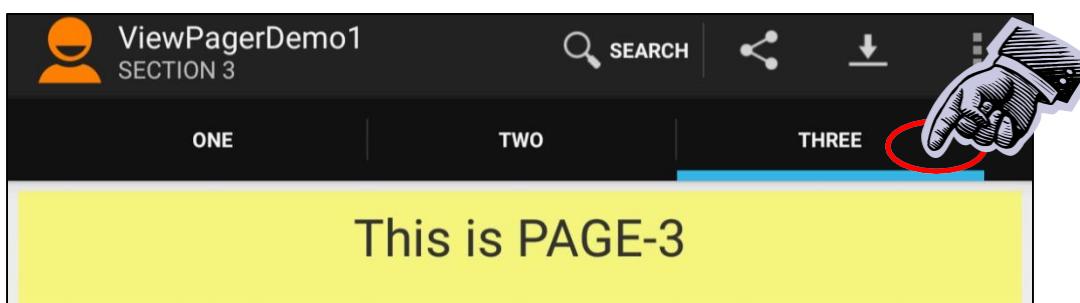
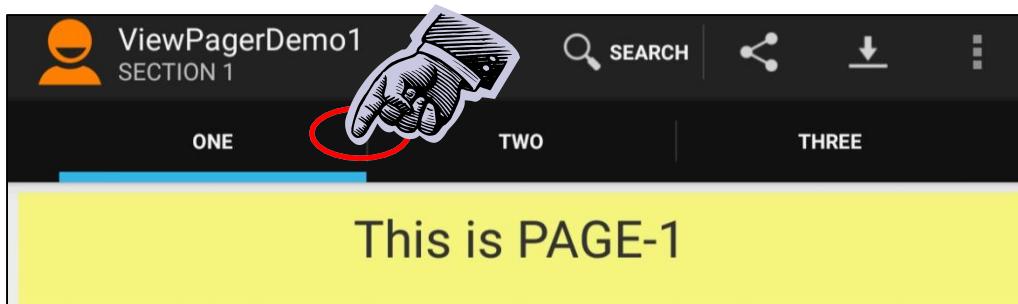
Funkcjonalność ViewPager jest często powiązany z komponentem **ActionBar** lub **TabStrip** by zapewnić nawigację opartą na zakładkach.



Przykład 9 – Wykorzystanie zakładek: ViewPager

W tym przykładzie komponent **ViewPager** zostanie powiązany z elementem **ActionBar** tworzonej aplikacji.

Jednakże od API-21 takie rozwiązanie ma status **deprecated** i zostanie wykorzystane jedynie w celu prezentacji ogólnej idei.

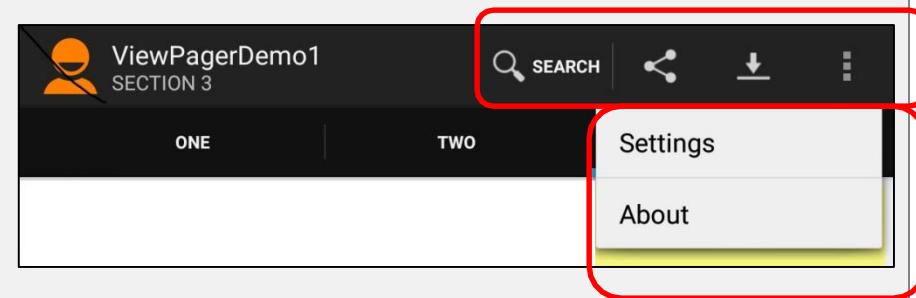


Zakładki są częścią **ActionBar**.

Kliknięcie na daną zakładkę powoduje stworzenie animacji symulującej poziomy ruch przewijania.

Przykład 9 – MENU: main_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>
</menu>
```



Przykład 9 – UKŁADY XML

activity_main.xml

```
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="6dp"
    tools:context="csu.matos.MainActivity" />
```

fragment_page_layout.xml

```
<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:padding="6dp"
    android:background="#77ffff00"
    tools:context="csu.matos.MainActivity$PlaceholderFragment" >

    <TextView
        android:id="@+id/section_label"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Hola"
        android:textSize="30sp" />

</LinearLayout>
```

Przykład 9 – MainActivity.java

1 z 6

```
public class MainActivity extends Activity implements TabListener {  
    SectionsPagerAdapter mSectionsPagerAdapter;  
    ViewPager mViewPager;  
    ActionBar actionBar;  
    Context context;  
    int duration = Toast.LENGTH_SHORT;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        context = MainActivity.this;  
  
        // adapter returns fragments representing pages added to the ViewPager  
        1 → mSectionsPagerAdapter = new SectionsPagerAdapter(getFragmentManager());  
        mViewPager = (ViewPager) findViewById(R.id.pager);  
        mViewPager.setAdapter(mSectionsPagerAdapter);  
  
        actionBar = getActionBar();  
        2 → actionBar.addTab(actionBar.newTab().setText("ONE").setTabListener(this));  
        actionBar.addTab(actionBar.newTab().setText("TWO").setTabListener(this));  
        actionBar.addTab(actionBar.newTab().setText("THREE").setTabListener(this));  
        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);  
  
        actionBar.setDisplayShowHomeEnabled(true);  
        actionBar.setDisplayShowTitleEnabled(true);
```

Przykład 9 – MainActivity.java

2 z 6

```
3 → mViewPager.setOnPageChangeListener(new OnPageChangeListener() {  
    //this listener reacts to the changing of pages  
    @Override  
    public void onPageSelected(int position) {  
        actionBar.setSelectedNavigationItem(position);  
        actionBar.setSubtitle(mSectionsPagerAdapter.getPageTitle(position));  
    }  
  
    @Override  
    public void onPageScrolled(int position, float positionOffset,  
                               int positionOffsetPixels) {  
    }  
  
    @Override  
    public void onPageScrollStateChanged(int state) {  
    }  
});  
}//onCreate  
  
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu; this adds items to the action bar if it is present.  
    getMenuInflater().inflate(R.menu.main, menu);  
    return true;  
}
```

Przykład 9 – MainActivity.java

3 z 6

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here.
    int id = item.getItemId();

    4 if (id == R.id.action_search) {
        Toast.makeText(context, "Search...", duration).show();
        return true;
    }
    else if (id == R.id.action_share) {
        Toast.makeText(context, "Share...", duration).show();
        return true;
    }
    else if (id == R.id.action_download) {
        Toast.makeText(context, "Download...", duration).show();
        return true;
    }
    else if (id == R.id.action_about) {
        Toast.makeText(context, "About...", duration).show();
        return true;
    }
    else if (id == R.id.action_settings) {
        Toast.makeText(context, "Settings...", duration).show();
        return true;
    }
    return false;
}
```

Przykład 9 – MainActivity.java

4 z 6

```
public class SectionsPagerAdapter extends FragmentPagerAdapter {  
  
    public SectionsPagerAdapter(FragmentManager fm) {  
        super(fm);  
    }  
  
    @Override  
    public Fragment getItem(int position) {  
        return PlaceholderFragment.newInstance(position + 1); //return a fragment  
    }  
  
    @Override  
    public int getCount() {  
        return 3; // Show 3 total pages.  
    }  
  
    @Override  
    public CharSequence getPageTitle(int position) {  
        Locale locale = Locale.getDefault();  
        switch (position) {  
            case 0: return getString(R.string.title_section1).toUpperCase(locale);  
            case 1: return getString(R.string.title_section2).toUpperCase(locale);  
            case 2: return getString(R.string.title_section3).toUpperCase(locale);  
        }  
        return null;  
    }  
}//SectionsPagerAdapter
```

5



Przykład 9 – MainActivity.java

5 z 6

```
public static class PlaceholderFragment extends Fragment {  
  
    private static final String ARG_SECTION_NUMBER = "section_number";  
  
    public static PlaceholderFragment newInstance(int sectionNumber)  
    {  
        PlaceholderFragment fragment = new PlaceholderFragment();  
        Bundle args = new Bundle();  
        args.putInt(ARG_SECTION_NUMBER, sectionNumber);  
        fragment.setArguments(args);  
        return fragment;  
    }  
    public PlaceholderFragment() { }  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                            Bundle savedInstanceState) {  
        int pagePosition = getArguments().getInt(ARG_SECTION_NUMBER, -1);  
        View rootView = inflater.inflate(R.layout.fragment_main, container,  
                                         false); TextView txtMsg = (TextView)  
        rootView.findViewById(R.id.section_label); String text = "This is PAGE-" +  
        pagePosition;  
        txtMsg.setText(text);  
        return rootView;  
    }  
}//PlaceholderFragment
```

6

7

Przykład 9 – MainActivity.java

6 z 6

```
// Implementing ActionBar TAB listener
@Override
public void onTabSelected(Tab tab, FragmentTransaction ft) {
    // move to page selected by clicking a TAB
    mViewPager.setCurrentItem(tab.getPosition());
}

@Override
public void onTabUnselected(Tab tab, FragmentTransaction ft) {
    // TODO: nothing - needed by the interface
}

@Override
public void onTabReselected(Tab tab, FragmentTransaction ft) {
    // TODO: nothing - needed by the interface
}

}
```

8



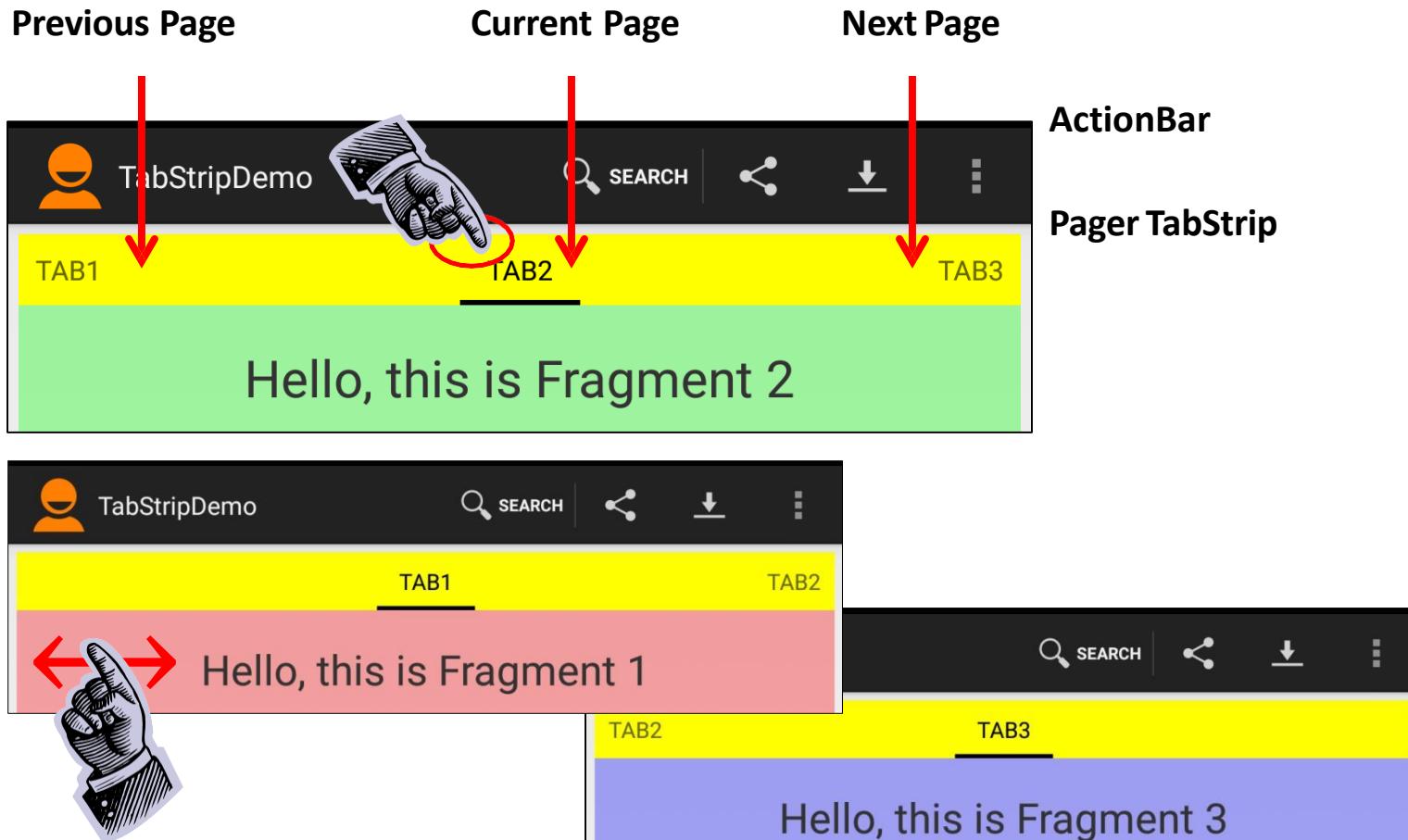
Przykład 9 – Wykorzystanie zakładek: ViewPager

Komentarz

1. Widżet ViewPager (zdefiniowany w układzie **activity_main.xml** jako **pager**), będzie zajmował całą przestrzeń roboczą. Własny **SectionsPagerAdapter** musi zostać zdefiniowany by pokazać osobne strony w ramach kontenera **pager**.
2. Po uzyskaniu dostępu do **ActionBar'a**, dodawane są trzy zakładki o nazwach “ONE”, “TWO” i “THREE”. Klauzula **setNavigationMode()** jest wykorzystana do aktywacji nawigacji po zakładkach (zakładki nie są pokazywane w przypadku pominięcia tej klauzuli).
3. Nasłuchiwanie **onPageChangeListener** reaguje na zmianę zakładek. Wówczas zmieniana jest również prezentowana treść zgodnie z wybraną zakładką.
4. Prócz zakładek **ActionBar** prezentuje również wszystkie komponenty zdefiniowane w pliku **res/menu/main.xml**. Metoda **onOptionsItemSelected** odpowiedzialna jest za wykonanie akcji powiązanej z daną pozycją menu.
5. Przy każdej zmianie zakładki nowy fragment jest tworzony by wypełnić aktualny widok. Metoda **getItem()** zwraca fragment dla odpowiedniej wybranej pozycji.

Przykład 10 – ViewPager oraz TabStrip

Widżet **PagerTabStrip** jest poziomym paskiem mogącym prezentować do 3 zakładek. Zakładki powiązane są z *aktualną*, *następną* i *poprzednią* stroną komponentu ViewPager. Każda zakładka składa się z *napisu* i *wskaźnika wyboru* (*podkreślenia*). Wskaźnik aktualnie wybranej zakładki znajduje się pod jej nazwą.



Przykład 10 – ViewPager oraz TabStrip

UKŁAD: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="6dp" >

    <android.support.v4.view.PagerTabStrip
        android:id="@+id/pager_tab_strip"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:background="#FFFFFF00"
        android:padding="10dp" />

</android.support.v4.view.ViewPager>
```

1

2

- Komponenty **ViewPager** oraz **TabStrip** zajmują cały ekran.
- Określone strony są osobnymi fragmentami tworzonymi na podstawie ich specyfikacji XML.

Przykład 10 – ViewPager oraz TabStrip

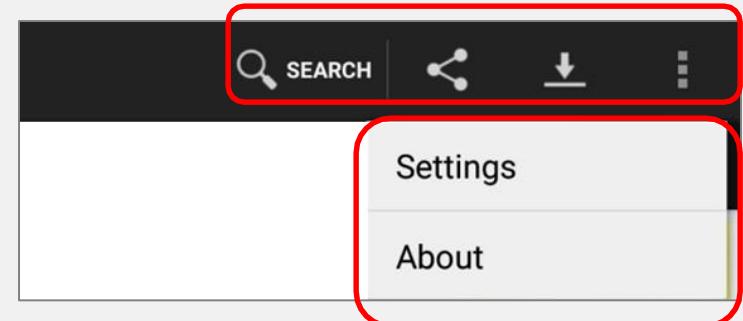
UKŁAD: fragment_page1.xml

```
<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" >  
  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_alignParentTop="true"  
        android:layout_centerHorizontal="true"  
        android:layout_marginTop="22dp"  
        android:textSize="30sp"  
        android:text="Hello, this is Fragment 1" />  
  
</RelativeLayout>
```

Dwa pozostałe układy: **fragment_page2** oraz **fragment_page3** są podobne, różnią się tylko etykietą i kolorem tła.

Przykład 10 – MENU: main_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>
</menu>
```



The app's menu is placed on the Action Bar. Keep in mind that the Tabs shown by the PagerTabStrip widget are NOT connected with the Action Bar.

Przykład 10 – MainActivity.java

1 z 2

```
public class MainActivity extends FragmentActivity {  
    Context context;  
    int duration = Toast.LENGTH_SHORT;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // Get the view from activity_main.xml  
        setContentView(R.layout.activity_main);  
        context = getApplication();  
  
        // Locate the viewPager in activity_main.xml  
        ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);  
  
        // Set the ViewPagerAdapter into ViewPager  
        viewPager.setAdapter(new MyViewPagerAdapter(getSupportFragmentManager()));  
    } //onCreate  
  
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        // Inflate the menu; this adds items to the action bar if it is present.  
        getMenuInflater().inflate(R.menu.main, menu);  
        return true;  
    }  
}
```

1 →

Przykład 10 – MainActivity.java

2 z 2

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    2 if (id == R.id.action_search) {
        Toast.makeText(context, "Search...", duration).show();
        return true;
    }
    else if (id == R.id.action_share) {
        Toast.makeText(context, "Share...", duration).show();
        return true;
    }
    else if (id == R.id.action_download) {
        Toast.makeText(context, "Download...", duration).show();
        return true;
    }
    else if (id == R.id.action_about) {
        Toast.makeText(context, "About...", duration).show();
        return true;
    }
    else if (id == R.id.action_settings) {
        Toast.makeText(context, "Settings...", duration).show();
        return true;
    }
    return false;
}
```

Przykład 10 – FragmentPage1.java

```
public class FragmentPage1 extends Fragment {  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                             Bundle savedInstanceState) {  
  
        // Get the view design from file: res/layout/fragment_page1.xml  
        View view = inflater.inflate(R.layout.fragment_page1, container, false);  
  
        view.setBackgroundColor(Color.parseColor("#55FF0000"));  
  
        return view;  
    }  
  
}
```

3 →

Dwa pozostałe fragmenty: **FragmentPage2**
i **FragmentPage3** są podobne.

Przykład 10 – MiViewPagerAdapter.java

```
public class MyViewPagerAdapter extends FragmentPagerAdapter {  
    // Tab Captions  
    private String tabCaption[] = new String[] { "TAB1", "TAB2", "TAB3" };  
  
    public MyViewPagerAdapter(FragmentManager fragmentManager) {  
        super(fragmentManager);  
    }  
  
    @Override  
    public int getCount() {  
        return tabCaption.length; // return 3 (numbers of tabs in the example)  
    }  
     →  
    @Override  
    public Fragment getItem(int position) {  
        switch (position) {  
            case 0: return new FragmentPage1();  
            case 1: return new FragmentPage2();  
            case 2: return new FragmentPage3();  
        }  
        return null;  
    }  
  
