Programowanie urządzeń mobilnych w systemie Android

Zagadnienia podstawowe

---

dr inż. Robert Arsoba
Robert.Arsoba@ie.tu.koszalin.pl

Koszalin 2011
Wersja 1.0

Tematyka wykładu

Wprowadzenie

- Czym jest system Android?
- Cechy systemu Android
- Architektura systemu Android
- Wirtualna maszyna Dalvik VM
- Pakiety Java w systemie Android
- Wersje systemu Android
- Rynek wersji systemu Android
- Środowisko programowania

Wprowadzenie

Android to system operacyjny firmy Google dla urządzeń mobilnych typu „*handheld*”, wspierany przez zrzeszenie OHA (*Open Handset Alliance*).

pierwszy smartfon z systemem Android
HTC T-Moble G1
2008 rok

tablet
Samsung Galaxy Tab 10.1
2011 rok

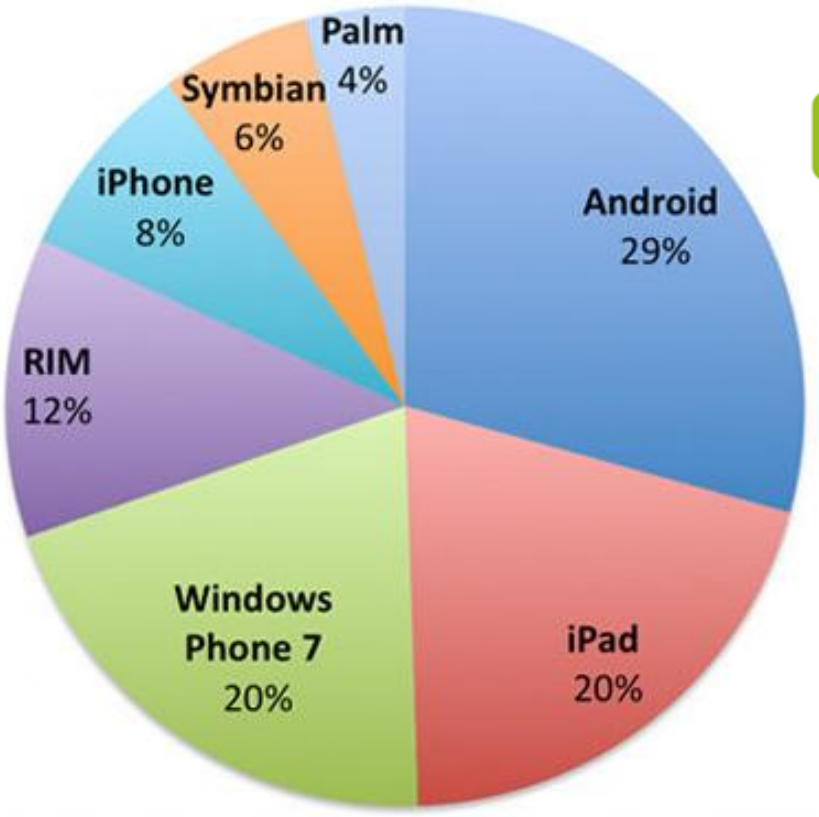
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wprowadzenie

Rynek aplikacji dla urządzeń mobilnych



Prognoza rynku aplikacji mobilnych na 2011 rok

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wprowadzenie

Cechy systemu Android:

- jawny, otwarty kod źródłowy, brak opłat licencyjnych
- jądro Linux 2.6
- pełna wielozadaniowość
- możliwość wykorzystania modelu przetwarzania w chmurze (*cloud computing*)
- otwarte środowisko aplikacji: wielokrotne użycie i wzajemna wymiana komponentów
- wirtualna maszyna Dalvik (DVM), zoptymalizowana dla urządzeń mobilnych
- zintegrowana przeglądarka oparta na silniku WebKit, obsługa HTML 5
- zoptymalizowana grafika (2D: Google Skia, 3D: Open GL ES), wspomaganie sprzętowe
- relacyjna baza danych SQLite
- obsługa multimediów (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- telefonia GSM
- Bluetooth, EDGE, 3G, Wi-Fi
- aparat fotograficzny, GPS, akcelerometr, kompas

} *wsparcie uzależnione od sprzętu*

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wprowadzenie

Architektura systemu Android

AlarmClock
Browser
Calculator
Calendar
Camera
Contacts
Email
GoogleSearch
HTML Viewer
IM

Launcher
Mms
Music
PackageInstaller
Phone
Settings
SoundRecorder
Stk
Sync
Updater
VoiceDialer

sterowniki urządzeń, usługi systemu operacyjnego

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wprowadzenie

Wirtualna maszyna Dalvik VM

W systemie Android kod bajtowy aplikacji jest uruchamiany w wirtualnej maszynie Dalvik.

Cechy Dalvik VM:

- zoptymalizowana do działania na urządzeniach o ograniczonych zasobach (pamięć, moc obliczeniowa, brak pliku wymiany)
- zoptymalizowana do jednoczesnego, wydajnego uruchamiania wielu swoich instancji
- pliki kodu bajtowego klas Java (`*.class`) przetwarzane na kod bajtowy w formacie Dalvik Executable (`*.dex`)
- optymalizacja pamięci (eliminacja informacji powtarzających się w plikach klas, plik DEX ma mniejszy rozmiar w stosunku do pliku JAR)
- generowanie kodu maszynowego oparte na rejestrach, a nie na stosach jak w JVM (redukcja liczby instrukcji)
- wbudowany kompilator JIT (*Just In Time*) od wersji Android 2.2

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wprowadzenie

Pakiety Java w systemie Android – część 1

Środowisko Android SDK zawiera ponad 40 pakietów i 700 klas.
Najważniejsze z nich znajdują się w przestrzeni nazw **`android.*`**.

nazwa pakietu	zawartość i przeznaczenie pakietu
`android.app`	model aplikacji, aktywności, kontrolki, okna dialogowe
`android.bluetooth`	obsługa łączności, urządzeń i funkcji Bluetooth
`android.content`	dostawcy treści, dostęp do magazynu danych, intencje
`android.content.pm`	manager pakietów
`android.content.res`	dostęp do plików zasobów
`android.database`	koncepcje abstrakcyjnej bazy danych
`android.database.sqlite`	koncepcje fizycznej bazy danych SQLite
`android.gesture`	obsługa gestów na ekranie dotykowym
`android.graphics`	klasy podstawowych obiektów graficznych
`android.graphics.drawable`	protokoły rysowania, rysunki tła, animacje
`android.graphics.drawable.shapes`	klasy kształtów do rysowania

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wprowadzenie

Pakiety Java w systemie Android – część 2

nazwa pakietu	zawartość i przeznaczenie pakietu
`android.hardware`	obsługa aparatu fotograficznego
`android.location`	obsługa geolokalizacji
`android.media`	obsługa multimediów (odtwarzanie, nagrywanie), rozpoznawanie twarzy
`android.net`	sieciowe interfejsy API dla gniazd
`android.net.wifi`	obsługa połączeń Wi-Fi
`android.opengl`	klasy grafiki trójwymiarowej OpenGL ES
`android.os`	usługi systemowe (m.in. komunikacja międzyprocesowa, obsługa baterii)
`android.preference`	ujednolicone zarządzanie ustawieniami aplikacji przez użytkowników
`android.provider`	predefiniowani dostawcy treści
`android.sax`	analityczne klasy SAX (interfejs API dla języka XML)
`android.speech`	definicje stałych wykorzystywanych w rozpoznawaniu mowy
`android.speech.tts`	konwersja tekstu na mowę (*text-to-speech*, Pico TTS)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wprowadzenie