    @Override  
    public CharSequence getPageTitle(int position) {  
        return tabCaption[position]; // return tab caption  
    }  
}
```

Przykład 10 – ViewPager oraz TabStrip

Komentarz

1. Aplikacja składa się z trzech niezależnych części: ActionBar na górze, PagerTabStrip oraz ViewPager. Komponent ViewPager zajmuje najwięcej obszaru roboczego. Wymaga stworzenia własnego adaptera by móc wyświetlić szereg stron. Elementem bazowym każdej strony jest określony fragment.
2. ActionBar tworzony jest na podstawie specyfikacji zawartej w **res/menu/main.xml**. Nasłuchiwanie **onOptionsItemSelected** rejestruje zdarzenia kliknięcia oraz wykonuje powiązane z nimi operacje. Należy zaznaczyć, że ActionBar oraz zakładki PagerTabStrips to dwa niezależne komponenty.
3. **MyViewPagerAdapter** jest odpowiedzialny za stworzenie etykiety oraz zawartości obiektów ViewPagerviews. Dwie najważniejsze metody w ramach tej klasy to: `getItem()` oraz `getPageTitle()`. Obie jako parametr przyjmują określony indeks (0, 1, 2,...) i zwracają odpowiednio stronę lub etykietę strony.

Przykład 11 – Pasek narzędziowy

Komponent typu **Toolbar** jest nową kontrolką wprowadzoną w SDK 5.0. Funkcjonalnie stanowi odpowiednik ActionBar'a, posiadając jednak kilka unikalnych funkcji niedostępnych w starszym bracie.

Podobnie jak ActionBar, pasek Toolbar może zawierać przyciski nawigacyjne, logo, tytuł, podtytuł, kafelki oraz dowolny własny widok.

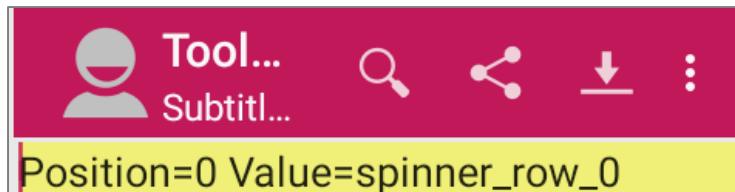
W przeciwieństwie do ActionBar'a – który może być umiejscowiony tylko na górze ekranu – pasek narzędziowy może być umieszczony gdziekolwiek na ekranie. Możliwe jest wykorzystaniu wielu pasków narzędziowych naraz.



Nie sposób wizualnie rozróżnić komponent Toolbar od ActionBar

Przykład 11 – Pasek narzędziowy

Toolbar
Na górze



W tym przykładzie aplikacja ma dwa niezależne paski.

Pasek górny zawiera: logo, tytuł, podtytuł, przyciski oraz menu.

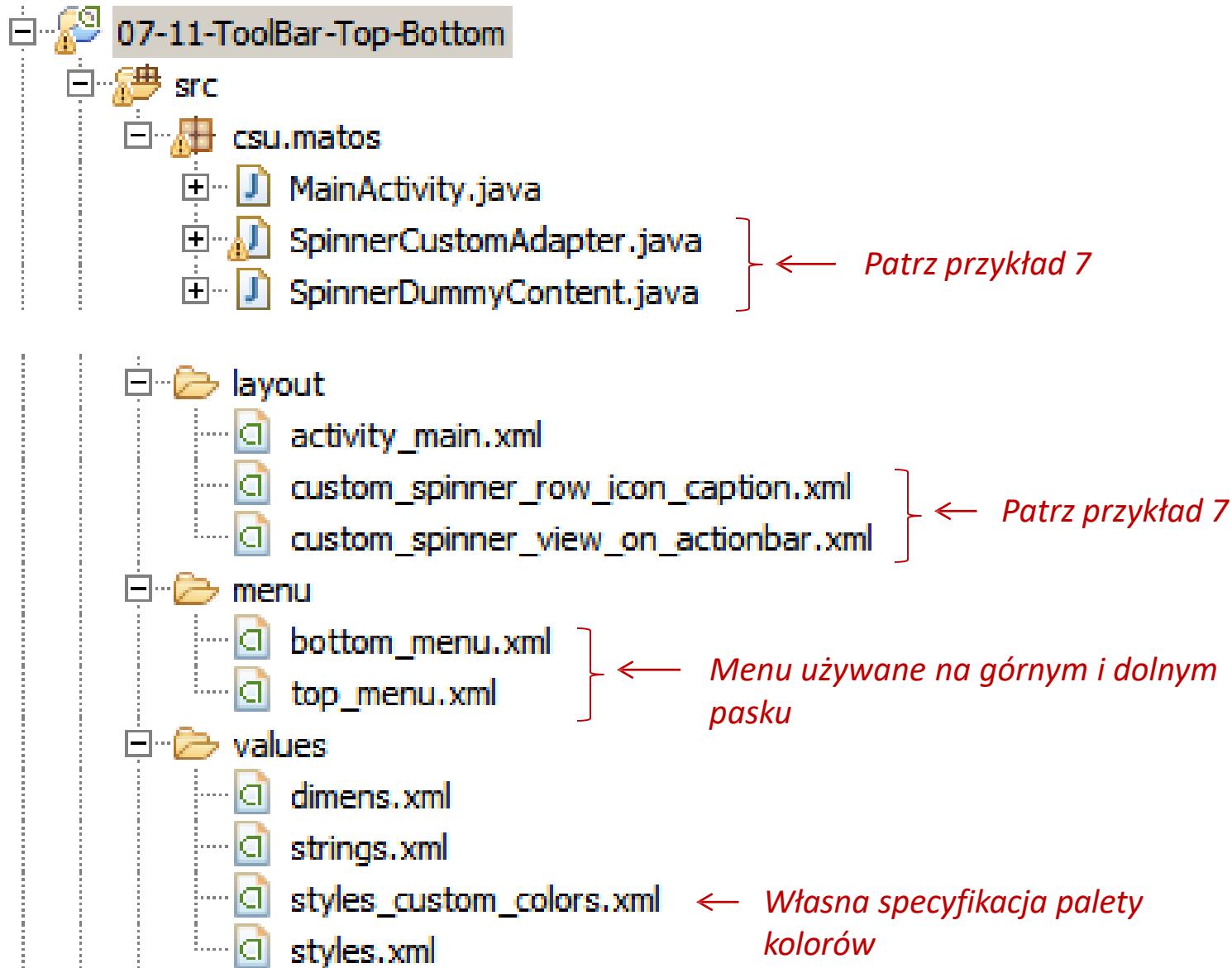
Toolbar
Na dole



Dolny pasek zawiera własny widok (składający się z obiektu Spinner), oraz dodatkowe przyciski.

Każdy pasek narzędziowy może wykorzystywać odrębny styl oraz paletę kolorów.

Przykład 11 – Pasek narzędziowy



Przykład 11 – Układ: activity_main.xml

1 z 2

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="?android:attr/colorBackground"
    android:orientation="vertical"
    android:padding="2dp" >           ← Toolbar górnny
    <Toolbar
        android:id="@+id/Toolbar_top"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?android:attr/colorPrimary"
        android:minHeight="?android:attr actionBarSize"
        android:popupTheme="@android:style/ThemeOverlay.Material.Light"
        android:theme="@android:style/ThemeOverlay.Material.Dark.ActionBar" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="2"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="2dp"
        android:background="#77ffff00"
        android:text="@string/hello_world" />
</LinearLayout>
```

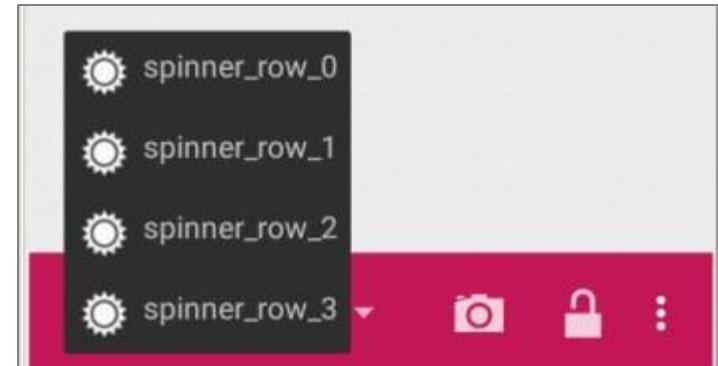


Przykład 11 – Układ: activity_main.xml

2 z 2

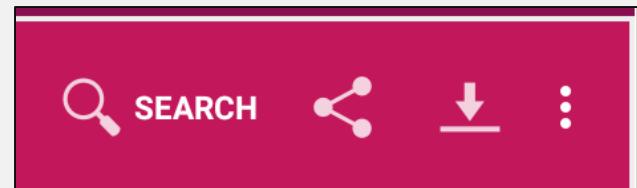
```
Toolbar dolny  
<Toolbar  
    android:id="@+id/Toolbar_bottom"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:background="?android:attr/colorPrimary"  
    android:minHeight="?android:attr/actionBarSize"  
    android:popupTheme="@android:style/ThemeOverlay.Material.Light"  
    android:theme="@android:style/ThemeOverlay.Material.Dark.ActionBar" />  
  
</LinearLayout>
```

Później obiekt typu *Spinner* zostanie dodany.



Przykład 11 – MENU: top_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>
</menu>
```

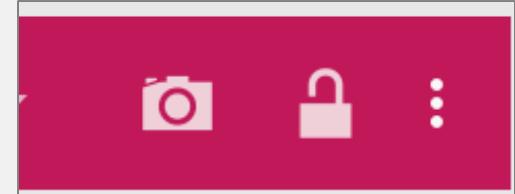


Część z menu jest wyświetlana
przez górny paszek

Przykład 11 – MENU: bottom_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/action_camera_bottom"
        android:icon="@drawable/ic_action_camera"
        android:orderInCategory="100"
        android:showAsAction="always"
        android:title="Camera"/>
    <item
        android:id="@+id/action_unlock_bottom"
        android:icon="@drawable/ic_action_unlock"
        android:orderInCategory="110"
        android:showAsAction="always"
        android:title="Unlock"/>
    <item
        android:id="@+id/action_settings_bottom"
        android:icon="@drawable/ic_launcher"
        android:orderInCategory="120"
        android:showAsAction="never"
        android:title="Settings Bottom "/>
</menu>
```



Druga część menu jest wyświetlana na dolnym pasku

Przykład 11 – STYLE: style_custom_colors.xml

Należy zmodyfikować plik **AndroidManifest** by zmienić paletę kolorów. Należy dodać następującą klauzulę do elementu <application>: **android:theme="@style/MyPinkTheme**

```
<resources>          ↗
<style name="MyPinkTheme" parent="@android:style/Theme.Material.Light.DarkActionBar">
    <!-- Customizing the COLOR PALETTE -->
    <!-- Using PINK theme colors from suggested Android palette. Visit link: -->
    <!-- http://www.google.com/design/spec/style/color.html#color-palette -->
    <!-- Elements of material design described at link: -->
    <!-- https://developer.android.com/training/material/theme.html -->

    <!-- do not show app title -->
    <item name="android:windowNoTitle">true</item>
    <!-- do not show ActionBar -->
    <item name="android:windowActionBar">false</item>

    <!-- colorPrimary: PINK700 (Toolbar background) -->
    <item name="android:colorPrimary">#C2185B</item>
    <!-- colorPrimaryDark: PINK900 (status bar) -->
    <item name="android:colorPrimaryDark">#880E4F</item>
    <!-- colorAccent: PINK700 (UI widgets like: checkboxes, text fields ) -->
    <item name="android:colorAccent">#C2185B</item>

    <!-- colorBackground: PINK100 (window background) -->
    <item name="android:colorBackground">#F8BBD0</item>

</style>
</resources>
```

Definicja dodana w
res/values/styles_custom_colors.xml

Przykład 11 – Pasek narzędziowy

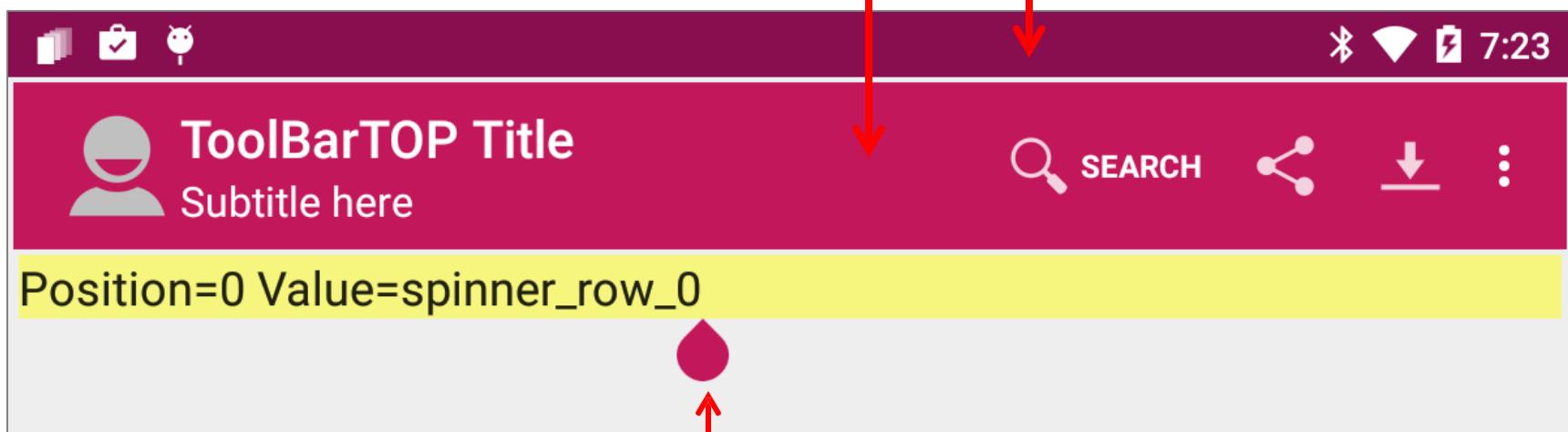
Paleta kolorów zdefiniowana w:

`styles_custom_colors.xml`

`android:colorPrimaryDark`

Toolbar górny

`android:colorPrimary`



`android:colorAccent`

Przykład 11 – MainActivity.java

1 z 4

```
public class MainActivity extends Activity
    implements OnMenuItemClickListener, OnItemSelectedListener {
    EditText txtMsg;
    Toolbar toolbarTop;
    Toolbar toolbarBottom;
    int selectedSpinnerRow = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtMsg = (EditText) findViewById(R.id.txtMsg);

1    //TOP toolbar -----
        toolbarTop = (Toolbar) findViewById(R.id.toolbar_top);
        toolbarTop.setTitle("ToolBarTOP Title");
        toolbarTop.setSubtitle("Subtitle here");
        toolbarTop.inflateMenu(R.menu.top_menu);
        toolbarTop.setLogo(R.drawable.ic_action_app_logo);
        toolbarTop.setOnMenuItemClickListener(this);
2    //BOTTOM toolbar -----
        toolbarBottom = (Toolbar) findViewById(R.id.toolbar_bottom);
//toolbarBottom.setTitle("ToolBarBOTTOM Title");
//toolbarBottom.setSubtitle("Subtitle here");
//toolbarBottom.setLogo(R.drawable.ic_action_app_logo);
        toolbarBottom.inflateMenu(R.menu.bottom_menu);
//adding a custom-view
        prepareCustomView(toolbarBottom);
        toolbarBottom.setOnMenuItemClickListener(this);
    }
}
```

Przykład 11 – MainActivity.java

2 z 4

```
@Override
public boolean onMenuItemClick(MenuItem item) {
    int id = item.getItemId();

    // Handle TOP action bar item clicks here.
    if (id == R.id.action_search) {
        txtMsg.setText("Search...");      return true;
    } else if (id == R.id.action_share) {
        txtMsg.setText("Share...");      return true;
    } else if (id == R.id.action_download) {
        txtMsg.setText("Download...");   return true;
    } else if (id == R.id.action_about) {
        txtMsg.setText("About...");     return true;
    } else if (id == R.id.action_settings) {
        txtMsg.setText("Settings...");   return true;
    }

    // Handle BOTTOM action bar item
    if (id == R.id.action_camera_bottom) {
        txtMsg.setText("Camera...");    return true;
    } else if (id == R.id.action_unlock_bottom) {
        txtMsg.setText("Unlock...");    return true;
    } else if (id == R.id.action_settings_bottom) {
        txtMsg.setText("Settings-Bottom..."); return true;
    }

    return false;
}
```

3 →

Przykład 11 – MainActivity.java

3 z 4

```
private void prepareCustomView(Toolbar toolbar) {  
    // 4 → // setup the BOTTOM toolbar  
    LayoutInflator inflater = (LayoutInflator) getApplication()  
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
  
    toolbar.addView(inflater.inflate(R.layout.custom_spinner_view_on_actionbar, null));  
  
    // create the custom adapter to feed the spinner  
    SpinnerCustomAdapter customSpinnerAdapter = new SpinnerCustomAdapter(  
        getApplicationContext(),  
        SpinnerDummyContent.customSpinnerList);  
  
    // plumbing - get access to the spinner widget shown on the actionBar  
    Spinner customSpinner = (Spinner) toolbar.findViewById(R.id.spinner_data_row);  
  
    // bind spinner and adapter  
    customSpinner.setAdapter(customSpinnerAdapter);  
  
    // put a listener to wait for spinner rows to be selected  
    customSpinner.setOnItemSelectedListener(this);  
  
    customSpinner.setSelection(selectedSpinnerRow);  
}  
//prepareCustomView
```

Przykład 11 – MainActivity.java

4 z 4

5

```
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    selectedSpinnerRow = position;

    TextView caption = (TextView) toolbarBottom.findViewById(
        R.id.txtSpinnerRowCaption);

    txtMsg.setText( "Position=" + position + " Value=" + caption.getText());

}

@Override
public void onNothingSelected(AdapterView<?> parent) {
    // TODO nothing to do here...
}

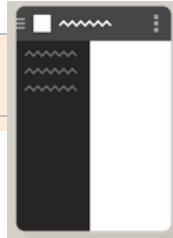
}
```

Przykład 11 – Pasek narzędziowy

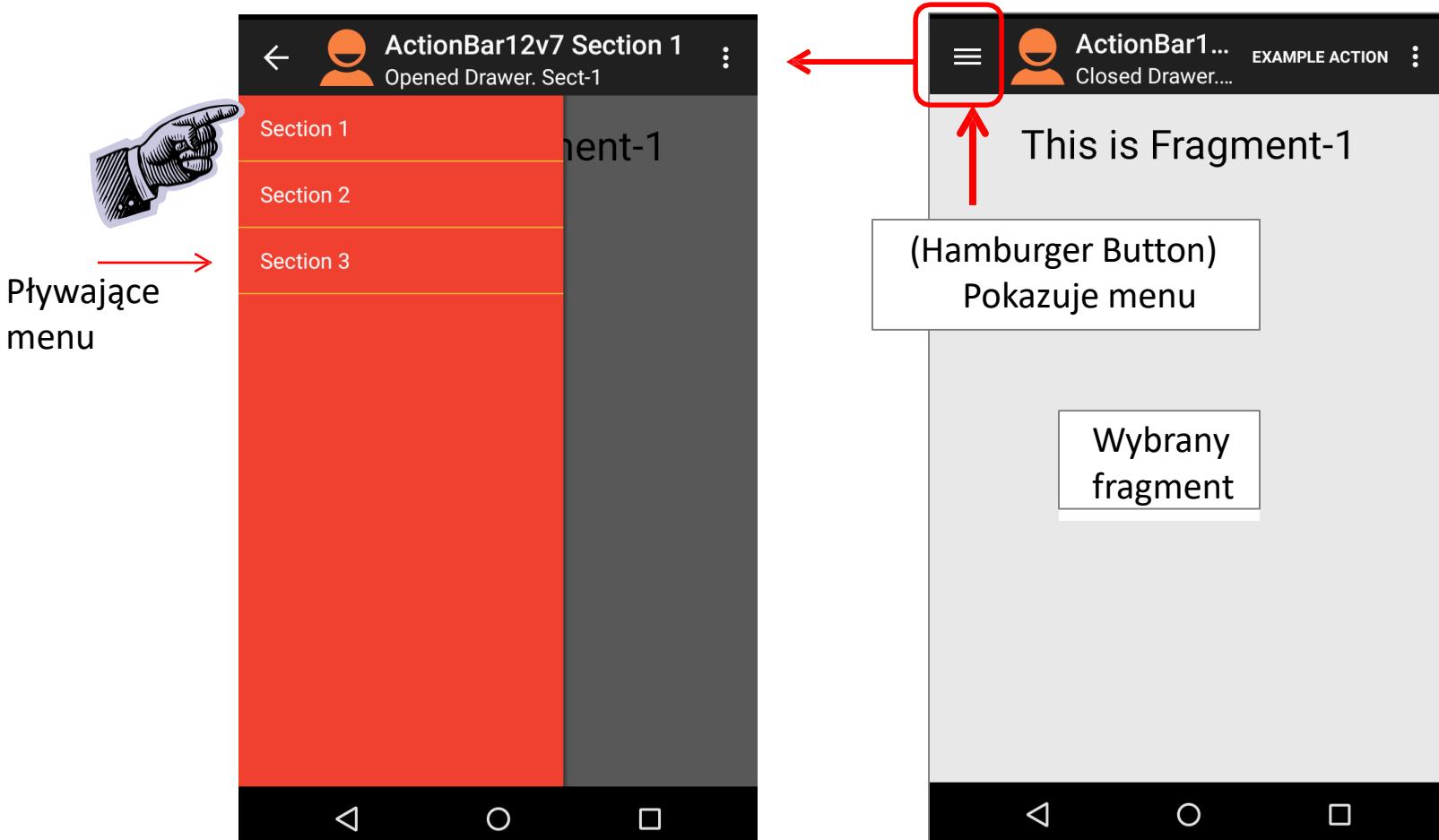
Komentarz

1. Górnego paska narzędziowego zyskuje logo, tytuł, podtytuł oraz pozycje menu głównego.
2. Podobnie dolny pasek zyskuje listę rozwijalną i inne pozycje menu.
3. Metoda `onMenuItemClick()` jest wspólna dla górnego i dolnego paska, a jej zadaniem jest obsługa zdarzenia kliknięcia na poszczególne opcje.
4. Metoda `prepareCustomView()` tworzy obiekty zgodnie z podaną specyfikacją xml. Każdy wiersz listy zawiera etykietę tekstową oraz obrazek (zgodnie z `custom_spinner_row_icon_caption.xml`). Własny adapter – jak w przykładzie 7 – jest odpowiedzialny za dodanie elementów do komponentu Spinner.
5. Gdy użytkownik wybierze pozycję z listy wywoływana jest metoda `onItemSelected()` by zapamiętać wybraną pozycję oraz na tej podstawie podjąć odpowiednie działania.

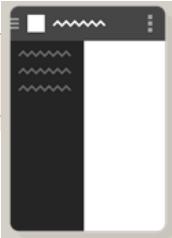
Przykład 12 – Pływające menu



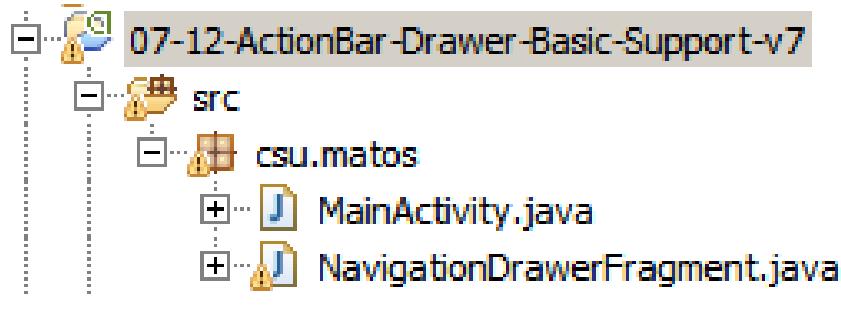
- ActionBar może być zmodyfikowany by zawierał specjalny przycisk (tzw. **HAMBURGER button**).
- Po kliknięciu, tymczasowo przykrywa ekran prezentując tzw. **DrawerLayout**.
- Zwykle ogranicza się do prezentacji listy opcji (zwanych sekcjami lub fragmentami).
- Po wyborze jednej z opcji, menu znika a widok aktywności jest aktualizowany na podstawie wybranej opcji.



Przykład 12 – Pływające menu



ARCHITEKTURA APLIKACJI

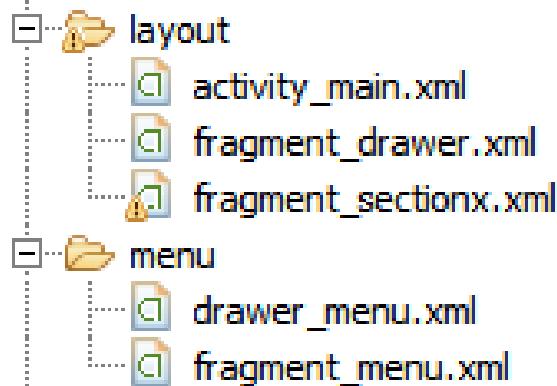


Przykład stworzony na podstawie odpowiedniego szablonu aplikacji w Android Studio.

Definicja sposobu wyświetlania listy z pływającym menu



Biblioteka **appcompat-v4** dla fragmentów, a **appcompat-v7** by wspierać ActionBar.



Podstawowe układy XML

Wygląd opcji menu

Przykład 12 – UKŁAD: activity_main.xml

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout_activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="csu.matos.MainActivity" >

    <!-- main content view, consumes the entire screen -->
    <FrameLayout
        android:id="@+id/main_fragment_frameLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!-- the navigation drawer will partially cover the screen -->
    <fragment
        android:id="@+id/navigationDrawerFragment_host_container"
        android:name="csu.matos.NavigationDrawerFragment"
        android:layout_width="@dimen/navigation_drawer_width"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        tools:layout="@layout/fragment_navigation_drawer" />

</android.support.v4.widget.DrawerLayout>
```

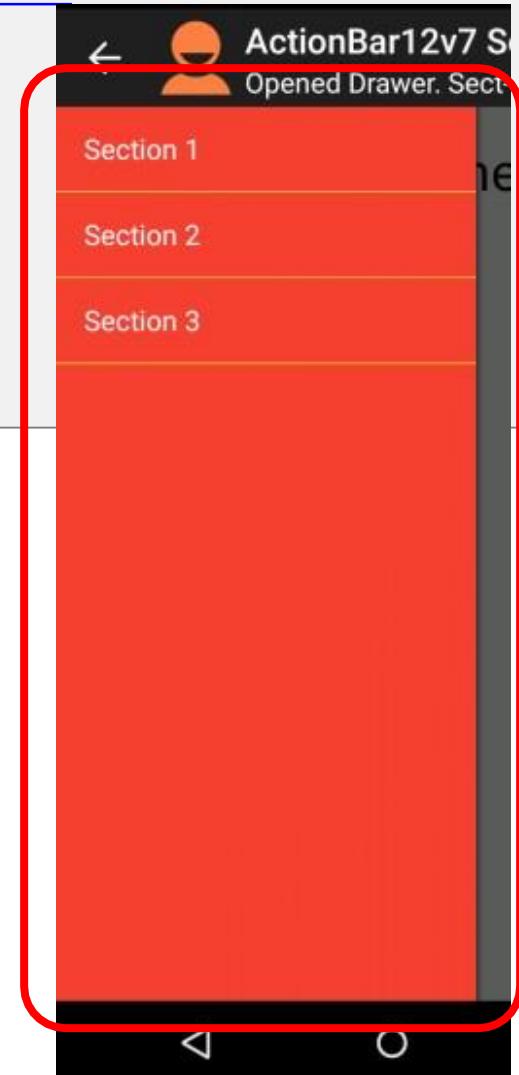


Miejsce na pływający panel menu

Przykład 12 – UKŁAD: fragment_drawer.xml

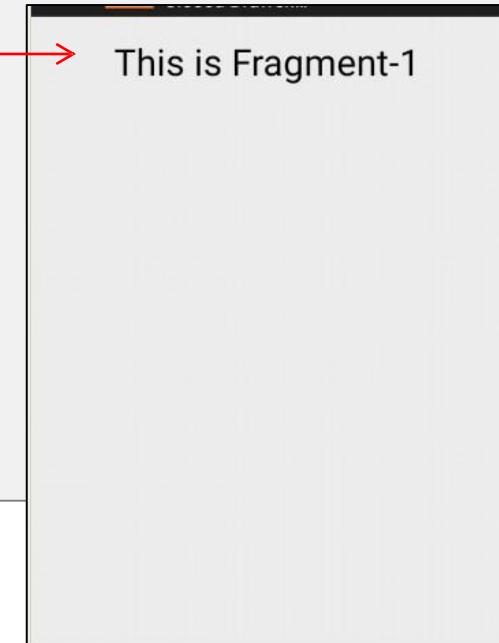
```
<ListView  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="#F44336"  
    android:choiceMode="singleChoice"  
    android:divider="@android:color/holo_orange_light"  
    android:dividerHeight="1dp"  
    tools:context="csu.matos.NavigationDrawerFragment" />
```

Prezentowana jest prosta lista.
Możliwa jest zmiana wyglądu
wierszy przez stworzenie
własnego adaptera.



Przykład 12 – UKŁAD: fragment_sectionx.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="csu.matos.MainActivity$PlaceholderFragment" >  
  
    <TextView  
        android:id="@+id/txt_section_label"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:gravity="center"  
        android:text="This is fragment-n"  
        android:textColor="#ff000000"  
        android:textSize="30sp" />  
  
</RelativeLayout>
```

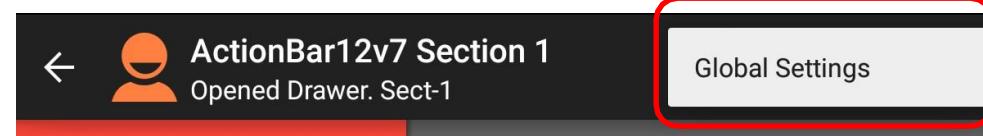


Potrzebny jest osobny układ dla każdego elementu menu pływającego. Dla uproszczenia pokazano układ dla jednego elementu.

Przykład 12 – MENU: fragment_main.xml

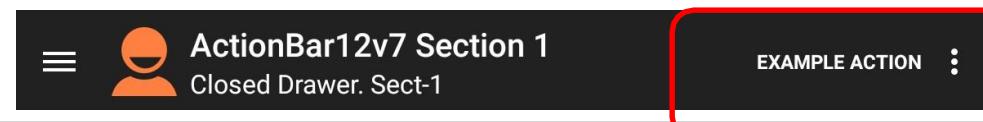
res/menu/drawer_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto" >
    <item
        android:id="@+id/action_global_settings"
        android:orderInCategory="100"
        android:title="Global Settings"
        app:showAsAction="never"/>
</menu>
```



res/menu/fragment_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:tools="http://schemas.android.com/tools" >
    <item
        android:id="@+id/action_example"
        android:title="@string/action_example"
        app:showAsAction="withText|ifRoom"/>
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never"/>
</menu>
```



Przykład 12 – MainActivity.java

1 z 5

```
import ...
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v7.app.ActionBar; ←
import android.support.v7.app.ActionBarActivity;

1 → public class MainActivity extends ActionBarActivity
        implements NavigationDrawerFragment.NavigationCallbacks {

    private NavigationDrawerFragment mNavigationDrawerFragment;
    private CharSequence mTitle;
    ActionBar actionBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // this is the navigation fragment
        mNavigationDrawerFragment = (NavigationDrawerFragment)
            getSupportFragmentManager().findFragmentById(
                R.id.navigationDrawerFragment_host_container);

        // link navigation drawerFragment to main_activity layout
        mNavigationDrawerFragment.setUp(
            R.id.navigationDrawerFragment_host_container,
            R.id.drawer_layout_activity_main );

        prepareActionBar();
    }
}
```

Przykład 12 – MainActivity.java

2 z 5

```
@Override
public void onNavigationDrawerItemSelected(int position) {
    // update the main GUI content with selected
    fragment(SectionX) FragmentManager fragmentManager =
        getSupportFragmentManager(); fragmentManager
            .beginTransaction()
            .replace(R.id.main_fragment_frameLayout,
                PlaceholderFragment.newInstance(position + 1))
            .commit();
}

public void onSectionAttached(int number) {
    switch (number) {
        case 1:
            mTitle = getString(R.string.title_section1);
            break;
        case 2:
            mTitle = getString(R.string.title_section2);
            break;
        case 3:
            mTitle = getString(R.string.title_section3);
            break;
    }
}
```

3

4

Przykład 12 – MainActivity.java

3 z 5

```
public void prepareActionBar() {
    actionBar = getSupportActionBar();
    actionBar.setDisplayShowTitleEnabled(true);
    actionBar.setDisplayShowHomeEnabled(true);
    actionBar.setLogo(R.drawable.ic_launcher);
    actionBar.setDisplayUseLogoEnabled(true);
    actionBar.setTitle(getTitle() + " " + mTitle);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    if (!mNavigationDrawerFragment.isDrawerOpen()) {
        getMenuInflater().inflate(R.menu.fragment_menu, menu);
        prepareActionBar();
        return true;
    }
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here.
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

5 →

6 →

7 →

Przykład 12 – MainActivity.java

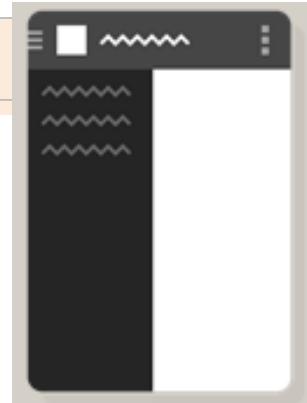
4 z 5

```
public static class PlaceholderFragment extends Fragment {  
  
    private static final String ARG_SECTION_NUMBER = "section_number";  
  
    8 → public static PlaceholderFragment newInstance(int sectionNumber) {  
        PlaceholderFragment fragment = new PlaceholderFragment();  
        Bundle args = new Bundle();  
        args.putInt(ARG_SECTION_NUMBER, sectionNumber);  
        fragment.setArguments(args);  
        return fragment;  
    }  
  
    public PlaceholderFragment() { }  
  
    @Override  
    9 → public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                                Bundle savedInstanceState) {  
        View fragmentView = inflater.inflate(R.layout.fragment_sectionx,  
                                             container, false);  
        TextView txtCaptionFragment = (TextView)fragmentView.findViewById(  
                                         R.id.txt_section_label);  
        txtCaptionFragment.setText("This is Fragment-"  
                               + getArguments().getInt(ARG_SECTION_NUMBER, 1));  
        return fragmentView;  
    }  
}
```

Przykład 12 – MainActivity.java

5 z 5

```
A → @Override  
public void onAttach(Activity activity) {  
    super.onAttach(activity);  
    ((MainActivity) activity).onSectionAttached getArguments()  
        .getInt(ARG_SECTION_NUMBER));  
}  
  
}//PlaceHolderFragment  
}
```



Komentarze

Aplikacja została stworzona na podstawie wbudowanego szablonu ‘**Drawer Navigation Activity**’.

*Aplikacja składa się z dwóch rodzajów fragmentów. Jeden jest dedykowany do stworzenia opcji menu pływającego. Nazwijmy ten fragment **drawer-fragment**. Pozostałe fragmenty dedykowane są obsłudze ekranów wyświetlanych po wyborze odpowiedniej opcji z listy przez użytkownika. Nazwijmy je **section-fragments**.*

1. Główna aktywność dziedziczy po klasie **ActionBarActivity**, która jest klasą bazową dla aktywności wykorzystujących bibliotekę zapewniającą kompatybilność wsteczną i implementację ActionBar'a (oraz metody takie jak `getSupportFragmentManager()`, `getSupportActionBar()`, itp.). Możliwe jest ich wykorzystanie dla API poziomu 7 bądź wyższego, pod warunkiem ustawienia skórki jako `Theme.AppCompat` (lub podobnej). Za obsługę pływającego menu (otwarcie, zamknięcie itp.) odpowiedzialna jest biblioteka `appcompat-v4`.

2. Plik zasobów **activity_main.xml** definiuje element `<fragment>` w ramach którego umieszczone jest menu pływające. Punkt 2 odnosi się do momentu w którym obiekt *drawer-fragment* jest tworzony i osadzony w miejście komponentu oznaczonego znacznikiem `<fragment>`. Pływające menu zostanie wyświetcone na ekranie gdy użytkownik kliknie specjalny przycisk (**Hamburger** buton) lub przesunie palcem przez długość ekranu.
3. Metoda **onNavigationDrawerItemSelected** stanowi implementację interfejsu **Callback** którego celem jest połączenie obiektu *drawer-fragment* oraz głównej aktywności. Po jej wywołaniu tworzony jest obiekt *section-fragment* zgodnie z wyborem użytkownika z listy.
4. Gdy fragment *section-fragment* jest dołączony do okna aplikacji (zwykle zastępując poprzedni widok) metoda **onSectionAttached** jest wywoływana. Następuje wówczas aktualizacja zmiennej globalnej **mTitle** na nazwę wybranej pozycji.
5. Pobierana jest referencja do komponentu **ActionBar** i dokonywana jest jego inicjalizacja podając tytuł, logo i przycisk główny.

6. Każdy fragment section-fragment może mieć swoje indywidualne menu. Metoda **onCreateOptionsMenu** odpowiedzialna jest za tworzenie właściwego menu i aktualizację komponentu ActionBar.
7. Gdy pozycja menu umieszczona na komponencie ActionBar zostaje wybrana, metoda **onOptionsItemSelected** jest wywoływana by obsłużyć żądanie użytkownika.
8. Dla uproszczenia, prezentowany przykład tworzy tylko jeden typ fragmentu *section-fragment*. Klasa **PlaceholderFragment** reprezentuje odpowiedź aplikacji na wybór użytkownika.
9. Każdy z fragmentów *section-fragments* będzie posiadał unikalny alias ('This is fragment-1', 'This is fragment-2, ...). Metoda **onCreateView** aktualizuje widok za każdym razem, gdy użytkownik wybierze pozycję z listy.

Przykład 12 – NavigationDrawerFragment.java

1 z 9

```
import ...  
import android.support.v4.app.Fragment;  
import android.support.v4.view.GravityCompat;  
import android.support.v4.widget.DrawerLayout;  
import android.support.v7.app.ActionBar;  
import android.support.v7.app.ActionBarActivity;  
import android.support.v7.app.ActionBarDrawerToggle;  
  
1 → public class NavigationDrawerFragment extends Fragment {  
  
    private static final String STATE_SELECTED_POSITION  
    ="chosen_drawer_position";  
    private static final String PREF_USER_LEARNED_DRAWER = "drawer_learned";  
  
    private NavigationDrawerCallbacks mCallbacks;  
    private ActionBarDrawerToggle mDrawerToggle;  
    private DrawerLayout mainActivityLayout;  
    private ListView mDrawerListView;  
    private View mDrawerFragmentHost;  
    private ActionBar actionBar;  
  
    private int mCurrentSelectedPosition = 0;  
    private boolean mFromSavedInstanceState;  
    private boolean mUserLearnedDrawer;  
  
    public NavigationDrawerFragment() {  
    }
```

Przykład 12 – NavigationDrawerFragment.java

2 z 9

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    // Read in the flag indicating whether or not the user has demonstrated  
    // awareness of the drawer. See PREF_USER_LEARNED_DRAWER for details.  
    SharedPreferences sp = PreferenceManager  
        .getDefaultSharedPreferences(getActivity());  
    mUserLearnedDrawer = sp.getBoolean(PREF_USER_LEARNED_DRAWER, false);  
  
    if (savedInstanceState != null) {  
        mCurrentSelectedPosition = savedInstanceState  
            .getInt(STATE_SELECTED_POSITION);  
        mFromSavedInstanceState = true;  
    }  
  
    // Select either the default item (0) or the last selected item.  
    selectedItem(mCurrentSelectedPosition);  
}  
  
@Override  
public void onActivityCreated(Bundle savedInstanceState) {  
    super.onActivityCreated(savedInstanceState);  
    // this fragment may alter the action bar.  
    setHasOptionsMenu(true);  
}
```

2 →

3 →

Przykład 12 – NavigationDrawerFragment.java

3 z 9

```
@Override
public View onCreateView( LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    mDrawerListView = (ListView) inflater.inflate(
        R.layout.fragment_drawer, container, false);

    mDrawerListView.setOnItemClickListener(new
        AdapterView.OnItemClickListener(){
            @Override
            public void onItemClick(AdapterView<?> parent, View view,
                                int position, long id) {
                selectItem(position);
            }
        });
    mDrawerListView.setAdapter(new ArrayAdapter<String>(
        getActionBar().getThemedContext(),
        android.R.layout.simple_list_item_1,
        android.R.id.text1,
        new String[] { getString(R.string.title_section1),
                      getString(R.string.title_section2),
                      getString(R.string.title_section3), } ) );
}

mDrawerListView.setItemChecked(mCurrentSelectedPosition, true);
return mDrawerListView;

}
```

4

5

Przykład 12 – NavigationDrawerFragment.java

4 z 9

```
public boolean isDrawerOpen() {  
    return mainActivityLayout != null  
        && mainActivityLayout.isDrawerOpen(mDrawerFragmentHost);  
}  
  
public void setUp(int drawerFragmentId, int activity_main_Id) {  
    //find the place in the main layout where drawer fragment is to be shown  
    mDrawerFragmentHost = getActivity().findViewById(drawerFragmentId);  
    //access the full activity_main layout (DrawerView)  
    mainActivityLayout = (DrawerLayout) getActivity().findViewById(  
                           activity_main_Id);  
    // set custom shadow to overlay main content when the drawer opens  
    mainActivityLayout.setDrawerShadow(R.drawable.drawer_shadow,  
                                      GravityCompat.START);  
    // set up the drawer's list view with items and click listener  
    actionBar = getActionBar();  
    actionBar.setDisplayHomeAsUpEnabled(true);  
    actionBar.setHomeButtonEnabled(true);  
  
    try {  
        mDrawerToggle = new CustomActionBarDrawerToggle(mainActivityLayout);  
        mainActivityLayout.setDrawerListener(mDrawerToggle);  
    } catch (RuntimeException e) {  
        Toast.makeText(getActivity(), "Error:"+e.getMessage(),  
                      Toast.LENGTH_SHORT).show();  
    }  
}
```

6



Przykład 12 – NavigationDrawerFragment.java

5 z 9

```
//the unseen drawer should be shown to the user (once at launch time)
if (!mUserLearnedDrawer && !mFromSavedInstanceState) {
    mainActivityLayout.openDrawer(mDrawerFragmentHost);
}

// Defer code dependent on restoration of previous instance state.
mainActivityLayout.post(new Runnable() {
    @Override
    public void run() {
        mDrawerToggle.syncState();
    }
});

private void selectItem(int position) {
    mCurrentSelectedPosition = position;
    if (mDrawerListView != null) { //user is learning - section checked!
        mDrawerListView.setItemChecked(position, true);
    }
    if (mainActivityLayout != null) {
        mainActivityLayout.closeDrawer(mDrawerFragmentHost);
    }
    if (mCallbacks != null) {
        mCallbacks.onNavigationDrawerItemSelected(position);
    }
}
```

7

Przykład 12 – NavigationDrawerFragment.java

6 z 9

```
@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    try {
        mCallbacks = (NavigationDrawerCallbacks) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException("Main must implement Nav.DrawerCallbacks.");
    }
}

@Override
public void onDetach() {
    super.onDetach();
    mCallbacks = null;
}

@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putInt(STATE_SELECTED_POSITION, mCurrentSelectedPosition);
}

@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    // Forward the new configuration the drawer toggle component.
    mDrawerToggle.onConfigurationChanged(newConfig);
}
```

8

9

Przykład 12 – NavigationDrawerFragment.java

7 z 9

```
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    // (Opened Drawer) show the global app actions in the action bar.
    if (mainActivityLayout != null && isDrawerOpen()) {
        inflater.inflate(R.menu.drawer_menu, menu);

    }
    super.onCreateOptionsMenu(menu, inflater);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (mDrawerToggle.onOptionsItemSelected(item)) {
        return true;
    }

    if (item.getItemId() == R.id.action_example) {
        Toast.makeText(getActivity(),"Ex. action.",Toast.LENGTH_SHORT).show();
        return true;
    }

    return super.onOptionsItemSelected(item);
}

private ActionBar getActionBar() {
    return ((ActionBarActivity) getActivity()).getSupportActionBar();
}
```

A →

Przykład 12 – NavigationDrawerFragment.java

8 z 9

```
public static interface NavigationDrawerCallbacks {  
    B → void onNavigationDrawerItemSelected(int position);  
}  
  
C → private class CustomActionBarDrawerToggle extends ActionBarDrawerToggle {  
  
    public CustomActionBarDrawerToggle(DrawerLayout mainActivityLayout) {  
        super(getActivity(), mainActivityLayout,  
              R.string.navigation_drawer_open,R.string.navigation_drawer_close );  
    }  
  
    @Override  
    public void onDrawerClosed(View view) {  
        actionBar.setSubtitle("Closed Drawer. Sect"+(mCurrentSelectedPosition + 1));  
        getActivity().invalidateOptionsMenu(); //next is onPrepareOptionsMenu()  
    }  
}
```

Przykład 12 – NavigationDrawerFragment.java

9 z 9

```
D → @Override
public void onDrawerOpened(View drawerView) {

    if (!isAdded()) {
        return;
    }

    if (!mUserLearnedDrawer) {
        // The user manually opened the drawer; store this flag to
        // prevent auto-showing the navig.drawer in the future.
        mUserLearnedDrawer = true;

        SharedPreferences sp = PreferenceManager
            .getDefaultSharedPreferences(getActivity());

        sp.edit().putBoolean(PREF_USER_LEARNED_DRAWER, true).apply();
    }

    actionBar.setSubtitle("Open Drawer. Sect-"+(mCurrentSelectedPosition + 1));
    getActivity().invalidateOptionsMenu(); // next is onPrepareOptionsMenu()
}

//CustomActionBarDrawerToggle

}// Fragment
```

Komentarze

1. Klasa **NavigationDrawerFragment** jest odpowiedzialna za obsługę zdarzeń dla obiektu *drawer-fragment*. Gdy użytkownik wybierze daną opcję, fragment reprezentujący wybór prezentowany jest na ekranie.
2. Gdy aplikacja jest uruchamiana po raz pierwszy, nie ma żadnego uprzednio zapisanego stanu aplikacji. Dlatego całe menu pływające zostanie wyświetcone na ekranie. Gdy użytkownik wypróbuje każdą opcję w menu, flaga boolowska **mUserLearnedDrawer** zostanie ustawiona by zapamiętać ten fakt. Przy każdym kolejnym uruchomieniu aplikacji, na podstawie zapamiętanej wartości zostanie podjęta decyzja czy pokazać panel z menu pływającym czy też nie.
3. Metoda **setHasOptionsMenu()** jest wywoływana by powiadomić system, że dany fragment chce wyświetlić własne menu.