Pakiety Java w systemie Android – część 3

nazwa pakietu	zawartość pakietu
android.telephony	klasy do obsługi telefonii
android.telephony.gsm	lokalizacja GSM, obsługa wiadomości SMS
android.telephony.cdma	obsługa telefonii CDMA
android.text	klasy do przetwarzania tekstu
android.text.method	klasy do wprowadzania tekstu w celu kontroli zmian
android.text.style	style tekstu
android.utils	klasy **Log**, **DebugUtils**, **TimeUtils**, **Xml**
android.view	klasy **Menu**,**View**,**ViewGroup**, nasłuchiwacze, wywołania zwrotne
android.view.animation	obsługa animacji metodą przejść, interpolatory
android.view.inputmethod	szkielet metody wprowadzania danych
android.webkit	klasy obsługujące przeglądarkę internetową
android.widget	klasy kontrolek interfejsu użytkownika
com.google.android.maps	klasy obsługujące mapy Google Maps

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wprowadzenie

Wersje systemu Android

wersja	nazwa kodowa	wersja API
Android 4.0	Ice Cream Sandwich	14
Android 3.2	Honeycomb MR2	13
Android 3.1.x	Honeycomb MR1	12
Android 3.0.x	Honeycomb	11
Android 2.3.3 - 2.3.7	Gingerbread MR1	10
Android 2.3.0 - 2.3.2	Gingerbread	9
Android 2.2.x	Froyo	8
Android 2.1.x	Eclair MR1	7
Android 2.0.1	Eclair 0.1	6
Android 2.0	Eclair	5
Android 1.6	Donut	4
Android 1.5	Cupcake	3
Android 1.1	Base 1.1	2
Android 1.0	Base	1

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wprowadzenie

Rynek wersji systemu Android

wersja	nazwa kodowa	API	%
Android 2.3.3 - 2.3.7	Gingerbread MR1	10	43,9
Android 2.2.x	Froyo	8	40,7
Android 2.1.x	Eclair MR1	7	10,7
Android 1.6	Donut	4	1,4
Android 3.2	Honeycomb MR2	13	0,9
Android 3.1.x	Honeycomb MR1	12	0,9
Android 1.5	Cupcake	3	0,9
Android 2.3.0 - 2.3.2	Gingerbread	9	0,5
Android 3.0.x	Honeycomb	11	0,1

Stan na listopad 2011, wg odwiedzin strony Android Market

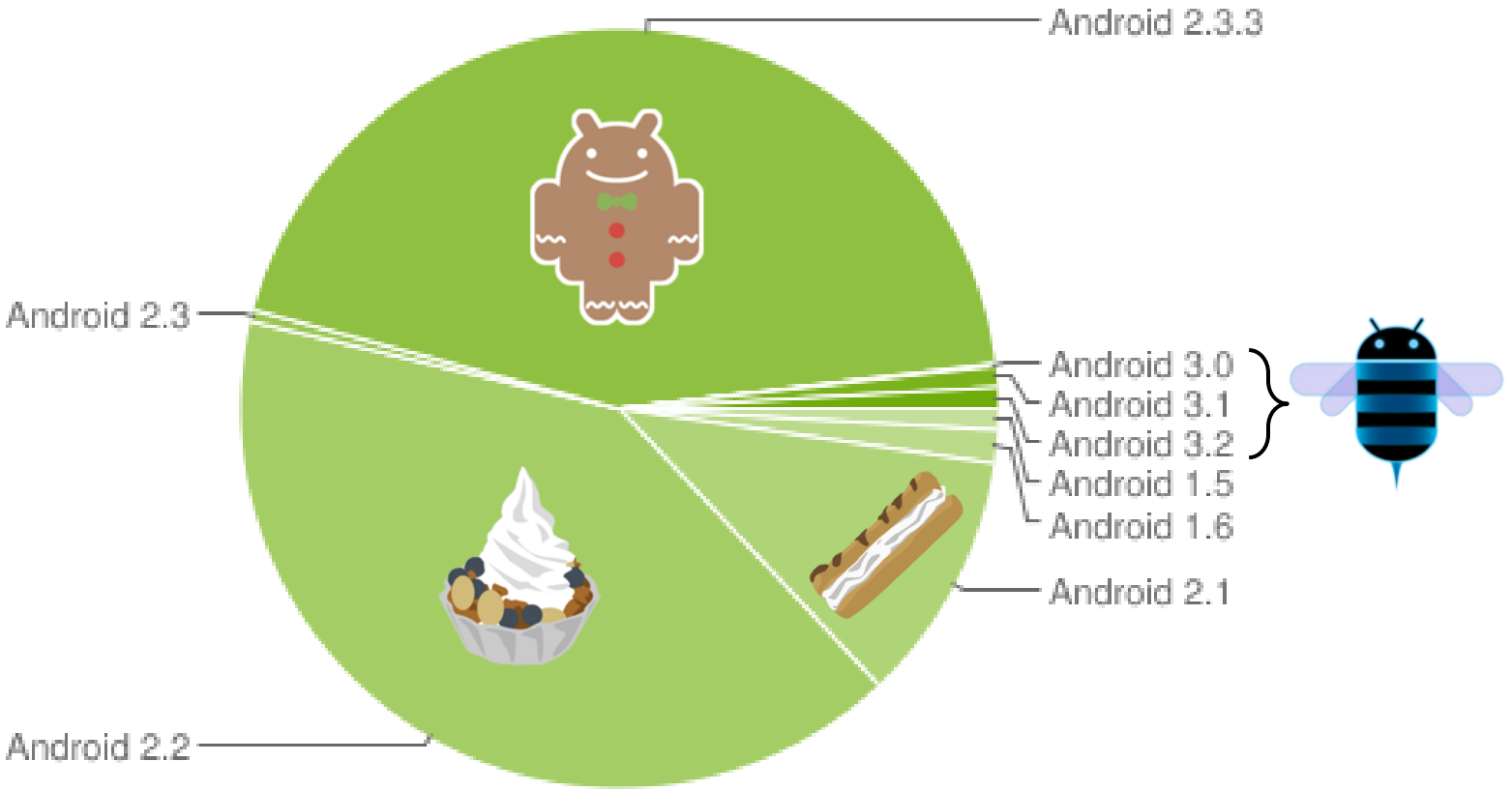
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wprowadzenie

Rynek wersji systemu Android



Stan na listopad 2011, wg odwiedzin strony Android Market

Wprowadzenie

Środowisko programowania

Głównym językiem programowania na platformie Android jest Java. Możliwe jest również tworzenie wstawek w kodzie C/C++ przy pomocy narzędzia Android NDK (*Native Development Tools*).