4. Wywołanie `onCreateView()` skutkuje stworzeniem obiektów na podstawie układu *drawer-fragment*. Nasłuchiwacz kliknięcia zapamiętuje pozycję wybranego wiersza listy. W naszym przykładzie tworzona jest zwykła lista łańcuchów znaków, jednak może prezentować dane dowolnego typu.
5. Przykład wykorzystuje prosty `ArrayAdapter<String>` by wypełnić wszystkie wiersze listy.
6. Menu pływające jest przygotowywane przez wywołanie metody `setUp()`. Na początku lokalizuje ona miejsce na ekranie gdzie należy umieścić fragment o nazwie *drawer-fragment*. Następnie dodawany jest cień do prawej krawędzi menu. Następnie do ActionBar'a dodawany jest przycisk typu **Hamburger button**, informujący użytkownika o istnieniu menu pływającego. Kliknięcie na ten przycisk powoduje wyświetlenie menu.



7. Gdy użytkownik wybierze pozycję z listy jej indeks jest zapamiętywany, jest ona oznaczona jako „odwiedzona”, a menu zostaje zamknięte. Główna aktywność jest informowana poprzez wywołanie zwrotne o dokonanym wyborze.
8. Fragment *drawer-fragment* musi mieć pewność, że główna aktywność nasłuchuje na jego komunikaty. Metoda `onAttach()` weryfikuje zatem implementację interfejsu `Callback`.
9. Zanim aplikacja zostanie zamknięta, pozycja ostatnio wybranego elementu jest zapamiętywana w kolekcji typu `Bundle`. Następna sesja z aplikacją rozpocznie się od wyświetlenia tej pozycji.
 - A. Fragment *drawer-fragment* tworzy obiekty odpowiedzialne za wyświetlenie menu.
 - B. Interfejs wywołania zwrotnego jest definiowany. W tym przykładzie składa się z pojedynczej metody `onNavigationDrawerItemSelected()` akceptującej liczbę całkowitą reprezentującą wybraną pozycję listy.

- C. Klasa CustomActionBarDrawerToggle opisuje co się dzieje w momencie otwarcia i zamknięcia menu pływającego. Metoda onDrawerClosed() aktualizuje tytuł ActionBar'a zgodnie z wybraną pozycją. Następuje również zmiana pozycji w menu.
- D. Za pierwszym wywołaniem metody onDrawerOpenned() fragment *drawer-fragment* nie został jeszcze dodany do aktywności bazowej, co implikuje wyświetlenie menu bocznego. Informacja o wyborze tej pozycji z menu jest przechowywana w kolekcji typu klucz-wartość. Metoda aktualizuje później tytuł ActionBar'a (który początkowo jest pusty) i zmienia zawarte tam menu.

Wykorzystanie menu i ActionBar

PYTANIA?

Programowanie urządzeń mobilnych

Wykorzystanie WebView

Tłumaczenie i adaptacja materiałów: dr Tomasz Xięski.
Na podstawie prezentacji udostępnionych przez Victor Matos, Cleveland State University.

Portions of this page are reproduced from work created and [shared by Google and](#) used according to terms

Metody tworzenia aplikacji

Istnieją trzy sposoby realizacji aplikacji na system Android:

1. Aplikacja natywna

Tworzona jest przy wykorzystaniu natywnego SDK i przechowywana jest na urządzeniu. Wykorzystuje wszelkie komponenty platformy Android jak widżety, usługi, aktywności, fragmenty, dostawcy treści, powiadomienia, itp.)

2. Aplikacja internetowa

Zwykle realizowana jest jako aplikacja zdalna i umożliwia pobranie zasobów z serwera z wykorzystaniem zainstalowanej przeglądarki internetowej.

3. Aplikacja hybrydowa

Realizowana jest jako aplikacja internetowa, jednakże prócz znaczników html i wykorzystania zdalnych zasobów możliwa jest interakcja z lokalnymi zasobami i możliwościami urządzenia (np. dostęp do czujników)

Metody tworzenia aplikacji

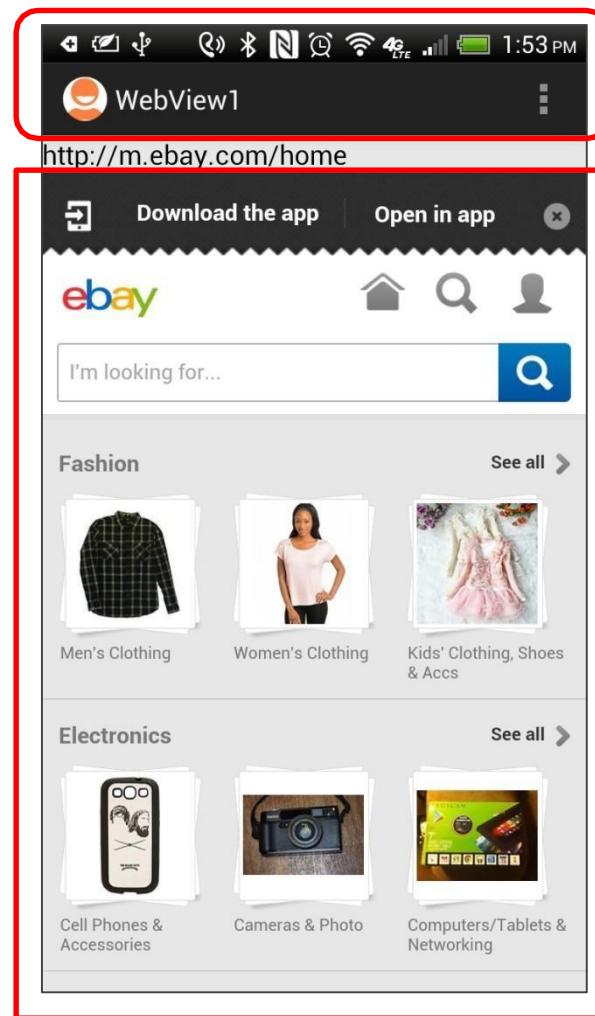
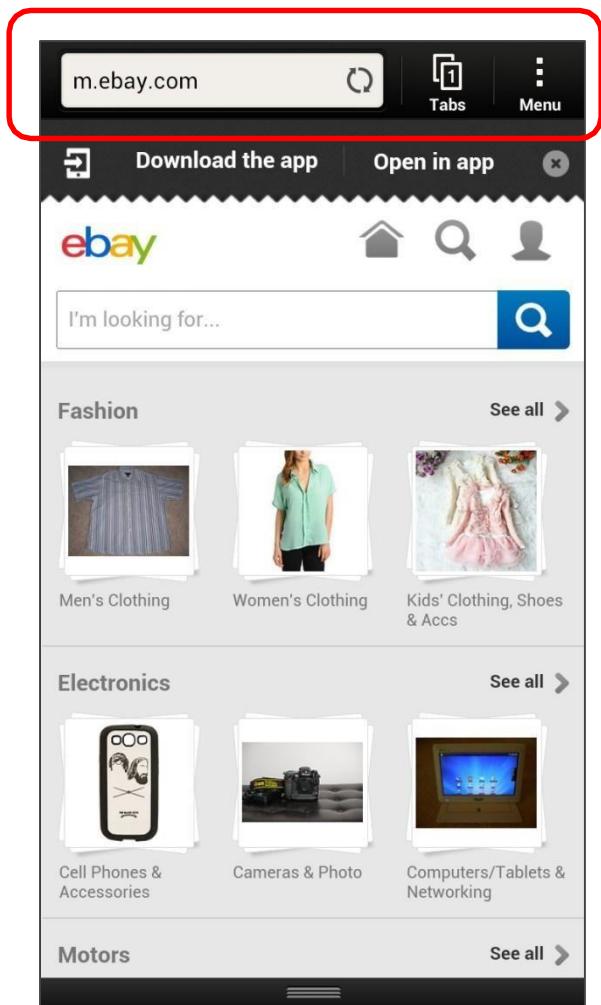
Każde podejście ma swoje **wady i zalety**. Przykładowo:

- Opcja 1 pozwala na tworzenie bogatszego interfejsu i wykorzystuje w pełni dostępne zasoby sprzętowe, ale ograniczona jest do platformy Android. Często wykorzystanie innej platformy wymaga przepisania znacznej części bądź całości aplikacji.
- Opcja 2 pozwala uzyskać największy stopień przenośności. Każde urządzenie z dostępem do przeglądarki internetowej może otworzyć taką aplikację, jednakże brak jest bezpośredniego dostępu do zasobów sprzętowych urządzenia mobilnego.
- Opcja 3 stanowi kompromis między pozostałymi podejściami. Wykorzystanie standardu HTML (włącznie z elementami wersji piątej) oraz dostęp z poziomu języka skryptowego do fizycznych zasobów urządzenia zapewnia duże możliwości i zachowanie przyzwoitych możliwości przenośności.

WebView - właściwości

1. Komponent WebView wykorzystuje otwarto-źródłowy silnik **WebKit** (znany z przeglądarek typu Chrome czy Opera).
2. Komponent WebView nie jest przeglądarką. Pomimo, że umożliwia wyświetlenie strony internetowej, ale nie posiada standardowych przycisków jak Wstecz, Dalej, Odśwież, itp.
3. Klasa WebView zawiera metody do:
 - Ładowania/Przeładowania strony, nawigacji wprzód lub wstecz poprzez historię przeglądarki,
 - Przybliżanie i oddalanie,
 - Wykorzystanie stylów CSS oraz skryptów JavaScript by zapewnić podobne GUI na różnych urządzeniach,
 - Wymiana danych z urządzeniem przez interfejs JavaScript,
 - Wyszukiwania tekstu, pobierania zdjęć,

WebView

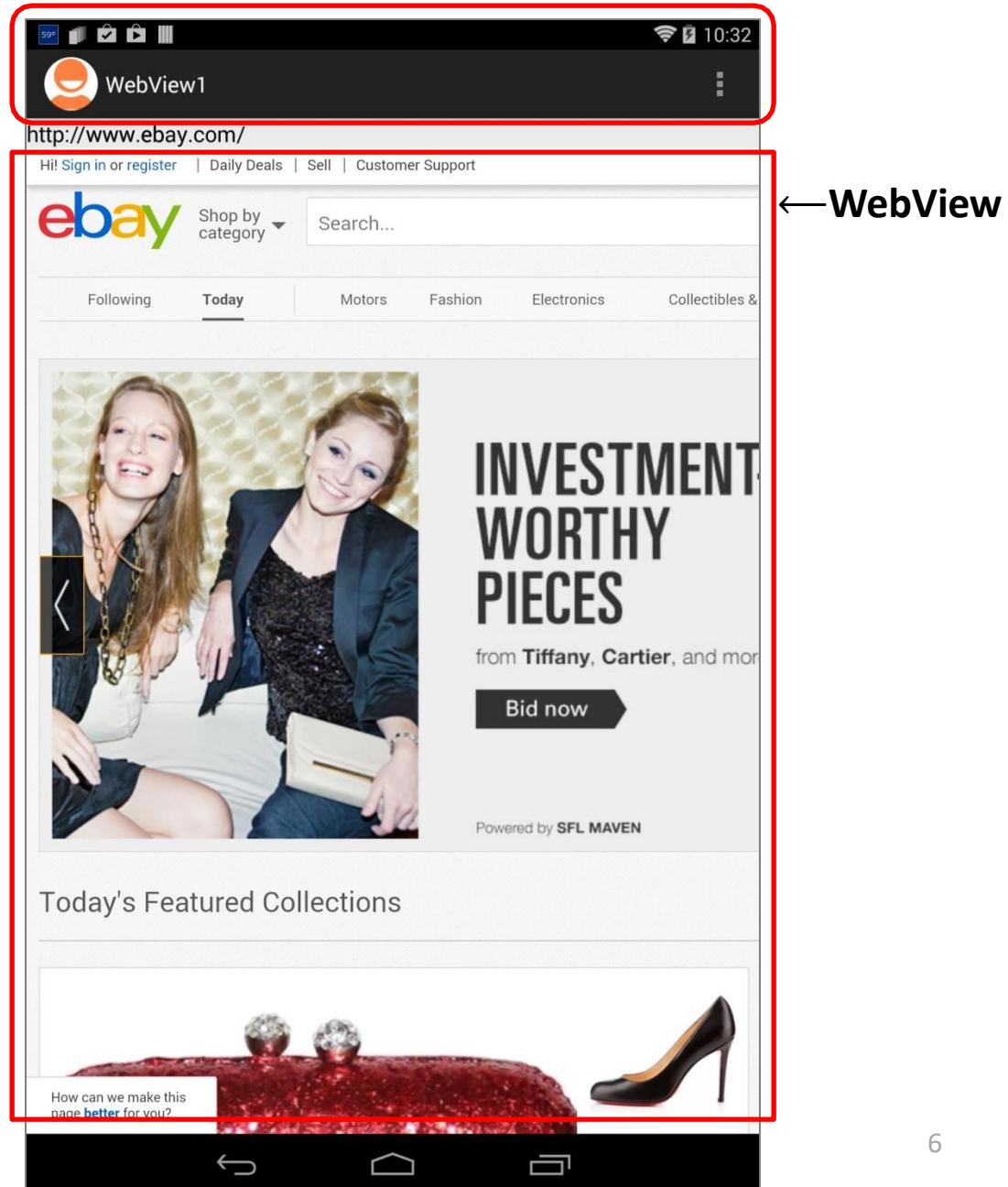


Przeglądarka prezentująca
stronę internetową
(<http://m.ebay.com>)

Aplikacja Androida wykorzystująca
WebView by wyświetlić tą samą
stronę.

WebView

Aplikacja działająca pod kontrolą tabletu. Oryginalna wersja strony www.ebay.com zostaje zaprezentowana (bez przekierowania jak w przypadku telefonu komórkowego).





Uprawnienia

Uwaga! By aktywność miała dostęp do internetu należy dodać uprawnienie **INTERNET** w pliku **Manifestu**. Jeżeli w aplikacji wykorzystywane są dodatkowe biblioteki jak np. **Google Maps**, należy dodać klauzulę <uses-library...>.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="csu.matos.webview_demo1"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <uses-permission android:name="android.permission.INTERNET"/>

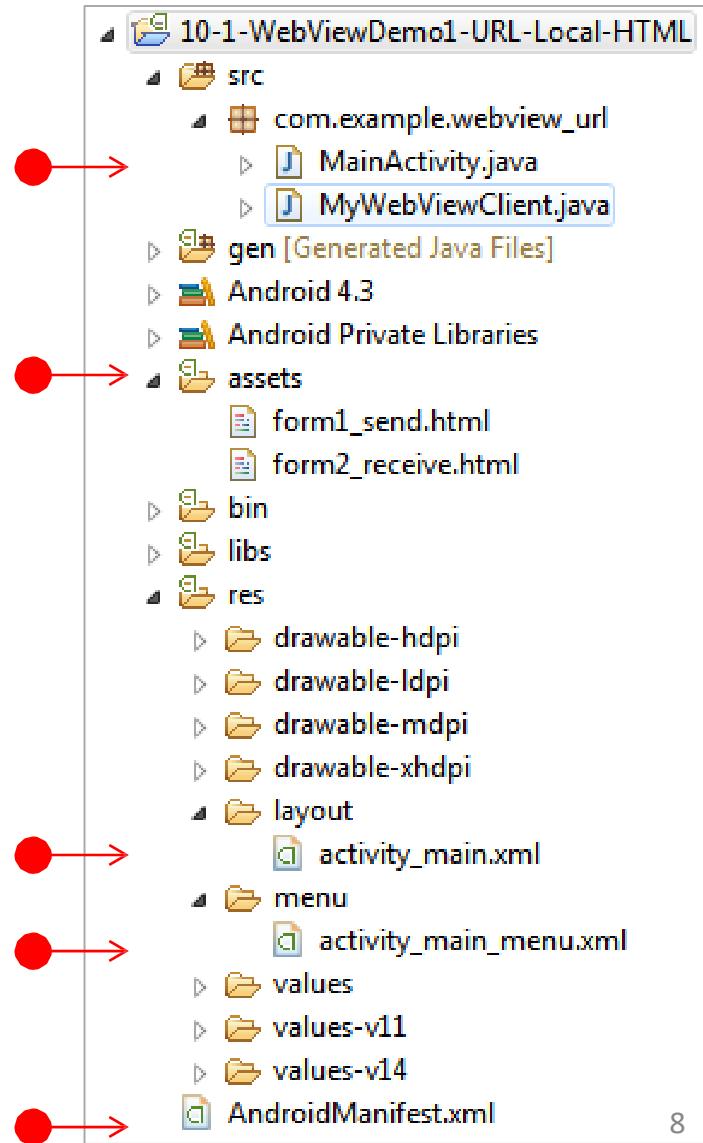
    <application
        ...
        <activity
            ...
            </activity>
        <uses-library android:name="com.google.android.maps" />
    </application>

</manifest>
```

Przykład 1. Wykorzystanie WebView

W tym przykładzie zostaną zaprezentowane 3 sposoby ładowania zasobów do komponentu WebView

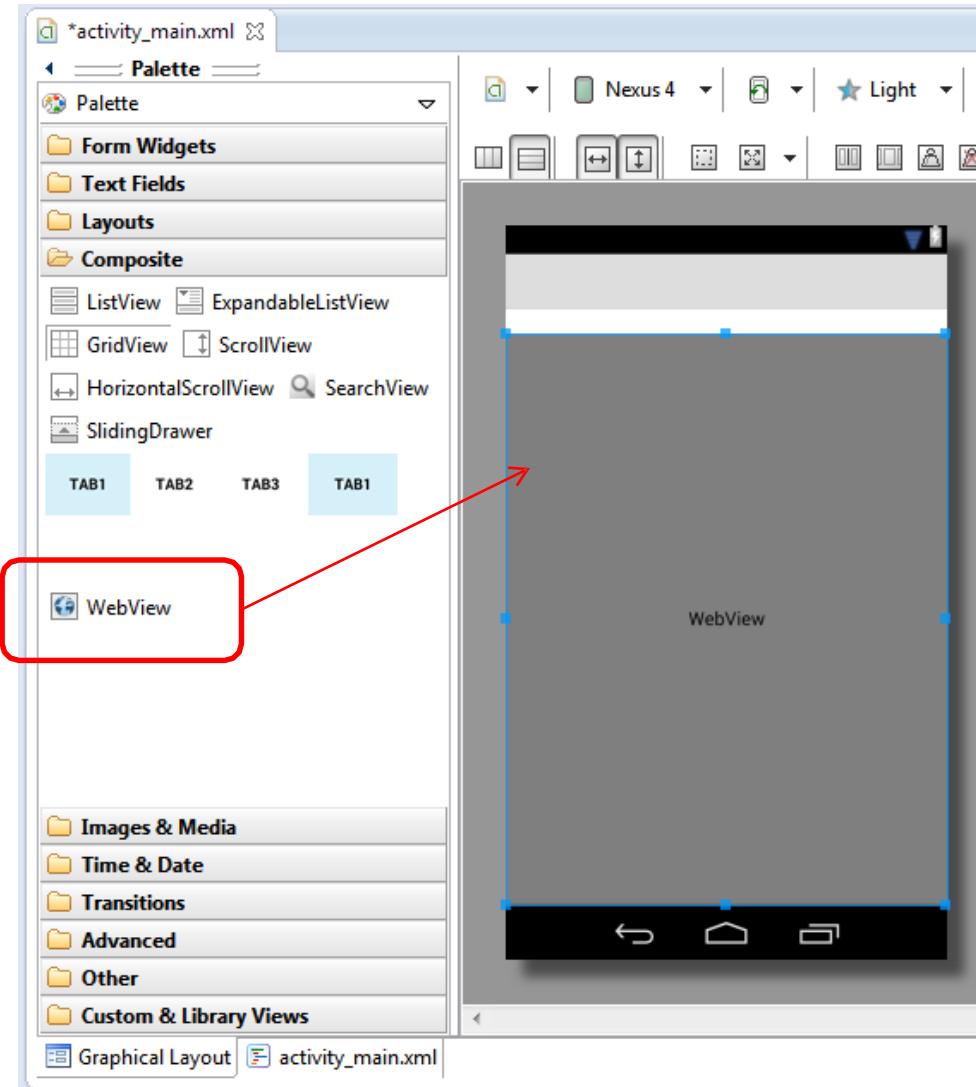
1. W pierwszym przypadku, źródłem danych jest strona internetowa znajdująca się na zdalnym serwerze.
2. W drugim przypadku, komponent WebView służy do wyświetlenia danych pochodzących z lokalnie zapisanych stron internetowych.
3. Trzeci przykład prezentuje sposób na dynamiczne doładowywanie danych z serwera.



WebView

Przykład 1. Wykorzystanie WebView

Najprościej rozpocząć od wyboru komponentu **WebView** z dostępnej palety komponentów.



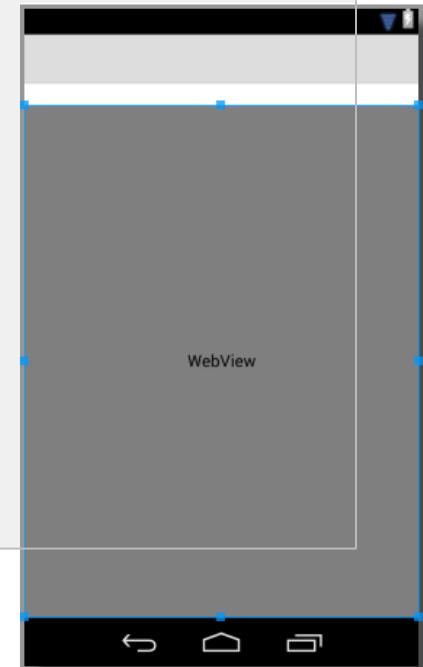
WebView

Przykład 1. Układ – activity_main.xml

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

    <TextView android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium" />

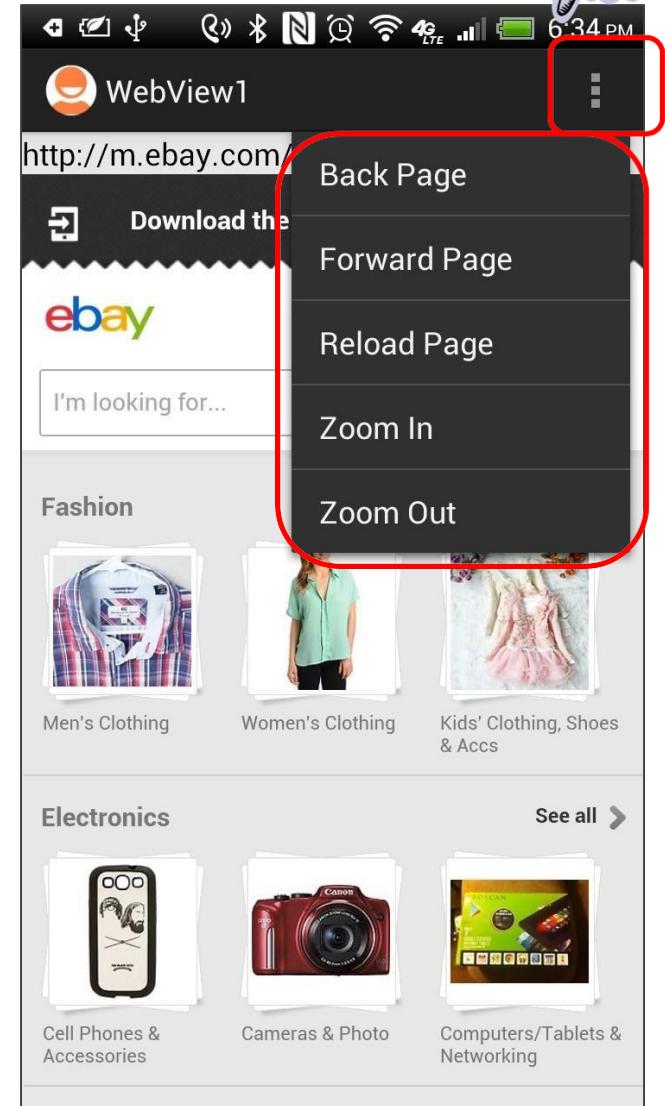
    <WebView android:id="@+id/webview1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
    />
</LinearLayout>
```



WebView

Przykład 1. Menu główne – activity_main_menu.xml

```
<menu  
    xmlns:android="http://schemas.android.com/apk/res/android">  
    <item  
        android:id="@+id/back_page"  
        android:orderInCategory="100"  
        android:showAsAction="never"  
        android:title="Back Page"/>  
    <item  
        android:id="@+id/forward_page"  
        android:orderInCategory="110"  
        android:showAsAction="never"  
        android:title="Forward Page"/>  
    <item  
        android:id="@+id/reload_page"  
        android:orderInCategory="120"  
        android:showAsAction="never"  
        android:title="Reload Page"/>  
    <item android:id="@+id/zoom_in"  
        android:orderInCategory="130"  
        android:showAsAction="never"  
        android:title="Zoom In"/>  
    <item  
        android:id="@+id/zoom_out"  
        android:orderInCategory="140"  
        android:showAsAction="never"  
        android:title="Zoom Out"/>  
</menu>
```



WebView

Przykład 1. MainActivity.java

```
public class MainActivity extends Activity {  
    TextView txtMsg;  
    WebView webview;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        txtMsg = (TextView) findViewById(R.id.txtMsg);  
  
        // Try demo1, demo2, or demo3 (please, uncomment one at the time!!!)  
        demo1TrySpecificUrl();  
  
        // demo2TryLocallyStoredHtmlPage();  
        // demo3TryImmediateHtmlPage();  
        // demo4TryRichGoogleMap();  
    } // onCreate  
  
    // -----  
    // Demo1, Demo2 and Demo3 code goes here...  
    // -----  
}  
//>MainActivity
```



WebView

Przykład 1. MainActivity.java - Zdalna strona www

```
@SuppressLint("SetJavaScriptEnabled")
private void demo1TrySpecificUrl() {

    webview = (WebView) findViewById(R.id.webview1);
    webview.getSettings().setJavaScriptEnabled(true);

    webview.setWebViewClient(new MyWebViewClient(txtMsg, "ebay.com"));

    webview.loadUrl("http://www.ebay.com");

}
```

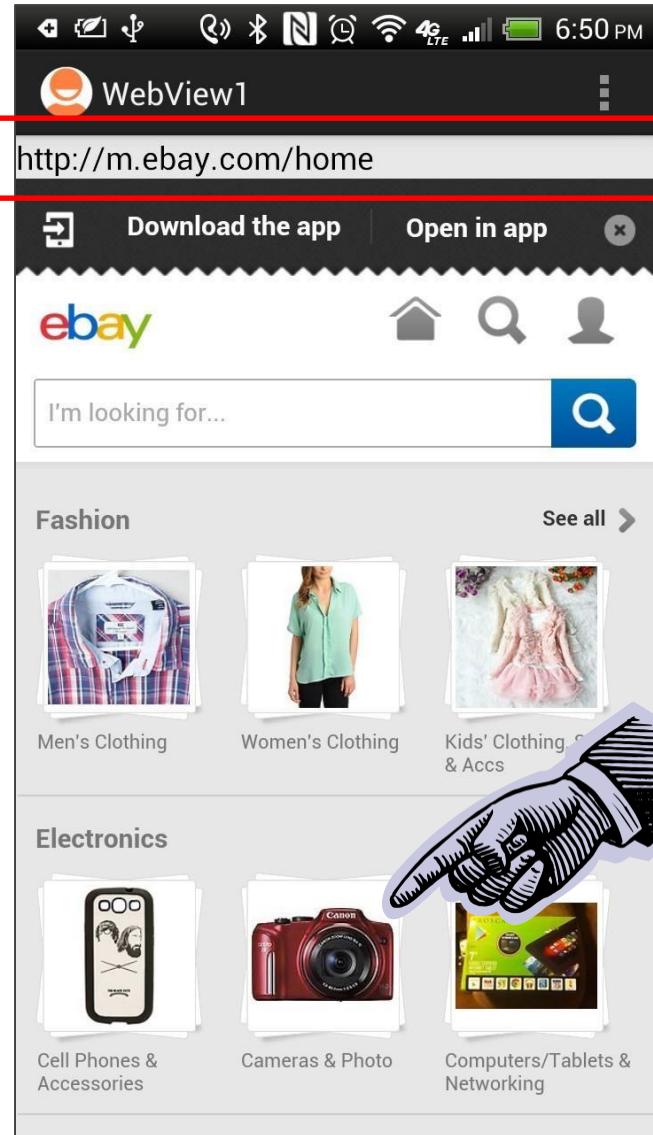
Bezpośrednie
podłączenie do
ebay.com

Komentarz

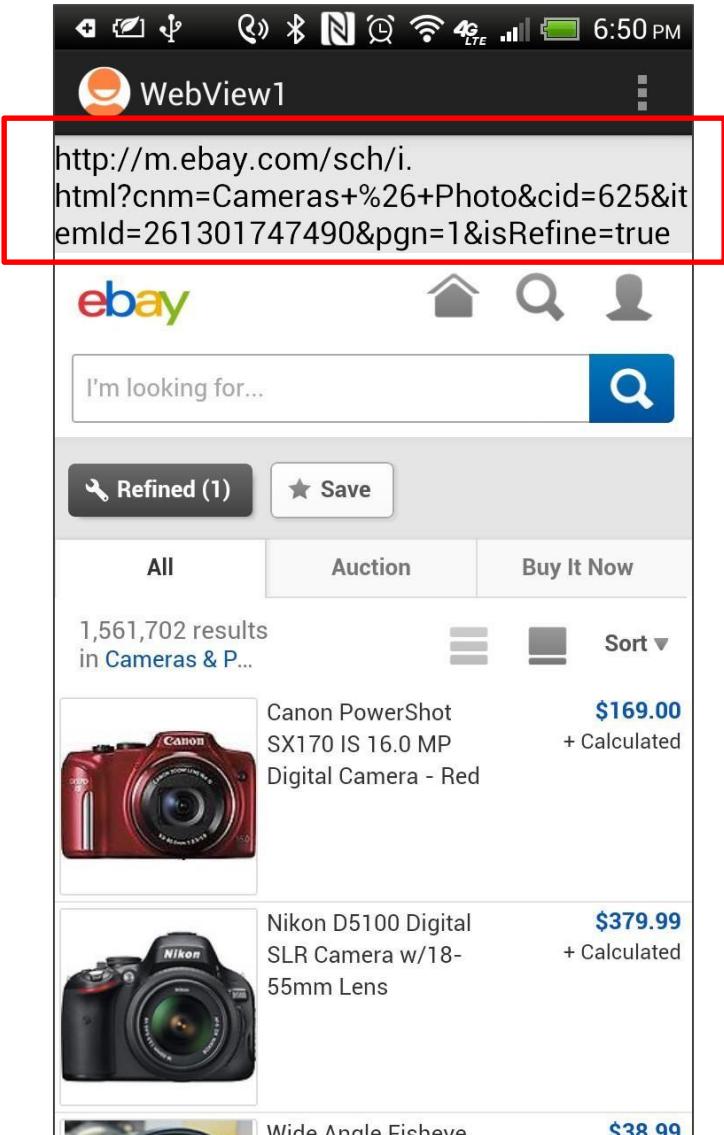
1. Zezwól stronie na wykonywanie skryptów JavaScript (jeśli takowe są wymagane)
2. Klasa **MyWebViewClient** odpowiada za kontrolę komunikacji sieciowej i aktualizację GUI wskazując adres URL. Tylko strony z domeną ebay.com mogą być wyświetlane w WebView – pozostałe mogą być pokazane tylko korzystając z wbudowanej przeglądarki internetowej.
3. Zadany **URL** rozpoczyna proces nawigacji. Jednakże ostateczny URL może być inny niż wskazany poprzez wymuszone przekierowanie przez zdalny serwer. Przykładowo wejście na www.ebay.com skutkuje przekierowaniem na stronę m.ebay.com

WebView

Przykład 1. MainActivity.java



←
Przekierowanie



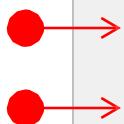
Dalsza
nawigacja
wymuszona
przez
użytkownika

WebView

Przykład 1. MainActivity.java - Lokalna strona html

```
@SuppressLint("SetJavaScriptEnabled")
private void demo2TryLocallyStoredHtmlPage() {
    webview = (WebView) findViewById(R.id.webview1);
    webview.getSettings().setJavaScriptEnabled(true);

    webview.setWebViewClient(new WebViewClient());
    webview.loadUrl("file:///android_asset/form1_send.html");
}
```

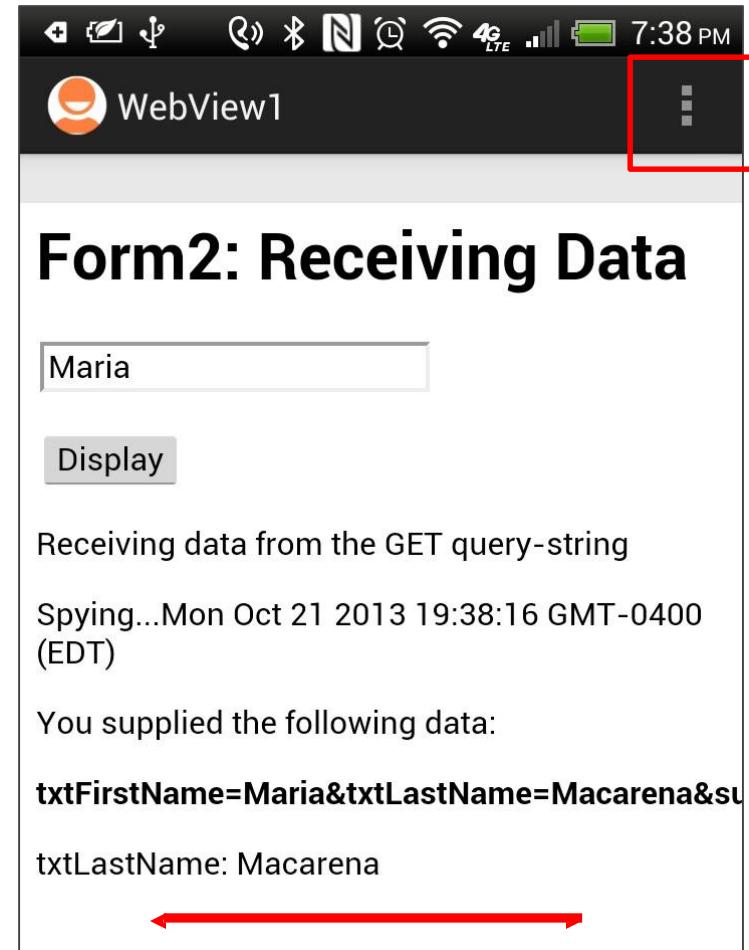
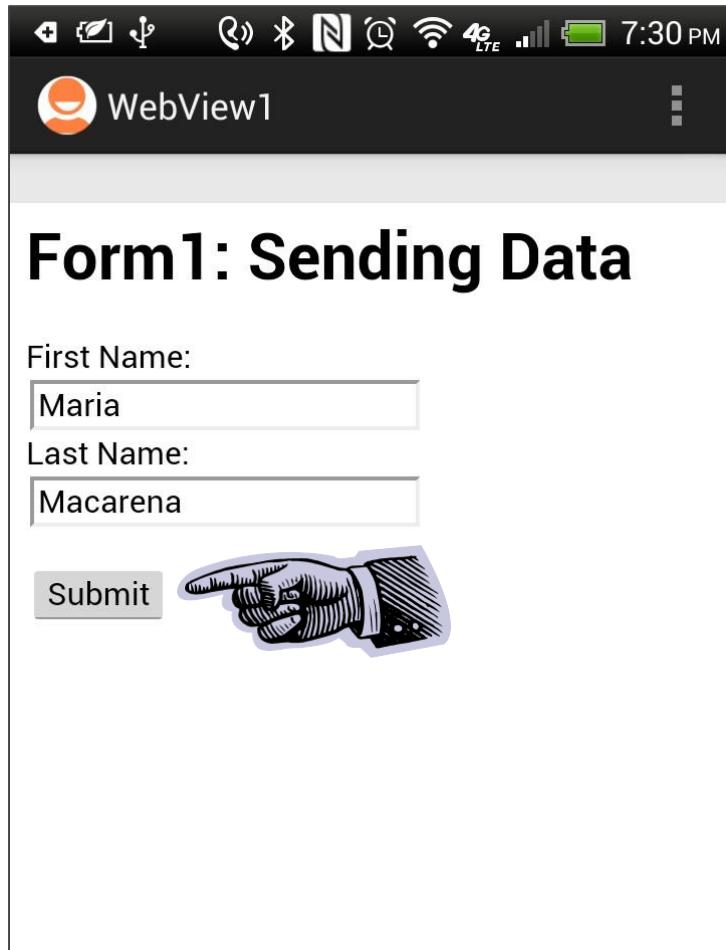


Komentarz

1. Wbudowany obiekt **WebViewClient** używany jest do kontroli nawigacji – dozwolone jest wyświetlanie dowolnych stron internetowych w ramach komponentu WebView.
2. Strona internetowa przechowywana jest w zasobach lokalnych aplikacji (ale może być również przechowywana np. na karcie pamięci). Zwykle strony przechowywane są w katalogu **/assets/** (na tym samym poziomie co **/res/** i **/java/**).