Wymagane oprogramowanie (patrz ćwiczenie 1):

- Java SE (JDK, min. 5) http://www.oracle.com/technetwork/java/javase/downloads/
- Android SDK http://developer.android.com/sdk/
- Eclipse IDE for Java Developers (min. 3.3) http://www.eclipse.org/downloads/
- ADT (*Android Development Tools*) – wtyczka do Eclipse obsługująca Android SDK

Alternatywne do Eclipse środowiska to:

- narzędzia wiersza poleceń Android SDK
- MOTODEV Studio for Android (Motorola) http://developer.motorola.com/docstools/motodevstudio/download/
- IntelliJ IDEA (JetBrains) http://www.jetbrains.com/idea/download/

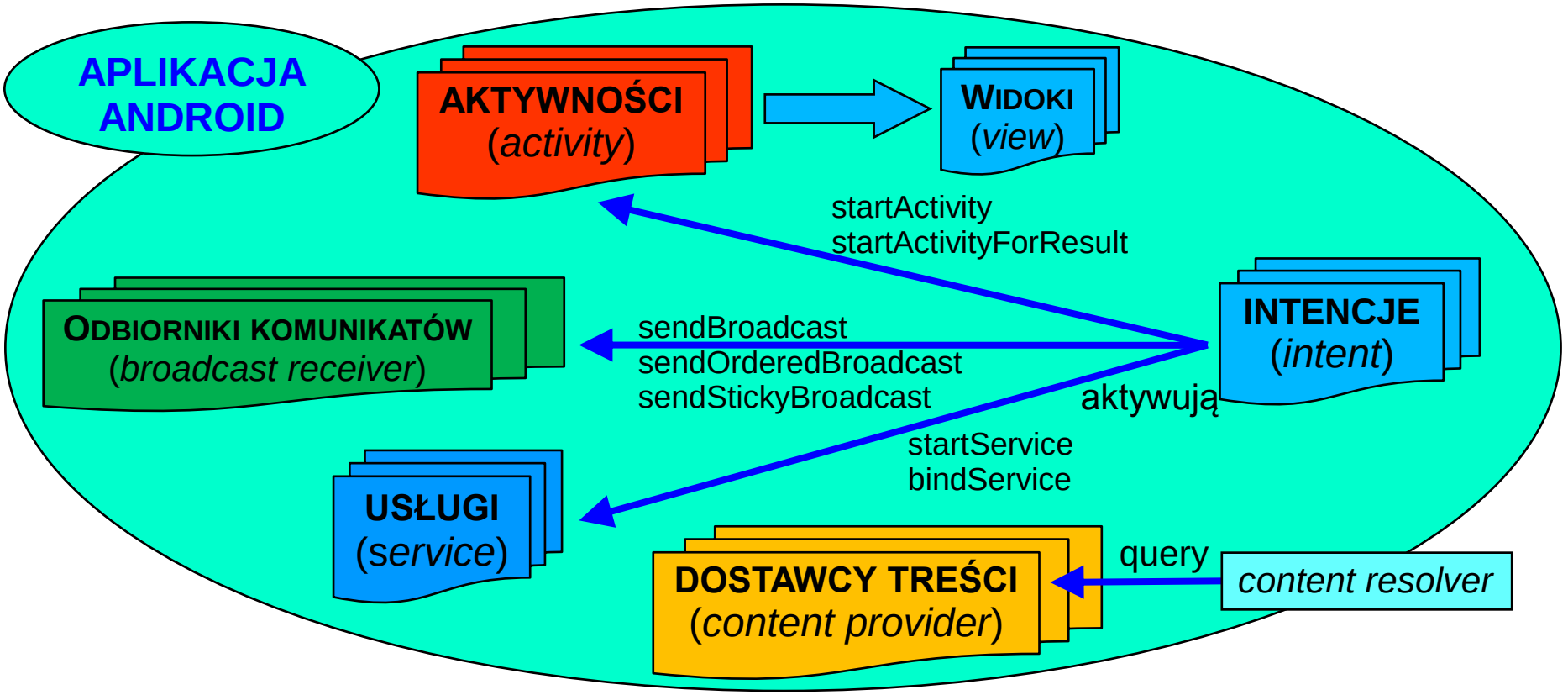
Budowa aplikacji w systemie Android

- Koncepcja aplikacji
- Struktura projektu aplikacji
- Cykl życia aplikacji
- Najprostsza aplikacja – kod źródłowy
- Najprostsza aplikacja – zasoby XML
- Emulator systemu Android
- Instalacja aplikacji w urządzeniu mobilnym

Budowa aplikacji

Koncepcja aplikacji

- Aplikacje są uruchamiane w oddzielnych procesach (instancjach Dalvik VM), czyli w środowisku chronionej pamięci (*sandboxing*).
- Każdy proces aplikacji ma przydzielony priorytet oraz identyfikator ID.
- Aplikacja składa się z wielu składników posiadających swój własny cykl życia.



KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu „Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Budowa aplikacji

Koncepcja aplikacji

Aktywność (*activity*):

- reprezentuje **ekran lub „okienko”** aplikacji (każdy ekran aplikacji jest oddzielną aktywnością)
- jednostka interfejsu użytkownika aplikacji przeznaczona do wykonywania przez użytkownika akcji, czynności
- może zawierać widoki (*view*), wyświetla interfejs użytkownika złożony z widoków i reaguje na zdarzenia
- uruchamiana przez intencję lub przez inną aktywność
- aktywności są grupowane i przechowywane na stosie
- w danym momencie użytkownik może być w interakcji z jedną aktywnością
- w aplikacji możliwe jest uruchamianie aktywności z innych aplikacji

Widok (*view*):

- element interfejsu użytkownika
- obiekt rysowany przybierający kształt etykiety, przycisku, pola edycyjnego itp. składników interfejsu

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Koncepcja aplikacji

Odbiornik komunikatów (*broadcast receiver*):

- umożliwia odbieranie komunikatów o zdarzeniach i odpowiadanie na nie (np. niski stan baterii, pobranie danych, połączenie przychodzące, wykonanie zdjęcia itp.)
- komunikaty mają formę intencji (*intent*) i są rozgłaszane przez system lub przez inne aplikacje
- nie posiada interfejsu – może tworzyć powiadomienia w pasku stanu

Intencja (*intent*):

- abstrakcyjny opis operacji, która ma być wykonana
- definiuje **zamiar** wykonania czynności takiej jak np. rozpoczynanie (uruchamianie) aktywności, uruchamianie usługi, nadawanie komunikatu itp.
- może być jawna lub niejawna
- wykorzystywana przez aplikacje lub przez system do powiadamiana różnych składników aplikacji o zdarzeniach
- dla aktywności i usług intencja definiuje **akcję** do wykonania (**`MAIN`**, **`VIEW`**, **`PICK`**, **`EDIT`**) i **dane** do przetworzenia (identyfikator URI danych), dla odbiorników komunikatów definiuje rozgłaszany komunikat

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Koncepcja aplikacji

Usługa (*service*):

- proces **działający w tle** przez dłuższy czas
- nie posiada interfejsu użytkownika
- może być lokalna (w bieżącej aplikacji) lub zdalna (do łączenia się z innymi aplikacjami)

Dostawca treści (*content provider*):

- abstrakcyjne źródło danych (dostawca lub adresat usług REST)
- zapewnia dostęp do składowanych danych (system plików, baza SQLite, sieć), umożliwia odczyt i zapis danych
- umożliwia współdzielenie danych między aplikacjami