WebView

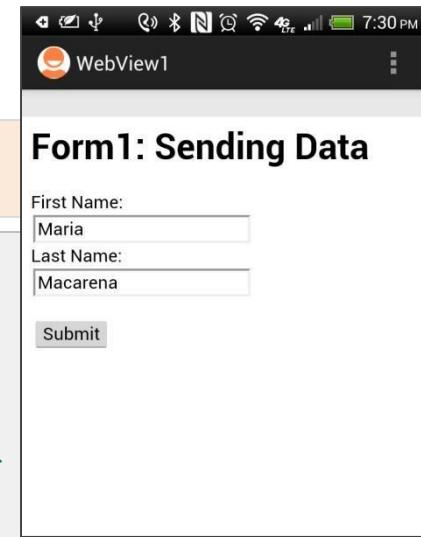
Przykład 1. MainActivity.java - Lokalna strona html



WebView

Przykład 1. form1_send.html - Lokalna strona html

```
<html>
<head>
</head>
<body>
    <FORM action="form2_receive.html" id=form1 method=GET name=form1>
        <h1>Form1: Sending Data</h1>
        First Name:
        <br>
        <INPUT id="txtFirstName" name="txtFirstName" value="Maria">
        <br>
        Last Name:
        <br>
        <INPUT id="txtLastName" name="txtLastName" value="Macarena">
        <p>
        <INPUT type="submit" value="Submit" id=submit1 name=submit1>
    </FORM>
</body>
</html>
```



Komentarz

Po wciśnięciu **Submit** wywoływana jest strona *form2_receive.html*. Zawartość formularza zostanie przekazana z wykorzystaniem metody **HTTP GET**:

txtFirstName=Maria&txtLastName=Macarena&submit1=Submit

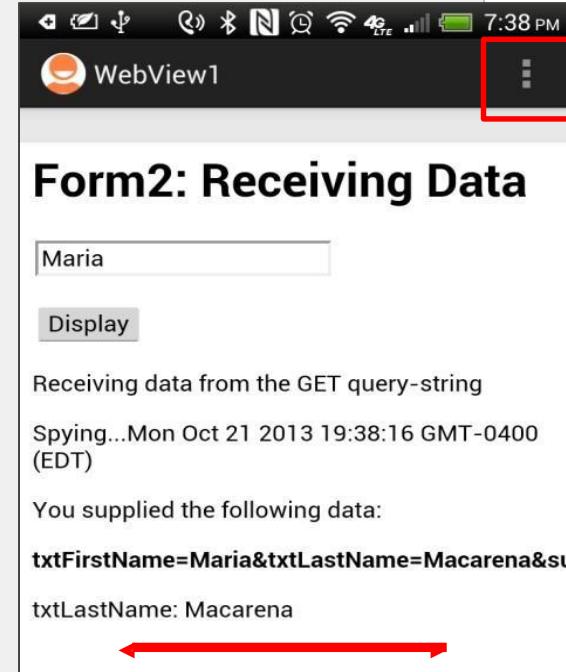
WebView

Przykład 1. form2_receive.html - Lokalna strona html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>Android & Plain HTML Demo</title>
    <script language="javascript">

        function extracting(variable) {
            var query = window.location.search.substring(1);
            alert(query);
            var personName = getQueryVariable(variable)
            document.getElementById("myMsg").value = personName;
        }

        function getQueryVariable(variable) {
            var query = window.location.search.substring(1);
            var vars = query.split('&');
            for (var i=0;i<vars.length;i++) {
                var pair = vars[i].split("=");
                if ((pair[0] == variable) || (variable == 0)) {
                    return pair[1];
                }
            }
            alert('Query Variable ' + variable + ' not found');
        }
    </script>
```



WebView

Przykład 1.form2_receive.html – Lokalna strona html

```
</head>
<body>
<h1>Form2: Receiving Data</h1>

<p><input TYPE="text" id="myMsg" ></p>

<p><input TYPE="button" value="Display" onClick="extracting('txtFirstName');"></p>



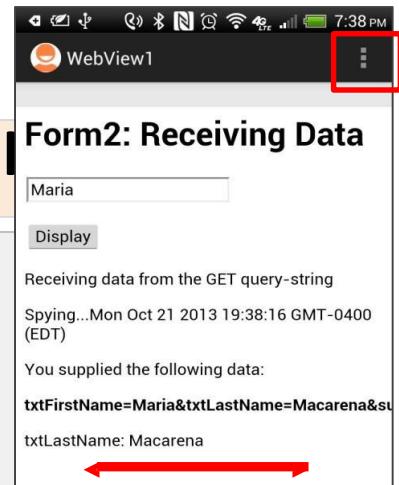
Receiving data from the GET query-string
<script>
    document.write("<p>Spying..." + new Date() );
    document.write("<p> You supplied the following data:");
    var querystring = window.location.search.substring(1);
    document.write("<p><b>" + querystring + "</b>");

    document.write("<p>txtLastName: " + getQueryVariable('txtLastName') );

</script>

</body>
</html>


```



WebView

Przykład 1. form2_receive.html – Lokalna strona html

Komentarz

Właściwość **window.location.search** zwraca dane przekazane przez formularz (wraz ze znakiem ?):

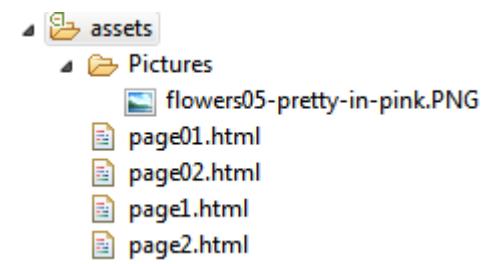
```
window.location.search =  
?txtFirstName=Maria&txtLastName=Macarena&Submit1=Submit
```

Zatem

window.location.search.substring(0) to ‘?’ natomiast

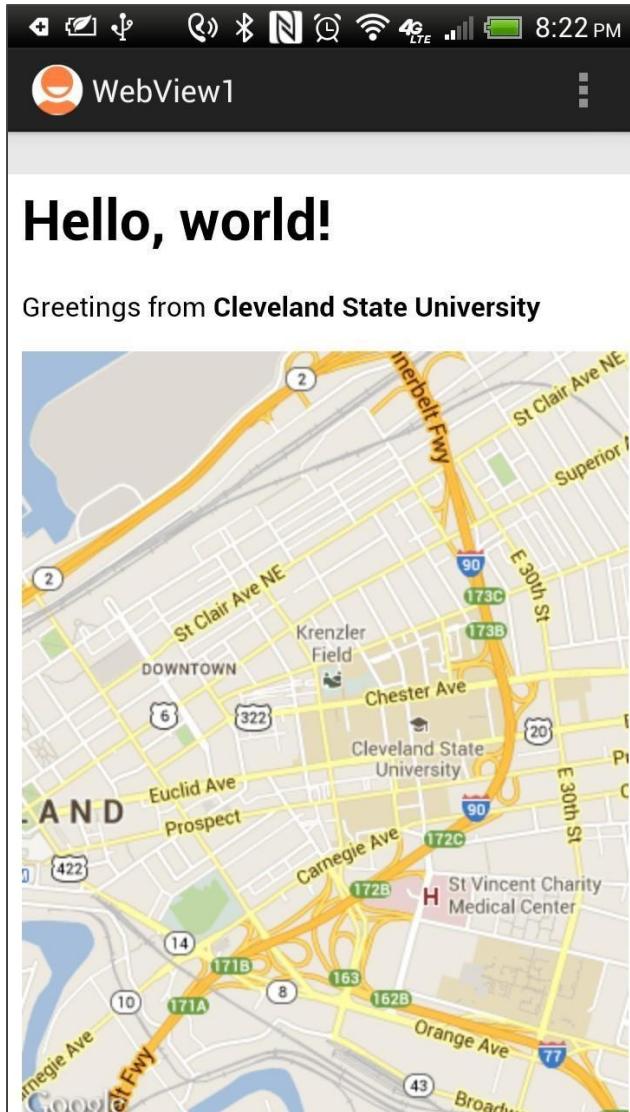
window.location.search.substring(1) to
txtFirstName=Maria&txtLastName=Macarena&submit1=Submit

Można dodać **obrazy** do formularza. Dodaj podkatalog **assets/Pictures**. Odwołanie wygląda następująco
``



WebView

Przykład 1. MainActivity.java - Ładowanie dynamiczne



Wykorzystanie Map Google do lokalizowania miejsc na podstawie współrzędnych:

<http://maps.googleapis.com/maps/api/staticmap?center=41.5020952,-81.6789717&zoom=14&size=350x450&sensor=false>

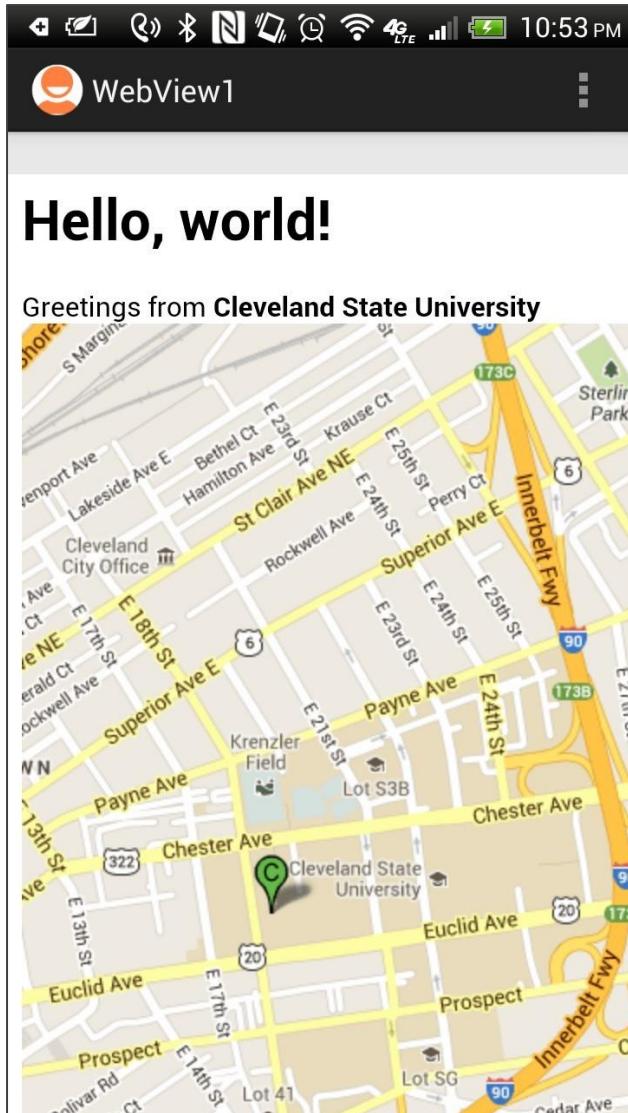
Umożliwia to wyświetlenie statycznej mapy wycentrowanej na podstawie podanych współrzędnych (długości i szerokości geograficznej)

Przykład pokaże jak skopiować to zachowanie korzystając z aplikacji na system Android.



Zdjęcie wygenerowano korzystając z **Google Maps API**. Więcej informacji pod adresem:
<https://developers.google.com/maps/>

Przykład 1. MainActivity - Ładowanie dynamiczne



Wykorzystanie Map Google do lokalizowania miejsc na podstawie nazwy:

[http://maps.googleapis.com/maps/api/staticmap
?center=Cleveland+State+University,Ohio
&zoom=15&size=480x700
&maptype=roadmap
&markers=color:green|label:C|41.5020952,-
81.6785000
&sensor=false](http://maps.googleapis.com/maps/api/staticmap?center=Cleveland+State+University,Ohio&zoom=15&size=480x700&maptype=roadmap&markers=color:green|label:C|41.5020952,-81.6785000&sensor=false)

Umożliwia to wyświetlenie statycznej mapy drogowej wycentrowanej na podstawie podanej nazwy danego miejsca.

Zdjęcie wygenerowano korzystając z **Google Maps API**. Więcej informacji pod adresem:
<https://developers.google.com/maps/>

Przykład 1. MainActivity.java - Ładowanie dynamiczne

```
@SuppressLint("SetJavaScriptEnabled")
private void demo3TryImmediateHtmlPage() {

    webview = (WebView) findViewById(R.id.webview1);
    webview.getSettings().setJavaScriptEnabled(true);
    webview.setWebViewClient(new WebViewClient());

    String aGoogleMapImage = "<img src=\"http://maps.googleapis.com/maps/api/"
        + "staticmap?center=41.5020952,-81.6789717&\""
        + "zoom=14&size=350x450&sensor=false\"> ";

    String myLocalHtmlPage =
        "<html> "
        + "<body>"
        + "<h1>Hello, world! </h1>"
        + "<p> Greetings from <b>Cleveland State University</b>"
        + "<p>" + aGoogleMapImage
        + "</body> "
        + "</html>";

    webview.loadData(myLocalHtmlPage, "text/html", "UTF-8");
}
```

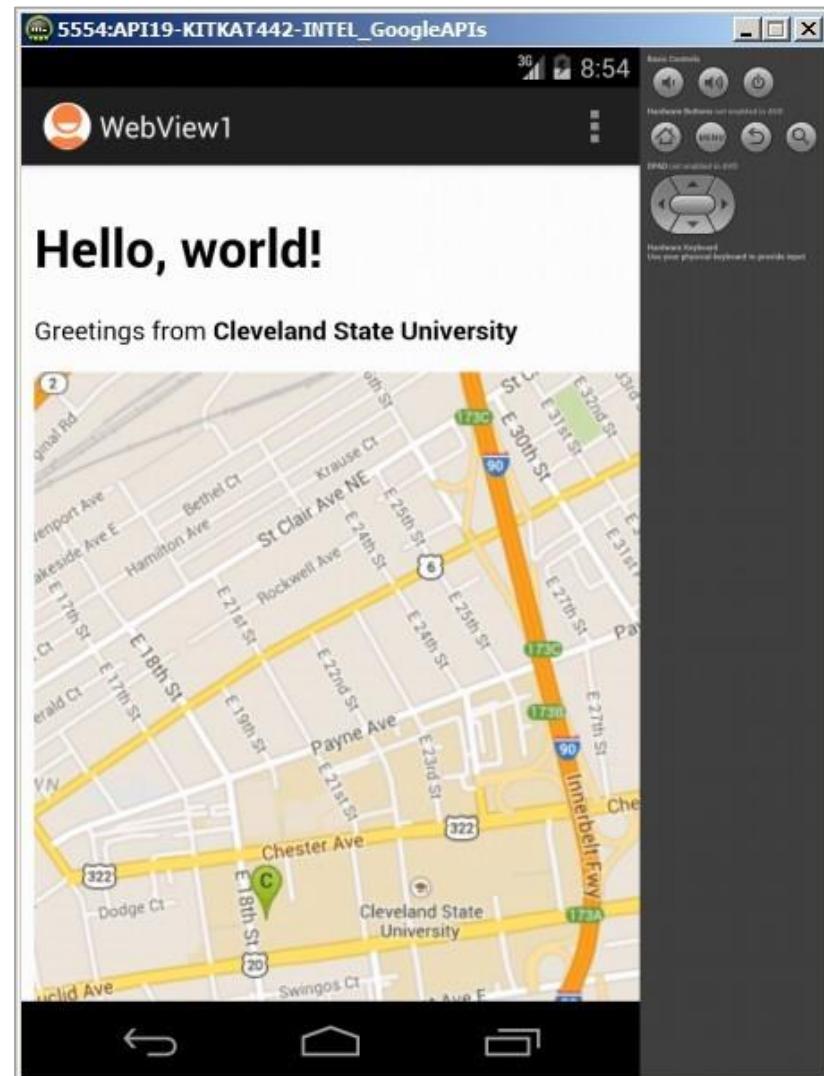
Przykład 1. Ładowanie dynamiczne

Uwaga!



Sprawdź, czy twoja aplikacja:

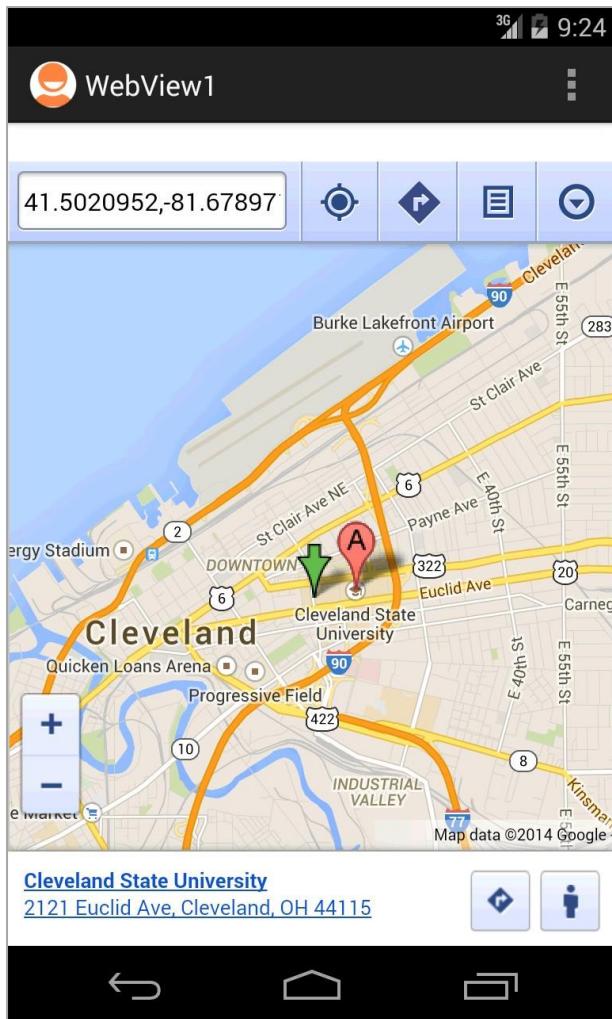
1. Ma odpowiednie **uprawnienia** zdefiniowane w pliku **Manifestu** (Internet oraz com.google.android.maps są wymagane)
2. Posiada wkompilowaną bibliotekę Google.API.
3. Emulator został prawidłowo skonfigurowany (z *Google APIs (x86 System Image)*)



WebView

Przykład 1. MainActivity.java - Mapy raz jeszcze

Tym razem zostanie wykorzystana biblioteka **Google Map API V3**



```
@SuppressLint("SetJavaScriptEnabled")
private void demo4TryRichGoogleMap() {
    webview = (WebView) findViewById(R.id.webView1);

    webview.getSettings().setJavaScriptEnabled(true);
    //show only WebViews
    webview.setWebViewClient(new WebViewClient());

    String mapUrl = "https://maps.google.com/maps"
        + "?q=41.5020952,-81.6789717&t=m&z=13";

    webview.loadUrl( mapUrl );
}
```

Tworzona jest „bogatsza” wersja mapy, posiadająca wbudowane kontrolki by przełączyć się na tryb street-view, zmienić powiększenie itp. Więcej informacji pod: <https://developers.google.com/maps/documentation/javascript/basics#Mobile>

Przykład 1. Mapy raz jeszcze

Uwaga:

Występuje drobna różnica względem poprzedniego przykładu

<http://maps.google.com/>

Wykorzystuje Google Maps V2

<http://maps.googleapis.com>

Wykorzystuje Google Maps V3

Biblioteka V3 jest szybsza od wersji V2 oraz została stworzona z myślą o urządzeniach mobilnych. Dodatkowo biblioteka w starszej wersji nie będzie dalej rozwijana

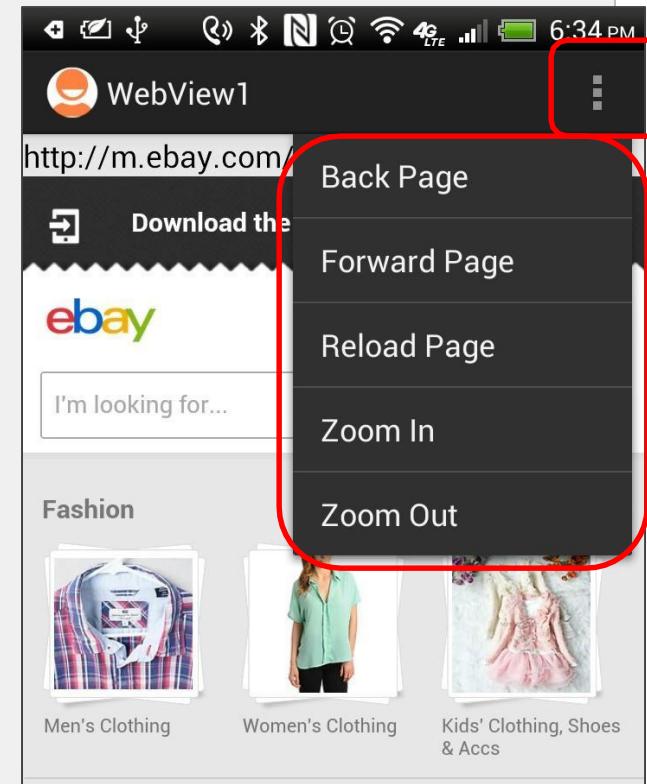
(zobacz <https://www.youtube.com/v/zl8at1EmJjA>)

WebView

Przykład 1. MainActivity.java - Menu

```
// browser navigation implemented through the option-menu
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.activity_main_menu, menu);
    return true;
}// onCreateOptionsMenu

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    String option = item.getTitle().toString();
    if (option.equals("Forward Page"))
        webview.goForward();
    if (option.equals("Back Page"))
        webview.goBack();
    if (option.equals("Reload Page"))
        webview.reload();
    if (option.equals("Zoom In"))
        webview.zoomIn();
    if (option.equals("Zoom Out"))
        webview.zoomOut();
    return true;
}// onOptionsItemSelected
}// class
```



Przykład 1. MyWebViewClient.java – Kontrola nawigacji

```
class MyWebViewClient extends WebViewClient {  
    // this object keeps a history-log of all visited links  
    // and validates url links against a "home-base" server address  
    Context context;  
    TextView txtMsg;  
    String hostServerSuffix; //this is the home-base  
  
    public MyWebViewClient(TextView txtMsg, String hostServerSuffix) {  
        this.txtMsg = txtMsg;  
        this.hostServerSuffix = hostServerSuffix;  
    }  
  
    @Override  
    public boolean shouldOverrideUrlLoading(WebView view, String url) {  
        context = view.getContext();  
  
        String host = Uri.parse(url).getHost();  
        // if 'host' portion extracted above keeps us inside valid  
        // base server (hostServerSuffix) then use a WebView to continue  
        // navigation, otherwise override navigation by showing requested  
        // page inside a full blown-up browser (security issues here...)  
  