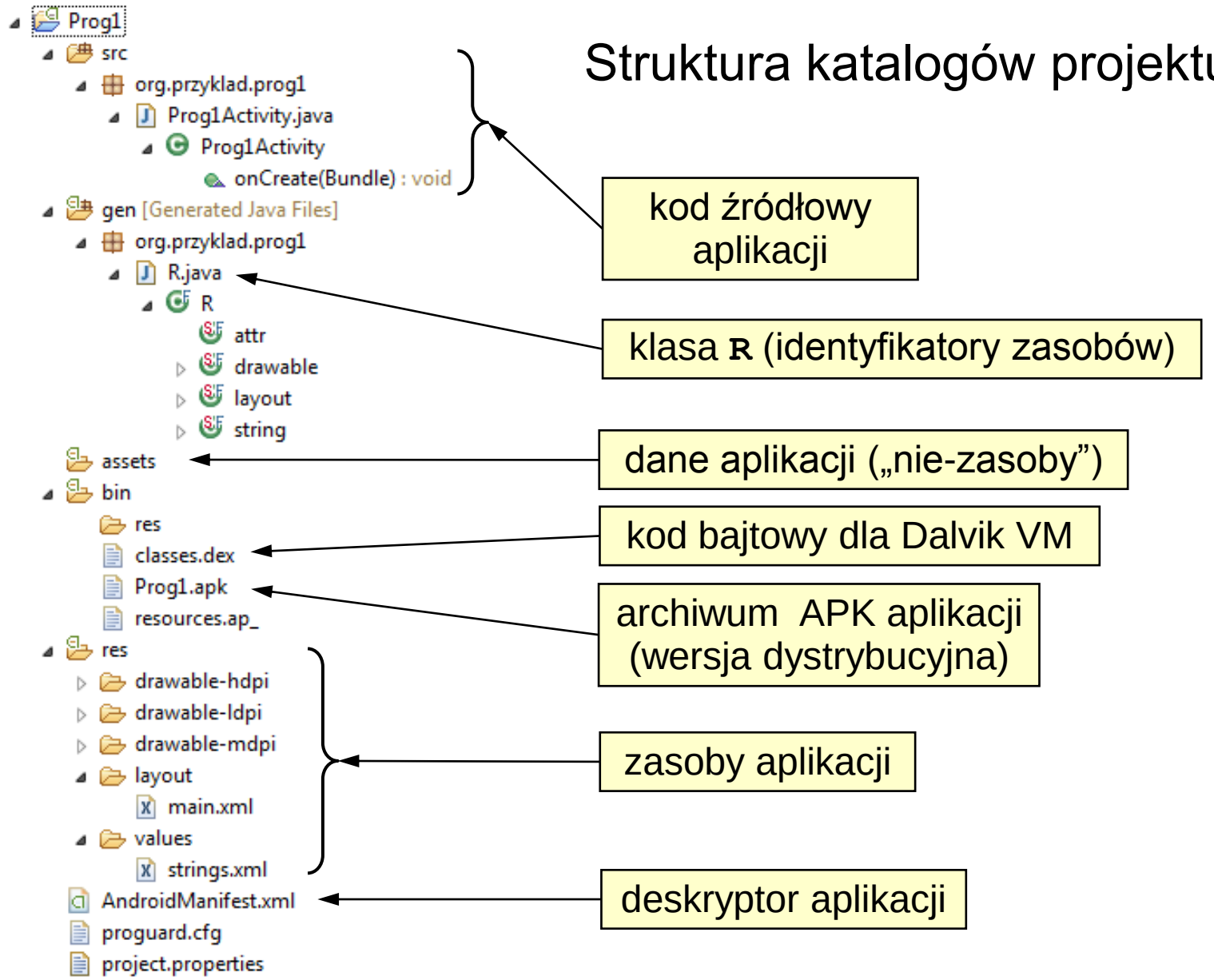
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Struktura katalogów projektu aplikacji



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Struktura katalogów projektu aplikacji – elementy składowe aplikacji

/assets	dowolne dane (nieskompresowane) dostępne przez AssetManager metodą **getAssets()**, nie traktowane jak zasoby, możliwe podkatalogi
/bin	pliki binarne ***.class**, ***.dex**, ***.apk**
/gen	pliki generowane (plik **R.java**) posegregowane na pakiety
/res	zasoby aplikacji
/res/anim	opisy animacji (np. plik **animation1.xml**)
/res/drawable-xxx	pliki obrazów, map bitowych dla aplikacji (np. plik ikony **icon.png**)
/res/layout	opis ekranów (widoków) i rozkładu UI (plik **main.xml**)
/res/menu	pliki opisujące listy menu
/res/raw	dodatkowe pliki danych, nieskompresowane (niekompilowane)
/res/values	opis napisów, kolorów, wymiarów, stylów (pliki **strings.xml**, **colors.xml**, **dimens.xml**, **styles.xml**, **values.xml**)
/res/xml	dodatkowe, własne pliki XML (kompilowane)
/src	pliki źródłowe ***.java** posegregowane na pakiety
AndroidManifest.xml	plik deskryptora aplikacji, definiuje wszystkie występujące w aplikacji aktywności, usługi, dostawców treści, adresatów intencji, odbiorniki komunikatów, uprawnienia dla aplikacji

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Cykl życia aplikacji

Cykl życia aplikacji związany jest z cyklem życia jego aktywności.

W zależności od działań podejmowanych przez użytkownika (ukrywanie, przywracanie, zatrzymywanie, zamykanie okien aktywności), aktywność zmienia stan i automatycznie uruchamiana jest przez system odpowiednia metoda cyklu życia aktywności (cyklem zarządza system Android).

metoda	przeznaczenie
`onCreate()`	pierwsze utworzenie aktywności: inicjalizacja, tworzenie interfejsu użytkownika
`onStart()`	uruchomienie aktywności
`onResume()`	na pierwszym planie, przejście ze stanu **`PAUSED`** do stanu działania **`RESUMED`**
`onPause()`	wstrzymanie aktywności (stan **`PAUSED`**), gdy inna aktywność przechodzi na pierwszy plan (bieżąca aktywność jest widoczna, ale częściowo zasłonięta). Ostatnia bezpieczna metoda wywoływana przed zamknięciem aplikacji.
`onRestart()`	ponowne uruchomienie aktywności (przywrócenie na pierwszy plan), przejście ze stanu **`STOPPED`** do stanu działania
`onStop()`	aktywność na drugim planie (stan **`STOPPED`**, więcej już niewidoczna)
`onDestroy()`	usuwanie i zakończenie działania aktywności

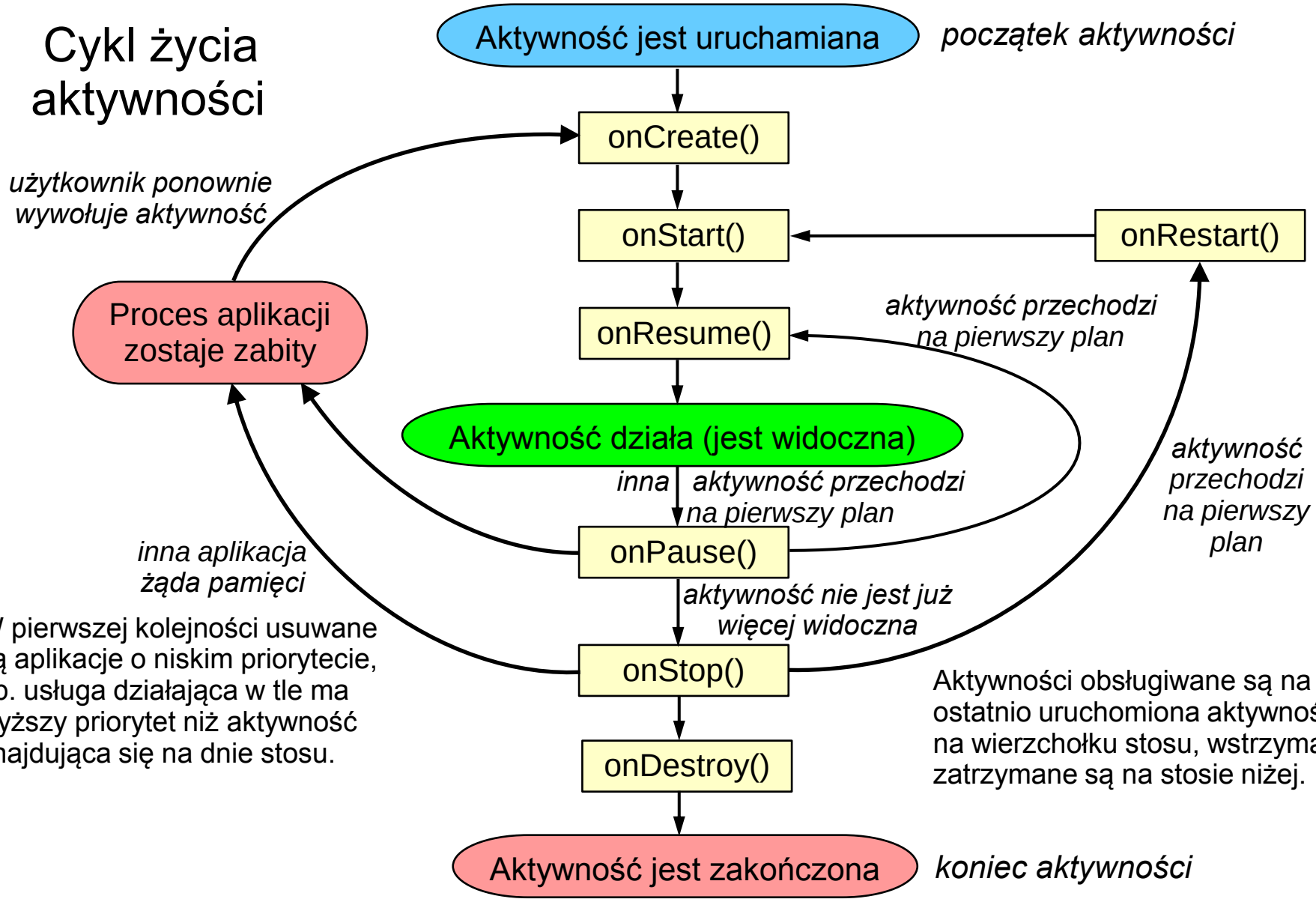
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Cykl życia aktywności



W pierwszej kolejności usuwane są aplikacje o niskim priorytecie, np. usługa działająca w tle ma wyższy priorytet niż aktywność znajdująca się na dnie stosu.

Aktywności obsługiwane są na stosie: ostatnio uruchomiona aktywność jest na wierzchołku stosu, wstrzymane i zatrzymane są na stosie niżej.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Najprostsza aplikacja

Tworzenie aplikacji (patrz ćwiczenie 2):

- utworzenie emulatora urządzenia mobilnego
- utworzenie projektu aplikacji
- uruchomienie aplikacji w emulatorze
- instalacja aplikacji na urządzeniu mobilnym

W środowisku Eclipse po utworzeniu projektu aplikacji, automatycznie generowany jest szablon aplikacji, składający się m.in. z następujących plików:

- plik **`Hello.java`** (kod źródłowy głównej klasy) — kod źródłowy aplikacji
- plik **`AndroidManifest.xml`** (deskryptor aplikacji) — deskryptor aplikacji
- plik **`R.java`** (kod klasy R, identyfikatory zasobów)
- pliki **`main.xml`** (główny plik zasobów, opis i definicja UI)
- plik **`strings.xml`** (dodatkowe zasoby: napisy)

} pliki zasobów

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Kod źródłowy głównej klasy aplikacji (plik **Hello.java**)

Najprostsza, przykładowa aplikacja wyświetla napis na ekranie urządzenia. Działanie aplikacji rozpoczyna się od tzw. aktywności początkowej.

```
package org.przyklad.prog1;

import android.app.Activity;
import android.os.Bundle;


public class Hello extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

aktywność początkowa (aktywność szczytowego poziomu)

metoda wywoływana automatycznie gdy aktywność rozpoczyna działanie

identyfikator **main** zasobów

wyświetlenie układu zdefiniowanego identyfikatorem **main** jako interfejsu w aktywności **Hello**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Plik **AndroidManifest.xml**

Deskryptor (manifest) aplikacji określa zawartość i zachowanie aplikacji. Deklaruje składniki aplikacji (aktywności, usługi i dostawców treści), uprawnienia konieczne do działania aplikacji i wymagania sprzętowe. Plik generowany jest automatycznie.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.przyklad.prog1"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="8" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".Hello" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

nazwa głównego pakietu jest kojarzona z nazwą aktywności i jej klasą: **com.example.prog1.Hello**

wersja API 8 => Android 2.2

ikona aplikacji

nazwa aplikacji

deklaracja aktywności

filtr intencji dla aktywności

Określenie aktywności początkowej: po uruchomieniu aplikacji odczytywany jest plik manifestu, wyszukiwane są i uruchamiane aktywności o akcji **MAIN** i kategorii **LAUNCHER**.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Plik `AndroidManifest.xml` – węzły

- **`<manifest>`** główny węzeł pliku: musi zawierać węzeł **`<application>`** i pola **`xlmns:android`** (przestrzeń nazw Android) i **`package`** (nazwa pakietu aplikacji)
- **`<supports-screens>`** określa wymagane rozmiary ekranu (*small*, *normal*, *large*, *extra large*) i rozdzielczość (*low*, *medium*, *high*, *extra high*)
- **`<uses-permission>`** określa uprawnienia, które muszą być udzielone aplikacji w czasie instalacji, konieczne do prawidłowego działania
- **`<uses-sdk>`** określa kompatybilność aplikacji z wersjami Android (minimalna wymagana wersja API)
- **`<uses-feature>`** określa wymagania sprzętowe (aparat, Bluetooth itp.)
- **`<application>`** deklaruje aplikację, opisuje jej właściwości, zawiera węzły:
  - **`<activity>`** deklaruje aktywność
  - **`<service>`** deklaruje usługę
  - **`<receiver>`** deklaruje odbiornik komunikatów
  - **`<provider>`** deklaruje dostawcę treści

  } mogą posiadać własne filtry intencji **`<intent-filter>`** określające, które intencje będą obsługiwane

  - **`<uses-library>`** deklaruje użycie dodatkowych bibliotek

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Plik **`AndroidManifest.xml`** – uprawnienia aplikacji

W zależności od specyfiki działania aplikacji, konieczne może być przydzielenie dla niej odpowiednich uprawnień w pliku deskryptora aplikacji.

Określanie uprawnień w środowisku Eclipse: w drzewie projektu wybór pliku **`AndroidManifest.xml`** – zakładka Permissions – przycisk Add... – wybór z listy polecenia Uses Permission i kliknięcie przycisku OK – z listy rozwijanej Name wybór uprawnienia **`android.permission.NAZWA_UPRAWNIENIA`**.

Wybrane, przykładowe uprawnienia:

- **`READ_CONTACTS, WRITE_CONTACTS`** – uprawnienia do odczytu/zapisu listy kontaktów
- **`READ_CALENDAR, WRITE_CALENDAR`** – uprawnienia do odczytu/zapisu danych użytkownika z kalendarza
- **`READ_SMS, WRITE_SMS`** – uprawnienia do odczytu/zapisu wiadomości SMS
- **`RECEIVE_SMS, SEND_SMS`** – uprawnienia do odbioru i wysyłania wiadomości SMS
- **`INTERNET`** – uprawnienie do tworzenia połączeń sieciowych na bazie gniazdek
- **`BLUETOOTH`** – uprawnienie do łączenia się z urządzeniami Bluetooth
- **`CAMERA`** – uprawnienie do obsługi aparatu fotograficznego
- **`BATTERY_STATS`** – uprawnienie do zbierania statystyk dotyczących baterii

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Główny plik zasobów **main.xml**

Węzeł **TextView** opisuje właściwości elementu interfejsu użytkownika (etykieta tekstowa).

Układ graficzny interfejsu zdefiniowany w pliku XML jest ładowany do okna aktywności **Hello**.

standardowy nagłówek rozpoczynający plik XML

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

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

przestrzeń nazw XML

szerokość i wysokość elementu na ekranie:
fill_parent – wypełnia ekran
wrap_content – dostosowana do zawartości elementu

napis umieszczony w etykiecie tekstowej **TextView**
– identyfikator napisu to **hello**,
– zawartość napisu jest zdefiniowana w pliku **strings.xml**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Główny plik zasobów **main.xml**

Dodany węzeł **LinearLayout** opisuje liniowy układ interfejsu użytkownika.