        String text = "URL:" + url.toString()  
            + "\n\nHOST:" + host  
            + "\n\nHOME should be:" + hostServerSuffix;
```

WebView

Przykład 1. MyWebViewClient.java – Kontrola nawigacji

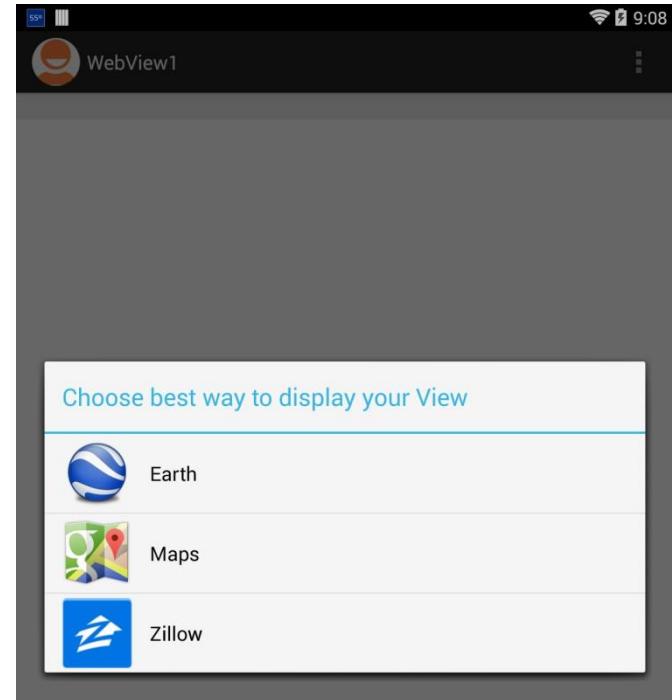
```
Toast.makeText(context, text, Toast.LENGTH_LONG).show();  
  
    if (url.contains(hostServerSuffix)) {  
        // do not override navigation (using big web-browser) as long as  
        // url points to some page in my defined home-server  
        return false;  
  
    } else {  
        // The supplied url is not for a page on my 'valid' site,  
        // in that case launch another Activity that handles URLs  
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));  
        context.startActivity(intent);  
        return true;  
    }  
}//shouldOverrideUrlLoading  
  
@Override  
public void onPageStarted(WebView view, String url, Bitmap favicon) {  
    // keep user informed - update txtMsg with current URL value  
    super.onPageStarted(view, url, favicon);  
    txtMsg.setText(url.toString());  
}//onPageStarted  
  
}//MyWebViewClient
```

Przykład 1. MyWebViewClient.java – Kontrola nawigacji

Co się stanie jeżeli nie zostanie wykorzystany obiekt typu **WebViewClient**?

Jeżeli **WebViewClient** nie zostanie wykorzystany, komponent **WebView** poprosi menedżera aktywności by obsługiwać URL (jeżeli dzieje się to po raz pierwszy, prezentowana jest lista kandydujących aplikacji)

Rysunek po prawej stronie prezentuje taką listę dla przykładu z mapami. Zawiera ona wszystkie zainstalowane aplikacje, dedykowane do obsługi map jak Google-Earth, Google-Maps, czy Zillow.



Zadaniem **WebViewClient** jest kontrola dostępu do adresów URL:

- jeżeli aktualny adres jest „akceptowany” zwraca on wartość FALSE i komponent **WebView** przejmuje odpowiedzialność za wyświetlenie strony,
- W przeciwnym przypadku aktywność decyduje o sposobie wyświetlania danych – zwykle poprzez wykorzystanie przeglądarki internetowej.

Uwaga – Wykorzystanie JavaScript i animacji Flash

Aplikacja **musi** jawnie nadać uprawnienie by umożliwić wykonanie kodu **JavaScript** na odwiedzanych stronach. Domyślnie interpretacja *Javascript* jest **wyłączona**. Aktywacja wygląda następująco:

→ `webview.getSettings().setJavaScriptEnabled(true);`

By usunąć ostrzeżenie o potencjalnych, niepożądanych skutkach dla bezpieczeństwa przez wykorzystanie JS, do każdej metody wykorzystującej JavaScript można dodać:

→ `@SuppressLint("SetJavaScriptEnabled")`

Uwaga – Wykorzystanie JavaScript i animacji Flash

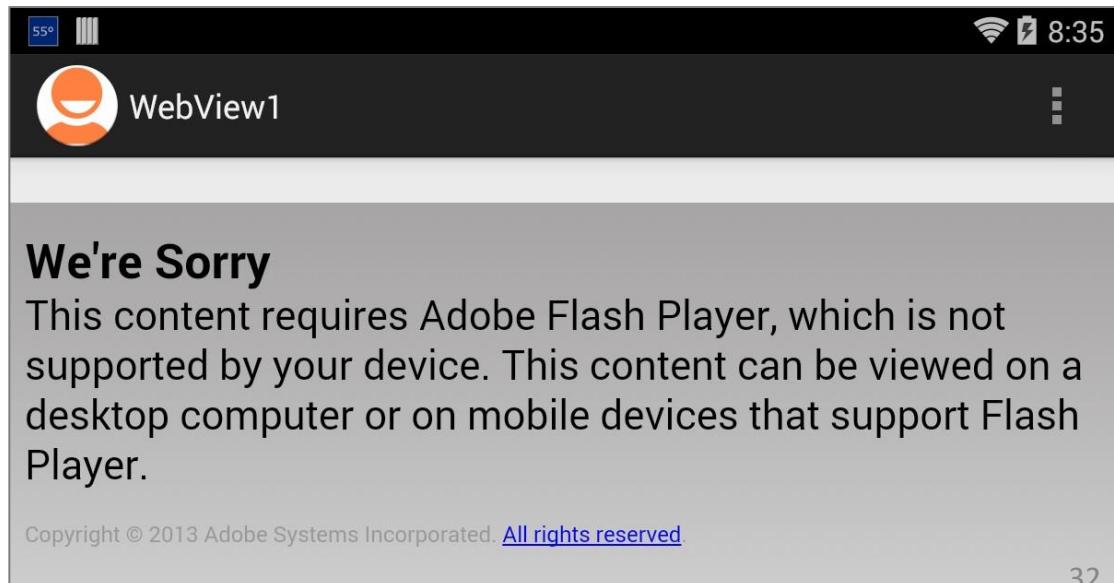
Wykorzystanie wtyczek ma status deprecated, zatem będzie niedozwolone w przyszłości. Należy unikać instrukcji typu:

→ `webview.getSettings().setPluginState(PluginState.ON);`

Przy próbie załadowania strony zawierającej animacje flash:

→ `webview.loadUrl("http://get.adobe.com/flashplayer/");`

najczęściej otrzymamy:



Często wykorzystywane metody WebView

Podane poniżej metody są przydatne podczas realizacji zadań związanych z przeglądaniem zasobów internetowych. Zazwyczaj są implementowane jako opcje menu głównego bądź inne elementy typu przycisk.

- **reload()** odśwież aktualnie przeglądaną stronę.
- **goBack()** cofnij się o jeden krok w historii (użyj **canGoBack()** by sprawdzić czy takie postępowanie jest możliwe).
- **goForward()** przesuń się o jedną pozycję do przodu (użyj **canGoForward()** by określić czy taki krok jest możliwy).
- **goBackOrForward()** przesuń się w przód lub wstecz, gdzie *ujemne/dodanie* wartości numeryczne parametru określają o ile kroków należy się przesunąć.
- Metoda **canGoBackOrForward()** umożliwia określenie czy można cofnąć się/przesunąć dalej o zadaną liczbę kroków.
- **clearCache()** wyczyść pamięć podręczną.
- **clearHistory()** wyczyść historię.
- **ZoomIn()** , **ZoomOut()** zmień powiększenie lub pomniejszenie.

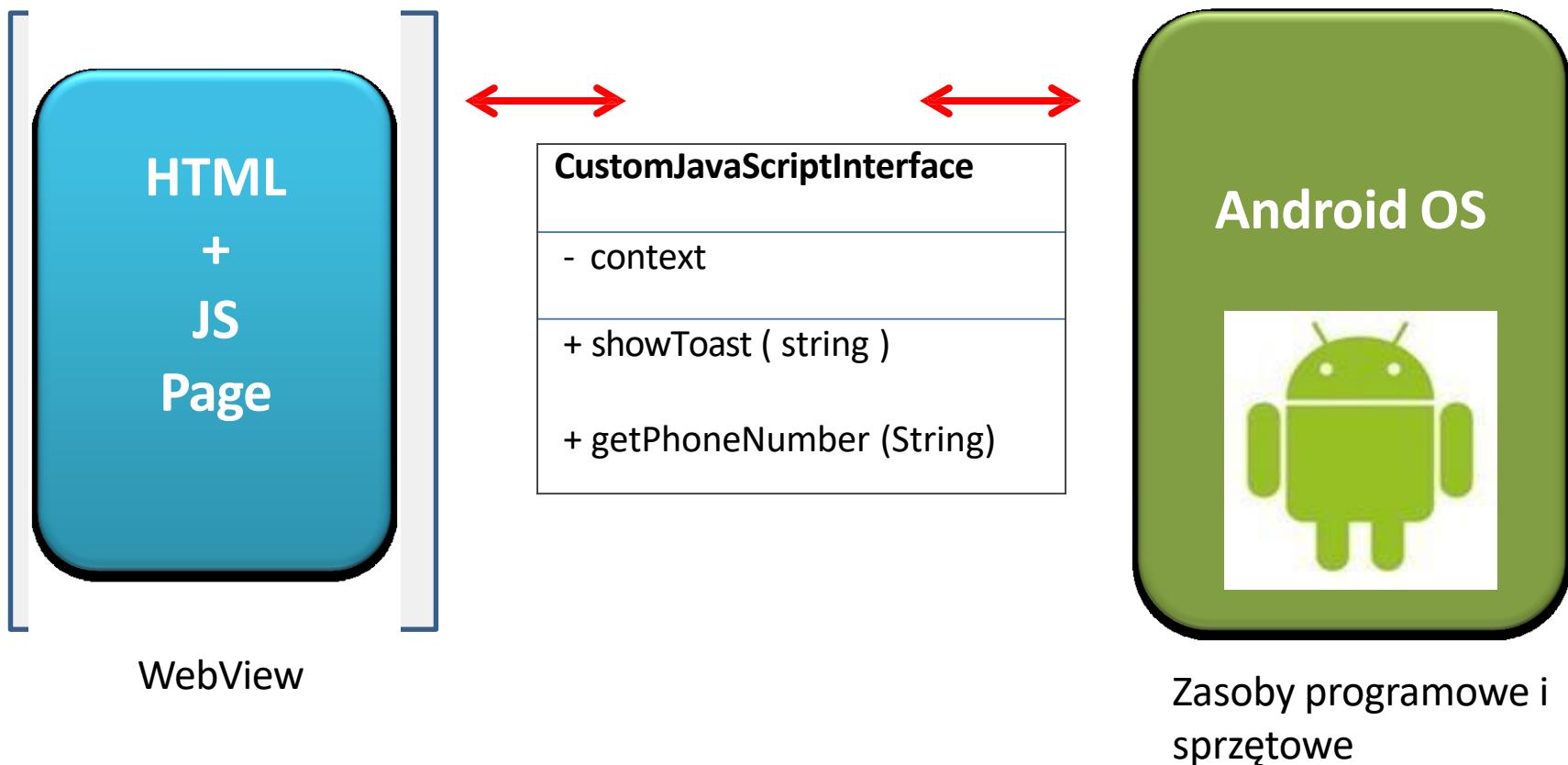
Przykład 2. Aplikacje hybrydowe

Poniższy przykład prezentuję kombinację wykorzystania kilku technologii (html/javascript + java) do stworzenia aplikacji mobilnej.

1. Część wizualna aplikacji składa się z dynamicznych stron internetowych w języku HTML. Lokalne strony wykorzystują język JavaScript do implementacji ich działania. Prezentowane są z wykorzystaniem komponentu WebView.
2. Obiekt łączący jest tworzony jako składowa aplikacji Android i przekazywany do wykorzystania w komponencie WebView.
3. Obiekt łączący pozwala na interakcję zasobów html ze wszelkimi metodami czy zasobami sprzętowymi dostępnymi po stronie języka Java. Dostęp odbywa się z poziomu języka skryptowego, poprzez wywołanie odpowiednich metod obiektu łączącego.

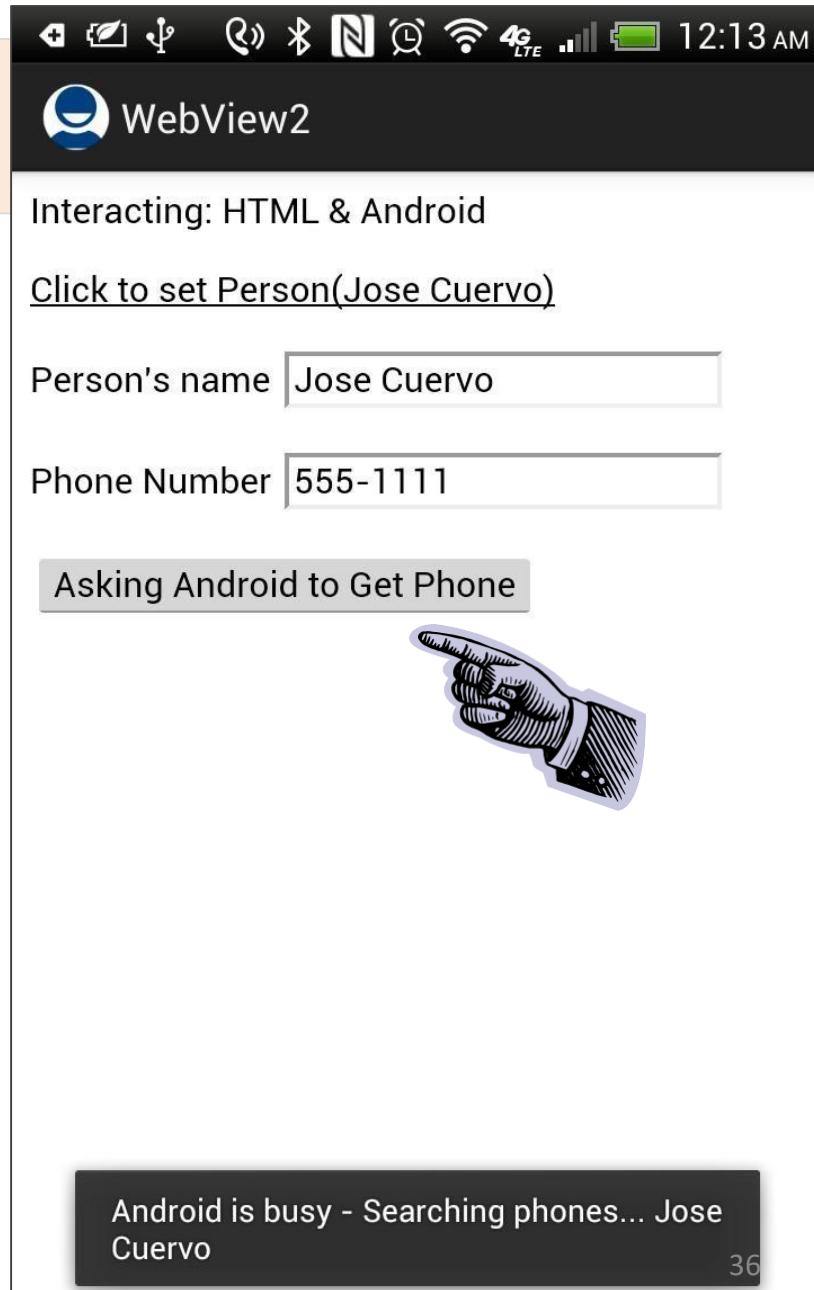
Przykład 2. Aplikacje hybrydowe

Metody w obiekcie łączącym powinny być publiczne by kod skryptu mógł je wywoływać.



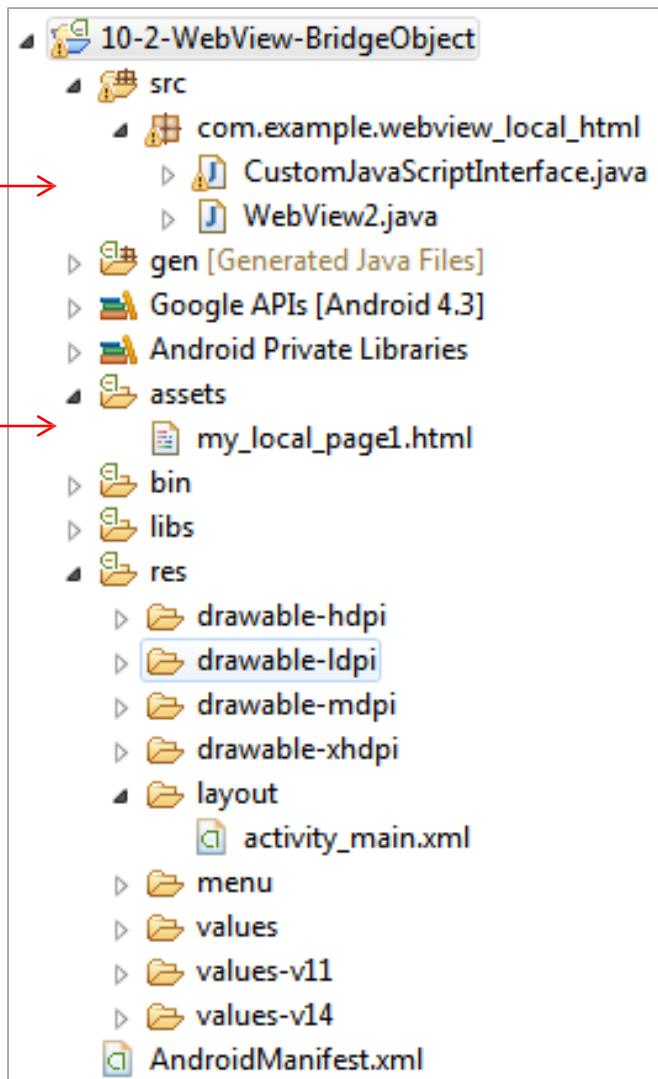
Przykład 2. Aplikacje hybrydowe

1. Aplikacja prezentuje przechowywaną lokalnie stronę internetową w komponentie WebView.
2. Aplikacja oczekuje wpisania nazwy użytkownika i przekazuje ją dalej.
3. Android dokonuje przeszukania bazy kontaktów i zwraca numer telefonu przypisany do danego kontaktu.
4. Użytkownikowi prezentowany jest komunikat typu Toast.



Przykład 2. Aplikacje hybrydowe

Struktura aplikacji



Układ: activity_main.xml

```
<RelativeLayout  
    xmlns:android="http://schemas.android.com/apk/re  
s  
    /android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" >  
  
    <WebView android:id="@+id/webView1"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:layout_alignParentLeft="true" />  
  
</RelativeLayout>
```

WebView

Przykład 2. MainActivity - WebView2.java

```
public class WebView2 extends Activity {  
  
    @SuppressLint("SetJavaScriptEnabled")  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        WebView webview = (WebView) findViewById(R.id.webView1);  
        webview.getSettings().setJavaScriptEnabled(true);  
  
         webview.addJavascriptInterface(new CustomJavaScriptInterface(this),  
                                         "HtmlAndroidBridge");  
  
         // if the HTML file is in the app's memory space use:  
        webview.loadUrl("file:///android_asset/my_local_page1.html");  
  
        // if the HTML files are stored in the SD card, use:  
        // webview.loadUrl("file:///sdcard/my_local_webpage1.html");  
  
        // CAUTION: Manifest must include  
        // <uses-permission android:name="android.permission.INTERNET"/>  
        // <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>  
  
    } //onCreate  
  
} //class
```

Przykład 2. CustomJavaScriptInterface.java

```
public class CustomJavaScriptInterface {  
    private Context context;  
  
    CustomJavaScriptInterface(Context context) {  
        // remember the application's context  
        this.context = context;  
    }  
  
    @JavascriptInterface  
    public void showToast(String toastMsg) {  
        // instead of dull JavaScript alert() use flashy Toasts  
        Toast.makeText(context, toastMsg, Toast.LENGTH_SHORT).show();  
    }  
  
    @JavascriptInterface  
    public String getPhoneNumber(String person) {  
        // accept a person's name and fake that you are  
        // searching the Android's Contacts Database  
        person = person.toLowerCase();  
        if (person.equals("jose cuervo"))  
            return "555-1111";  
        else if (person.equals("macarena"))  
            return "555-2222";  
        else  
            return "555-3333";  
    }  
}
```

Przykład 2. CustomJavaScriptInterface.java

Komentarz

Klasa **CustomJavaScriptInterface** stanowi most między zawartością HTML oraz systemem Android. Programista decyduje jakie dane są dostępne oraz definiuje odpowiednie akcesory czy modyfikatory.

Obiekt tworzony jest w języku Java a referencja do niego jest przekazywana do środowiska skryptowego poprzez:

```
webview.addJavascriptInterface( new CustomJavaScriptInterface(this),  
                               "HtmlAndroidBridge" );
```

Kod skryptu zawarty jest w stronach HTML (wyświetlanych przez komponent webview) i uzyskuje dostęp do współdzielonych metod poprzez:

```
HtmlAndroidBridge.nazwaMetody(arg)
```

Od API 17, tego typu metody muszą posiadać adnotację [@JavascriptInterface](#) (inaczej są ignorowane).

Przykład 2. HTML – my_local_page1.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Android GetPhone Demo</title>
<script language="javascript">
function setPerson() {
    // move name 'Jose Cuervo' to input box
    document.getElementById("myName").value = 'Jose Cuervo' ;
}
function askAndroidToFindPhone() {
    // bridge object used to send local (html) data to android app
    // passing a person's name, waiting for phone to be returned var person =
    document.getElementById("myName").value; HtmlAndroidBridge.showToast('Android is
    busy - Searching phones... ' + person); document.getElementById("myPhone").value =
    HtmlAndroidBridge.getPhoneNumber(person);
}
</script>
</head>
<body>
<p>Interacting: HTML & Android</p>
<p><a onClick="setPerson()"><u>Click to set Person(Jose Cuervo)</u></a></p>
<p> Person's name <input type="text" id="myName" />
<p> Phone Number <input type="text" id="myPhone" />
<p> <input type="button" onclick= "askAndroidToFindPhone()"
        value="Asking Android to Get Phone">
</body>
</html>
```



Przykład 2. HTML – my_local_page1.html

Komentarz

1. Instrukcja:

```
HtmlAndroidBridge.showToast('Android is busy -  
Searching phones... ' + person);
```

która jest wywoływana z metody JS, przekazuje ciąg znaków i nakazuje systemowi Android wyświetlić komunikat typu Toast z zadanym tekstem.

2. Wyrażenie:

```
HtmlAndroidBridge.getPhoneNumber(person);
```

umożliwia przeszukanie książki adresowej pod kątem zadanej osoby. Jeżeli wyszukiwanie się powiedzie, numer telefonu tej osoby jest przekazywany do strony internetowej.

Przykład 3. Wykorzystanie istniejących bibliotek

W tym przykładzie, do aplikacji zostanie dołączona biblioteka **Google Maps API**, która umożliwia wzbogacenie poprzednich przykładów.