```
<?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="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>
```

układ wypełnia ekran

elementy UI rozkładane w poziomie

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Plik zasobów **strings.xml**

Dodatkowy plik zasobów przechowuje napisy w węzłach **<string>**.
Właściwość **name** to identyfikator napisu.

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

<resources>

    <string name="hello">Hello World, Prog1Activity!</string>
    <string name="app_name">Prog1</string>

</resources>
```

napis o identyfikatorze **hello**

napis o identyfikatorze **app_name**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Kod źródłowy klasy **R** (plik **R.java**)

```
package org.przyklad.prog1;

public final class R
{
    public static final class attr
    {   }

    public static final class drawable
    {
        public static final int ic_launcher = 0x7f020000;
    }

    public static final class layout
    {
        public static final int main = 0x7f030000;
    }

    public static final class string
    {
        public static final int app_name = 0x7f040001;
        public static final int hello = 0x7f040000;
    }
}
```

Generowana automatycznie klasa **R** zawiera identyfikatory zasobów aplikacji z katalogu **/res**.

Dla każdego rodzaju zasobów tworzona jest klasa wewnętrzna: **drawable**, **layout**, **string** itd.

Identyfikatory zasobów mają postać stałych typu **int**. Odpowiadają zasobom, elementom zdefiniowanym w plikach XML lub plikom.

identyfikator pliku **main.xml**

identyfikatory zasobów z pliku **strings.xml**

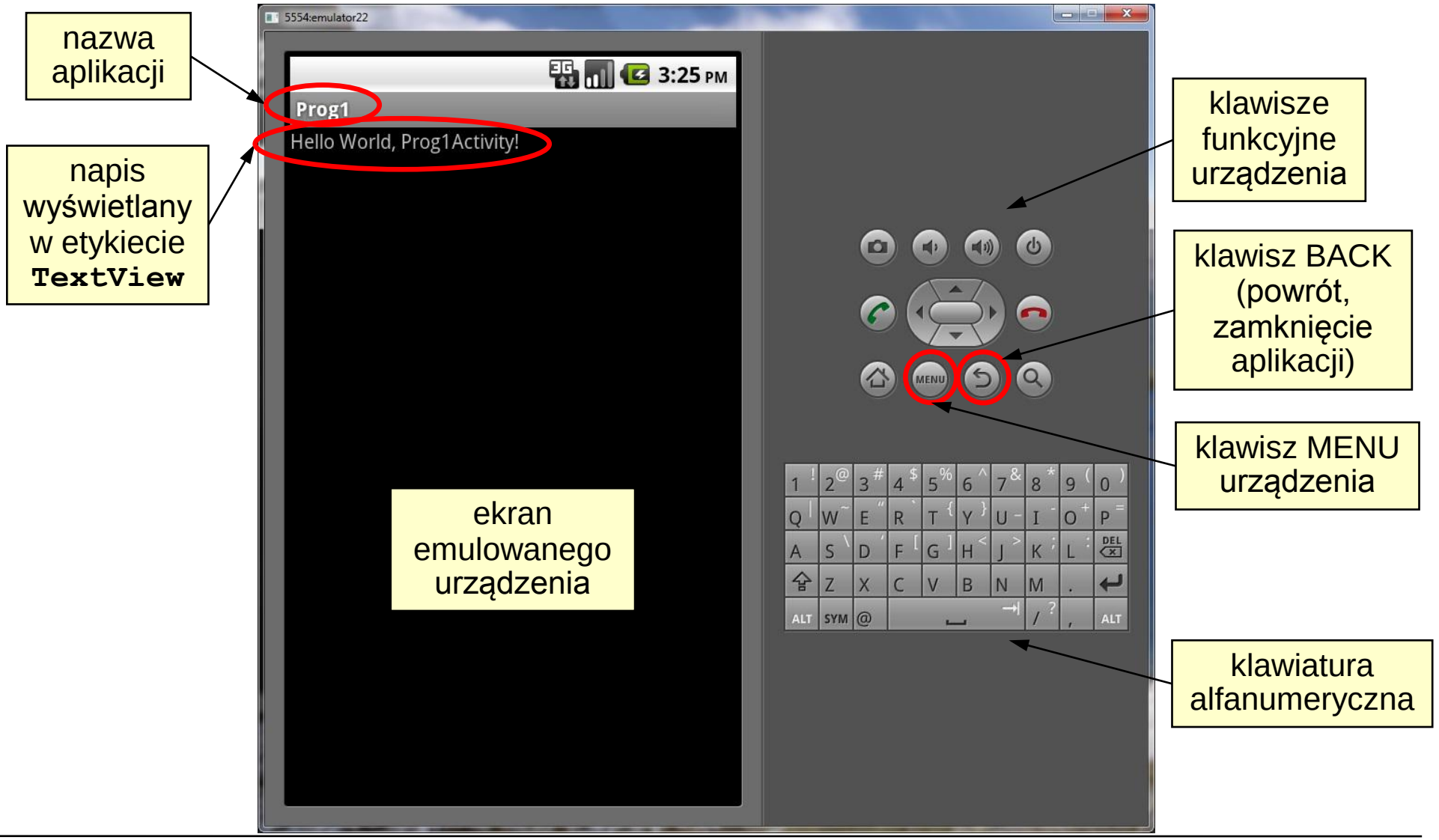
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Najprostsza aplikacja uruchomiona w emulatorze