Dlaczego?

Zalety systemu Android:

1. Dostęp do natywnych mechanizmów (np. lokalizacyjnych).
2. Łatwe dodanie do sklepu Play.
3. Szybsze tworzenie aplikacji korzystając z mechanizmów RAD.

Zalety Google Maps:

1. Główny kod aplikacji znajduje się na serwerze.
2. Dokonane zmiany dostępne są natychmiast dla użytkowników.
3. Częstsze aktualizacje bezpośrednio od Google.
4. Wieloplatformowość: aplikacja działa na wielu różnych platformach.

Przykład 3. Wykorzystanie istniejących bibliotek

Jakie są założenia?

- Prezentacja mapy danego obszaru z użyciem **Google Maps JavaScript API V3.** (<https://developers.google.com/maps/documentation/javascript/>)
- Mapa powinna być wyposażona w najnowsze funkcje i być aktualna.

Jak to zrealizować?

- Stwórz standardową aplikację internetową **HTML + JS.** Może być ona tworzona z użyciem standardowych narzędzi wspomagających programowanie aplikacji internetowych.
- Stwórz aplikację działającą pod kontrolą systemu Android, która wyświetla wspomnianą stronę internetową w komponencie typu **WebView.**

Przykład 3. Wykorzystanie istniejących bibliotek

Czym jest Google Maps JavaScript API V3 ?

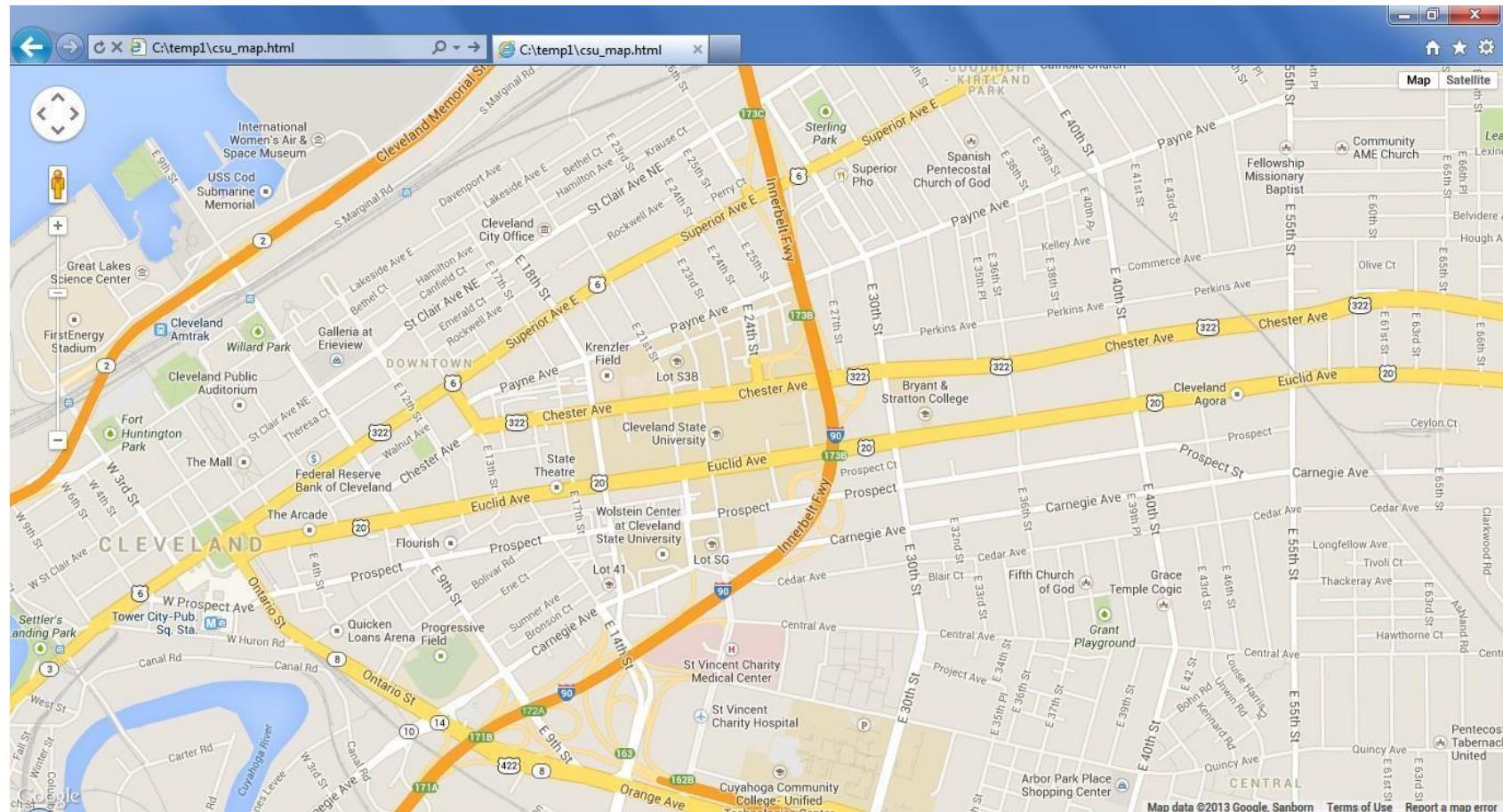


- **Google Maps Javascript API** jest darmową usługą pozwalającą na osadzenie map google we własnych aplikacjach internetowych.
- Została stworzona z myślą o urządzeniach mobilnych (aczkolwiek sprawdza się również w przypadku urządzeń typu komputer stacjonarny)
- Posiada wiele metod do manipulowania wyświetlaniem tego typu danych (tak jak strona internetowa <http://maps.google.com>) oraz dodawania własnych warstw graficznych (np. znaczników pozycji).

Link: <https://developers.google.com/maps/documentation/javascript/>

Przykład 3. Wykorzystanie istniejących bibliotek

Wykorzystanie Google Maps JavaScript API V3



Link: <https://developers.google.com/maps/documentation/javascript/basics?cs=1>

Przykład 3. Wykorzystanie istniejących bibliotek

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
1  <style type="text/css">
  html { height: 100% }
  body { height: 100%; margin: 0px; padding: 0px }
  #map_canvas { height: 100% }
</style>
2  <script type="text/javascript"
    src="http://maps.google.com/maps/api/js?sensor=false">
</script>
<script type="text/javascript">
3    function initialize() {
      var latlng = new google.maps.LatLng(41.5020952, -81.6789717);
      var myOptions = {
        zoom: 15,
        center: latlng,
        mapTypeId: google.maps.MapTypeId.ROADMAP
      };
      var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
    }
</script>
</head>
<body onload="initialize()">
4    <div id="map_canvas" style="width:100%; height:100%"></div>
</body>
</html>
```

google_map.html

Strona wyświetlająca mapę google

Przykład 3. Wykorzystanie istniejących bibliotek

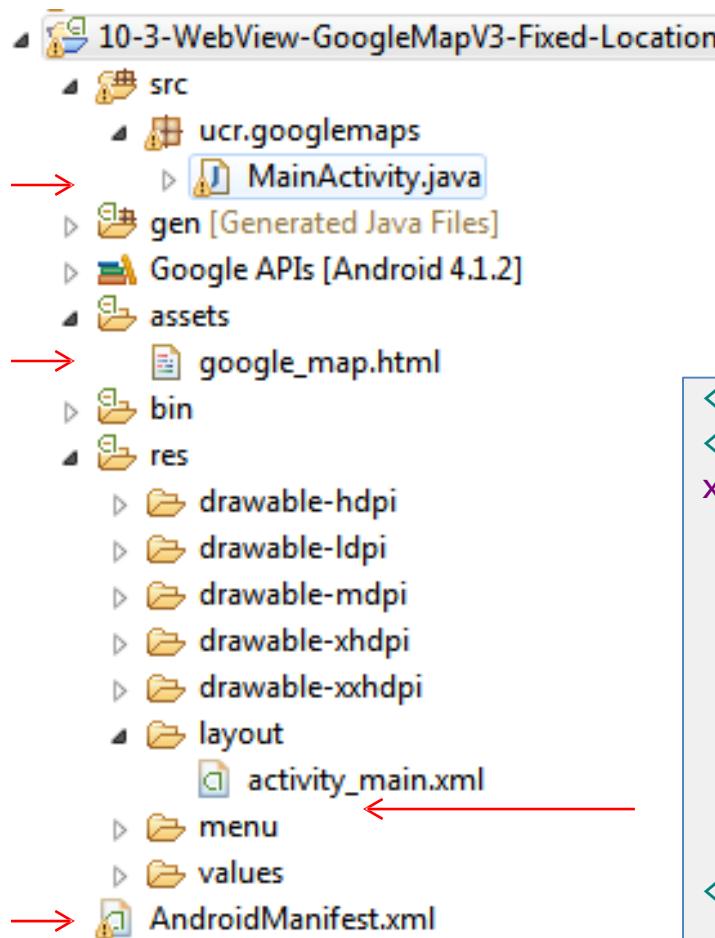
Komentarz

google_map.html to strona HTML odpowiedzialna za rysowanie mapy.

1. Znacznik **<style>** ustawia cały ekran (100% szerokości i wysokości) jako płótno na którym rysowania będzie cała mapa.
2. Pierwszy fragment w **<script>** wywołuje metodę API do rysowania mapy. Nie podano jeszcze konkretnej lokalizacji (mapa nie zostanie jeszcze wyświetlona).
3. Funkcja **Initialize** ustawia przyporządkowanie argumentów. Mapa jest centrowana wokół podanych współrzędnych oraz ustawiany jest poziom powiększenia.
4. Po załadowaniu strony HTML, funkcja **Initialize** jest wywoływana i użytkownikowi prezentowana jest odpowiednia mapa.

Przykład 3. Wykorzystanie istniejących bibliotek

Struktura aplikacji:



Tworzenie aplikacji:

- Umieść komponent **WebView** w pliku `activity_main.xml`.
- Dodaj stronę internetową do folderu **assets**.
- Dodaj uprawnienie **Internet** do pliku manifestu.

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

  <WebView android:id="@+id/webview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>

</LinearLayout>
```

WebView

Przykład 3. Wykorzystanie istniejących bibliotek

```
public class MainActivity extends Activity {  
    WebView webview;  
  
    @SuppressLint("SetJavaScriptEnabled")  
    @Override  
    public void onCreate(Bundle savedInstanceState)  
    { super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        // load into WebView local HTML page defining mapping  
        operation webview = (WebView) findViewById(R.id.webview);  
        webview.getSettings().setJavaScriptEnabled(true);  
  
        webview.loadUrl("file:///android_asset/google_map.html");  
    } //onCreate  
  
} //class
```



Jedynym zadaniem aktywności jest załadowanie określonej strony internetowej.

Przykład 4. Zbieranie danych GPS



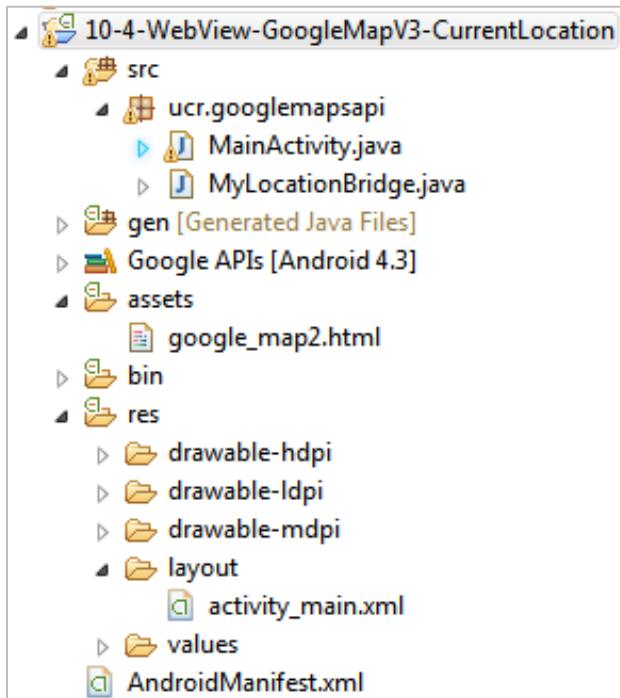
Przykład łączy ze sobą dwa poprzednie:

- Moduł GPS określa rzeczywiste położenie urządzenia mobilnego w danej chwili czasu.
- Obiekt JavaScript przekazuje ustalone współrzędne do strony HTML.
- Strona zawiera kod wyświetlający mapę wycentrowaną na ustalone współrzędne geograficzne.

Rzeczywiste współrzędne geograficzne

Przykład 4. Zbieranie danych GPS

Struktura aplikacji



Układ - activity_main.xml

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

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <WebView android:id="@+id/webview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"/>

</LinearLayout>
```

WebView

Przykład 4. MainActivity.java

```
public class MainActivity extends Activity implements LocationListener {  
    // a LocationListener could use GPS, Cell-ID, and WiFi to establish  
    // a user's location. Deciding which to use is based on choices  
    // including factors such as accuracy, speed, and battery-efficiency.  
  
    private WebView webview;  
    LocationManager locationManager;  
    MyLocationBridge locationBridge = new MyLocationBridge();  
  
    @Override  
    protected void onDestroy() {  
        super.onDestroy();  
        // cut location service requests  
        locationManager.removeUpdates(this);  
    }  
  
    private void getLocation() {  
        locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);  
        Criteria criteria = new Criteria();  
        // criteria.setAccuracy(Criteria.ACCURACY_FINE); //use GPS (you should be outside)  
        criteria.setAccuracy(Criteria.ACCURACY_COARSE); // cell towers, wifi  
        String provider = locationManager.getBestProvider(criteria, true);  
  
        // In order to make sure the device is getting the location, request  
        // updates [wakeup after changes of: 5 sec. or 10 meter]  
        locationManager.requestLocationUpdates(provider, 5, 10, this);  
        locationBridge.setNewLocation(locationManager.getLastKnownLocation(provider));  
    }  
}
```

WebView

Przykład 4. MainActivity.java

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    getLocation();
    setupWebView();
    this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
} //onCreate

// Set up the webview object and load the page's URL
@SuppressWarnings("SetJavaScriptEnabled")
private void setupWebView() {
    final String centerMapURL = "javascript:centerAt("
        + locationBridge.getLatitude() + ","
        + locationBridge.getLongitude() + ")";
    // set up the webview to show location results
    webview = (WebView) findViewById(R.id.webview);
    webview.getSettings().setJavaScriptEnabled(true);
    webview.addJavascriptInterface(locationBridge, "locationBridge");
    webview.loadUrl("file:///android_asset/google_map2.html");
    // Wait for the page to load then send the location information
    webview.setWebViewClient(new WebViewClient() {
        @Override
        public void onPageFinished(WebView view, String url) {
            webview.loadUrl(centerMapURL);
        }
    });
}
```

Przykład 4. MainActivity.java

```
@Override
public void onLocationChanged(Location location) {
    String lat = String.valueOf(location.getLatitude());
    String lon = String.valueOf(location.getLongitude());
    Toast.makeText(getApplicationContext(), lat + "\n" + lon, 1).show();
    locationBridge.setNewLocation(location);
}

@Override
public void onProviderDisabled(String provider) {
    // needed by Interface. Not used
}

@Override
public void onProviderEnabled(String provider) {
    // needed by Interface. Not used
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
    // needed by Interface. Not used
}

}//class
```

Przykład 4. MyLocationBridge.java

```
// An object of type "MyLocationBridge" will be used to pass data back and
// forth between the Android app and the JS code behind the HTML page.

public class MyLocationBridge {
    private Location mostRecentLocation;

    @JavascriptInterface
    public void setNewLocation(Location newCoordinates){
        mostRecentLocation = newCoordinates;
    }
    @JavascriptInterface
    public double getLatitude() {
        if (mostRecentLocation == null) return (0);
        else return mostRecentLocation.getLatitude();
    }

    @JavascriptInterface
    public double getLongitude() {
        if (mostRecentLocation == null) return (0);
        else return mostRecentLocation.getLongitude();
    }
}

// MyLocationFinder
```

Przykład 4. google_map2.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
    <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
    <title>Google Maps JavaScript API v3 Example: Marker Simple</title>

    <style type="text/css">
      html { height: 100% }
      body { height: 100%; margin: 0px; padding: 0px }
      #map_canvas { height: 100% }
    </style>

    <script type="text/javascript"
      src="http://maps.google.com/maps/api/js?sensor=false"></script>
    <script type="text/javascript">

      function initialize() {
        // get latitude and longitude from the BRIDGE object (JavaScriptInterface)
        var myLatlng = new google.maps.LatLng(locationBridge.getLatitude(),
                                              locationBridge.getLongitude());
        var myOptions = {
          zoom: 17,
          center: myLatlng,
          mapTypeId: google.maps.MapTypeId.ROADMAP
        }
      }

    </script>
  </head>
  <body>
    <div id="map_canvas" style="width:100%; height:100%; border: 1px solid black; position: absolute; top: 0; left: 0; z-index: 1; background-color: #f0f0f0;"></div>
    <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; z-index: 2; background-color: black; opacity: 0.5;"></div>
  </body>
</html>
```

Przykład 4. google_map2.html

```
var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);

var marker = new google.maps.Marker({
    position: myLatlng,
    map: map
});

}

</script>

</head>
<body onload="initialize()">
    <div id="map_canvas"></div>
</body>
</html>
```

Przykład bazuje na:

http://code.google.com/apis/maps/articles/android_v3.html

<http://code.google.com/apis/maps/documentation/javascript/overlays.html#MarkerAnimations>

<http://gmaps-samples.googlecode.com/svn/trunk/articles-android-webmap>

WebView

Przykład 4. Testowanie przy użyciu emulatora

DDMS - 10-4-WebView-GoogleMapV3-CurrentLocation/src/ucr/goolgemapsapi/Main.java - Eclipse

File Edit Refactor Source Navigate Search Project Run Window Help

Devices 33

Name: IceCreamMex-16 [emulator-5554] Online

- com.android.deskclock 340
- com.android.systemui 201
- com.android.calendar 313
- com.android.providers.calenc 380
- com.android.phone 229
- com.android.mms 407
- com.android.exchange 467
- android.process.acore 294
- android.process.media 417
- com.android.launcher 244
- system_process 146
- com.android.inputmethod.latin 215
- com.android.settings 270
- com.android.email 449
- com.android.defcontainer 550
- com.android.keychain 566
- com.svox.pico 580
- com.android.quicksearchbox 594
- ucr.googlemapsapi 625
- com.example.playingwithweb 899

Emulator Control

Telephony Status

Voice: home Speed: Full

Data: home Latency: None

Telephony Actions

Incoming number:

Voice
 SMS

Message:

Call Hang Up

Location Controls

Manual GPX KML

Decimal
 Sexagesimal

Longitude: -122.084095

Latitude: 37.422006

Send ←

WebView4

Map Satellite

Amphitheatre Pkwy

Charleston Rd

Charlie's Cafe

Google Bldg 41

Cafe Slice 11

Stan

Google Bldg 40

Google Bldg 42

Google Bldg 43

Google

Map data ©2012 Google - Terms of Use

WebView

Pytania ?

Co dalej?

Google Maps Developers

<https://developers.google.com/maps/documentation/>

Google Maps API – Webservices

<http://code.google.com/apis/maps/documentation/webservices/index.html>

Google Maps JavaScript API V3

<http://code.google.com/apis/maps/documentation/javascript/tutorial.html>

DODATEK A. Sugerowane materiały dodatkowe

Oracle Mobile Application Framework

<http://www.oracle.com/technetwork/developer-tools/maf/overview/index.html>

Tworzenie aplikacji pod Android oraz iOS wykorzystując platformę Oracle MAF

Intel App Framework

<http://app-framework-software.intel.com/>

Biblioteka JavaScript do tworzenia wieloplatformowych aplikacji w HTML5

Wskazówki wydajnościowe dla Geo API

<https://www.youtube.com/v/zl8at1EmJjA>

Dokumentacja Google Maps Developers

<https://developers.google.com/maps/documentation/>

Kurs Google Maps JavaScript API V3

<http://code.google.com/apis/maps/documentation/javascript/tutorial.html>

Tworzenie aplikacji wykorzystujących WebView

<http://developer.android.com/guide/webapps/webview.html>

Debugowanie aplikacji internetowych

<http://developer.android.com/guide/webapps/debugging.html>

Dobre praktyki dla aplikacji internetowych

<http://developer.android.com/guide/webapps/best-practices.html>

DODATEK B. Rzeczywista funkcja do przeszukiwania książki adresowej – prosta, choć mało efektywna wersja

Dodaj do pliku manifestu:

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

```
public String findPhoneNumber(String searchName) {
    //look for first contact entry partially matching searchName

    String name = "n.a.";
    String number = "n.a.";
    //defina an iterator to traverse the contact book
    Cursor phones = getContentResolver().query(
        ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
        null, null, null);

    while (phones.moveToNext()) {
        String NAME = ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME;
        name = phones.getString(phones.getColumnIndex(NAME));
        String NUMBER = ContactsContract.CommonDataKinds.Phone.NUMBER;
        number = phones.getString(phones.getColumnIndex(NUMBER));

        if (name.toLowerCase().contains(searchName.toLowerCase())) {
            return number; // a good match, phone number found
        }
    }
    // Not found
    return "n.a.";
} // findPhoneNumber
```