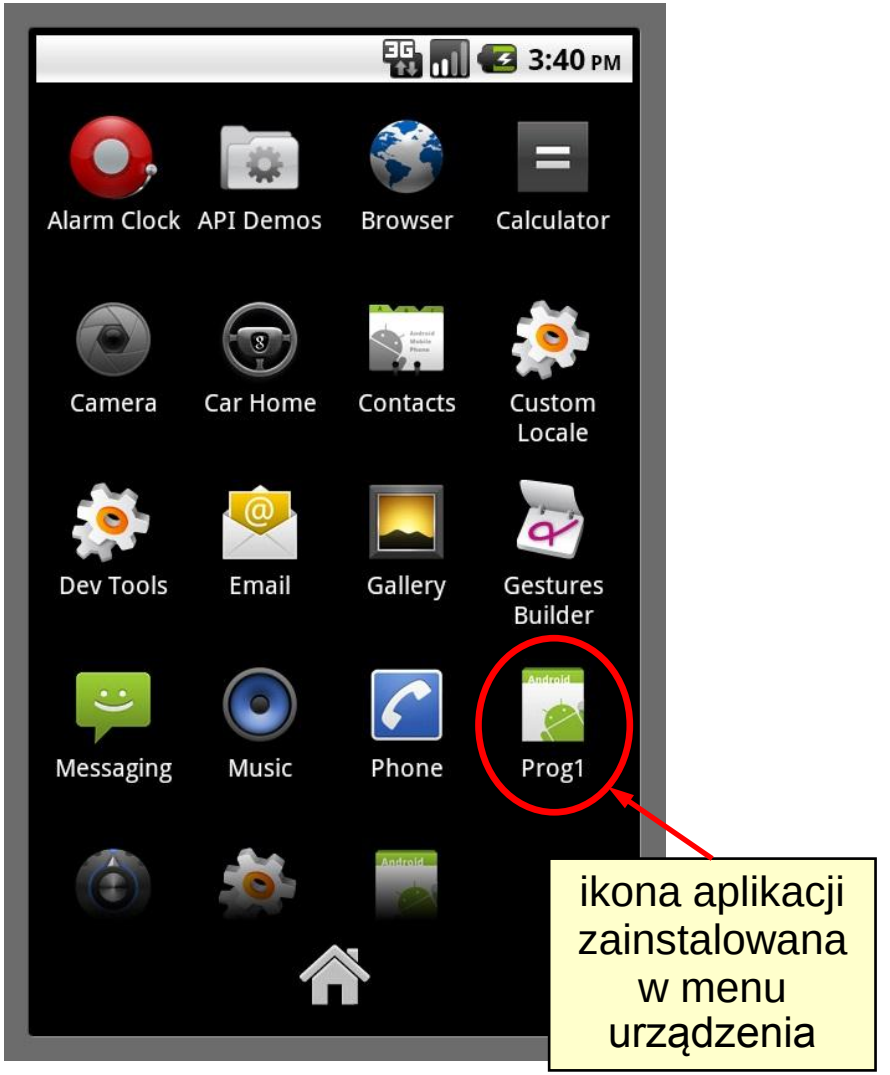
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Najprostsza aplikacja uruchomiona w emulatorze



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Budowa aplikacji

Emulator systemu Android

Emulator to wirtualne urządzenie AVD (*Android Virtual Device*) pozwalające na uruchomienie aplikacji bez konieczności instalowania jej w urządzeniu mobilnym.

W środowisku IDE możliwe jest tworzenie wielu emulatorów o różnych parametrach, dla różnych wersji sytemu Android (patrz ćwiczenie 1).

Charakterystyka emulatora:

- naśladuje większość funkcji urządzenia dzięki technologii emulacji procesora (QEMU) na poziomie jednostki centralnej
- symulowany jest 32-bitowy tryb ARM (*Advanced RISC Machine*) – na tak symulowanym procesorze uruchamiany jest Linux i platforma Android
- ograniczenia emulatora: połączenia USB i Bluetooth, wykonywanie zdjęć aparatem fotograficznym, nagrywanie wideo, obsługa słuchawek, symulacja baterii.

Budowa aplikacji

Instalacja aplikacji w urządzeniu mobilnym

W celu instalacji i uruchomienia aplikacji, do urządzenia należy przenieść plik **`*.apk`** (*Android Package*) dostępny w katalogu **`bin`** projektu. Plik ten zawiera kod wykonywalny aplikacji oraz jej zasoby.

W trakcie instalacji aplikacji nadawane są uprawnienia wymagane do jej prawidłowego działania (patrz strona 29).

Aplikację można zainstalować kilkoma metodami:

- metoda 1 (najwygodniejsza dla programistów): użycie środowiska Eclipse
- metoda 2 (najprostsza): użycie karty pamięci i menadżera plików dostępnego w urządzeniu
- metoda 3: instalacja przy użyciu Android SDK
- metoda 4: instalacja przy użyciu aplikacji AndroidCommander dla PC
- metoda 5: użycie przeglądarki internetowej w urządzeniu

Wszystkie metody instalacji opisane są w ćwiczeniu 5.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

- Koncepcja zasobów w systemie Android
- Interfejs użytkownika w kodzie Java
- Interfejs użytkownika w kodzie XML
- Odniesienia do zasobów
- Rodzaje zasobów

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Koncepcja zasobów w systemie Android

W systemie Android zasoby są deklaracyjne, tzn. definiowane przy użyciu plików XML.

Kod aplikacji (warstwa logiki) jest odseparowany od zasobów (warstwa prezentacji). Separacja umożliwia modyfikację zasobów bez konieczności wprowadzania zmian w kodzie źródłowym aplikacji lub ponownej kompilacji. Użycie alternatywnych zestawów zasobów ułatwia również obsługę ekranów o różnych rozdzielczościach i lokalizację aplikacji (tworzenie wersji językowych).

Dla każdego zasobu automatycznie generowany jest identyfikator.

Identyfikatory tworzone są przez kompilator zasobów AAPT (*Android Asset Packaging Tool*) i przechowywane w automatycznie generowanej klasie **`R`** (patrz strona 33).

Jakiekolwiek modyfikacje zasobów w plikach XML (katalog **`/res`** projektu) powodują automatycznie odpowiednie zmiany i aktualizacje w klasie **`R`** (plik **`R.java`**).

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Interfejs użytkownika w kodzie Java

Interfejs UI aplikacji można tworzyć metodą programową w kodzie Java.

plik **Hello.java**

Metoda niezalecana

```
package org.przyklad.prog1;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Hello extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Witaj świecie!");
        setContentView(tv);
    }
}
```

utworzenie widoku **TextView** i wyświetlenie napisu

metoda **setContentView** przypisuje treść widoku **tv** do widoku głównego

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Interfejs użytkownika w kodzie XML

Możliwe jest również tworzenie interfejsu UI metodą deklaracyjną, przy użyciu zasobów w plikach XML.

plik **/res/layout/main.xml**

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

<TextView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="Witaj świecie" />
```

napis wyświetlany w kontrolce **TextView**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Interfejs użytkownika

Metoda zalecana

Zalecane jest stosowanie metody hybrydowej: deklaracyjne tworzenie interfejsu w plikach XML i stosowanie odnośników w kodzie Java. Należy zatem utworzyć odnośniki do elementu **`TextView`** i do jego napisu, wykorzystując identyfikatory tych elementów.

plik **`/res/layout/main.xml`**

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/tv1"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@+string/powitanie" />
```

kontrolka **`TextView`** ma identyfikator **`tv1`** poprzez który odwołujemy sie do niej w kodzie

plik **`Hello.java`**

```
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    TextView tv = (TextView) findViewById(R.id.tv1);
}
```

wczytanie zasobów przed użyciem **`findViewById`**

napis ustalony w pliku **`strings.xml`** jako identyfikator **`powitanie`**

odwołanie sie do kontrolki zdefiniowanej w XML (metoda **`findViewById`**)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Składnia odnośnika do zasobu

Odnośnik do zasobu wiąże zasób zdefiniowany w pliku XML z jego identyfikatorem.

Ogólnie: **`@[pakiet:][+]typ/identyfikator`**

Przykład: **`@android:+string/napis`**

`@+string/napis`

`@string/napis`

`pakiet` nazwa pakietu (brak oznacza pakiet lokalny)

`+` opcjonalny, oznacza, że **`identyfikator`** zostanie utworzony jeżeli jeszcze nie istnieje

`typ` rodzaj zasobu, określany przez przestrzeń nazw **`R.string`**, **`R.layout`**, **`R.id`**, **`R.drawable`** lub **`R.attr`** (klasy wewn. **`R`**)

`identyfikator` identyfikator (nazwa) zasobu (będzie stałą typu **`int`** w pliku **`R.java`**)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Rodzaje zasobów

Rozróżniane są następujące rodzaje zasobów:

- zasoby *string* – napisy
- zasoby *layout* – układy graficzne
- zasoby *color* – kolory
- zasoby *dimension* – wymiary
- zasoby *image* – obrazy graficzne
- zasoby *color-drawable* – kolorowe obiekty do narysowania
- własne pliki XML (*xml*)
- własne pliki nieskompresowane (*raw*)

Pliki dodatkowe (*assests*) nie są traktowane jako zasoby.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Zasoby *string* – napisy

Napisy definiowane są w pliku **`/res/values/strings.xml`** (nazwa pliku dowolna). W głównym węźle **`<resources>`** umieszczane są elementy podrzędne **`<string>`** definiujące napisy.

Identyfikatory napisów są dostępne w przestrzeni nazw **`R.string.*`**

plik **`strings.xml`**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Prog1Activity!</string>
    <string name="app_name">Prog1</string>
</resources>
```

nazwy właściwości **`name`** stają się identyfikatorami zasobów w klasie **`R`**

plik **`R.java`**

```
public final class R
{
  .....
  public static final class string
  {
    public static final int app_name = 0x7f040001;
    public static final int hello = 0x7f020000;
  }
}
```

Klasa wewnętrzna **`R.string`** tworzy przestrzeń nazw dla identyfikatorów napisów. Identyfikatory mają postać stałych typu **`int`**.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Zasoby *string* – napisy

Napisy definiowane w zasobach mogą wykorzystywać mechanizmy formatowania ciągów znaków Java oraz wybrane znaczniki HTML.

plik **strings.xml**

identyfikatory napisów

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="napis1">Prosty ciąg znaków</string>
    <string name="napis2">”Cytat w cudzysłowie”</string>
    <string name="napis3">\”Inny przykład cytatu\”</string>
    <string name="napis4">My name is %2$s. %1$s %2$s</string>
    <string name="napis5">To jest <b><i>gruby ukośnik<i><b></string>
</resources>
```

kod Java

referencja do obiektu aktywności

```
String s1 = activity.getString(R.string.napis1);

String s4 = activity.getString(R.string.napis4);
String wynik = String.format(s4, ”James”, ”Bond”);

String s5 = activity.getString(R.string.napis5);
String wynik = android.text.Html.fromHtml(s5);
```

%1 %2

odnośnik do zasobu w kodzie Java

odnośnik do zasobu w kodzie XML

plik **main.xml**

```
<TextView android:text="@string/napis5"/>
```


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Zasoby *layout* – układy graficzne

Układy graficzne (wyglądy ekranów) definiowane są w pliku **`/res/layout/main.xml`**. Dla każdego ekranu aplikacji należy zdefiniować oddzielny plik układu graficznego (nazwy plików dowolne).

plik **`main.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="fill_parent"
    android:orientation="vertical" >
    <TextView
      android:id="@+id/tv1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/powitanie" />
    <Button
      android:id="@+id/przycisk1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@+string/powitanie" />
</LinearLayout>
```

UWAGA: od wersji API 8 (Android 2.2) zamiast stałej **`fill_parent`** można stosować stałą **`match_parent`**

identyfikatory elementów UI

układ **`LinearLayout`** posiada tutaj dwa elementy podrzędne: etykietę tekstową **`TextView`** i przycisk **`Button`**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Zasoby *layout* – układy graficzne

Dla identyfikatorów elementów interfejsu tworzona jest w klasie **R** wewnętrzna klasa **id**.

plik **R.java**

```
public final class R
{
   .....
   public static final class id
   {
      public static final int tv1 = 0x7f050000;
      public static final int przycisk1 = 0x7f050001;
   }

   public static final class layout
   {
      public static final int main = 0x7f030000;
   }
}
```

klasa wewnętrzna **R.layout** zawiera stałą **main** powiązaną z układem zdefiniowanym w pliku **main.xml**

odnośniki do zasobów w kodzie Java

kod Java

```
setContentView(R.layout.main);
TextView tv = (TextView) this.findViewById(R.id.tv1);
Button b1 = (Button) this.findViewById(R.id.przycisk1);
```


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Zasoby *color* – kolory

Kolory definiowane są w pliku **`/res/values/colors.xml`** (nazwa dowolna) w węzłach **`<color>`**. Identyfikatory kolorów są dostępne w przestrzeni nazw **`R.color.*`**. Podstawowy zestaw kolorów jest zdefiniowany domyślnie w przestrzeni nazw **`android.R.color`**.

plik **`colors.xml`**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="czerwony">#ff0000</color>
    <color name="zielony">#00ff00</color>
    <color name="niebieski">#0000ff</color>
    <color name="kolor_tla">#ffffff00</color>
</resources>
```

odnośnik do zasobu w kodzie Java

referencja do obiektu aktywności

```
int kolor = activity.getResources().getColor(R.color.zielony);
```

kod Java

plik **`main.xml`**

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/czerwony"
    android:text="@string/powitanie" />
```

odnośnik do zasobu w kodzie XML

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Zasoby *dimension* – wymiary

Wymiary stosowane są do rozmieszczania elementów interfejsu użytkownika i do tworzenia stylów. Definiowane są w węzłach **`<dimen>`** w pliku **`/res/values/dimens.xml`** (nazwa pliku dowolna).

Identyfikatory wymiarów są dostępne w przestrzeni nazw **`R.dimen.*`**

Rodzaje wymiarów:

- piksele (px) – fizyczne piksele ekranu
- cale (in)
- milimetry (mm)
- punkty (pt) – jeden punkt to 1/72 cala
- piksele niezależne od gęstości (dip, dp) – odniesieniem jest ekran o gęstości 160 dpi (pikseli na cal) – jest on mapowany na rzeczywisty ekran
- piksele niezależne od skali (sp) – stosowane dla czcionek, rzeczywisty rozmiar uwzględnia rozmiar czcionki i ustawienia użytkownika.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Zasoby *dimension* – wymiary

plik **dimens.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="rozmiar_w_pikselach">20px</dimen>
    <dimen name="rozmiar_w_dp">5dp</dimen>
    <dimen name="rozmiar_sredni">100sp</dimen>
</resources>
```

kod Java

odnośnik do zasobu w kodzie Java

```
float size;

size=activity.getResources().getDimension(R.dimen.rozmiar_w_dp);
```

plik **main.xml**

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="@dimen/rozmiar_w_pikselach"
    android:text="@string/powitanie" />
```

odnośnik do zasobu w kodzie XML

właściwości rozmiarów akceptują wymiary podawane w różnych jednostkach

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Zasoby *image* – obrazy

Obsługiwane są formaty obrazów GIF, JPG, PNG.

Każdy obraz umieszczony w katalogu **`/res/drawable`** otrzymuje identyfikator tworzony na podstawie nazwy pliku (np. plik o nazwie **`obrazek.jpg`** otrzyma identyfikator zasobu **`R.drawable.obrazek`**).

Identyfikatory obrazów są dostępne w przestrzeni nazw **`R.drawable.*`**

plik **`main.xml`**

```
<Button
      android:id="@+id/przycisk1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@+string/powitanie"
      android:background="@drawable/obrazek" />
```

odnośnik do zasobu w kodzie XML

kod Java

```
BitmapDrawable bd;
bd = activity.getResources().getDrawable(R.drawable.obrazek);
// tło przycisku pobrane z obiektu bitmapy:
przycisk.setBackgroudDrawable(bd);
// tło przycisku pobrane bezpośrednio z zasobu:
przycisk.setBackgroudResource(R.drawable.obrazek);
```

odnośnik do zasobu w kodzie Java

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Zasoby *color-drawable* – kolorowy obiekt

Kolorowy obiekt rysowany (*color-drawable*) to prostokąt wypełniony kolorem. Używany jest jako tło widoków lub zwykły element do narysowania. Może być zdefiniowany w dowolnym pliku XML w katalogu **`/res/values`** lub **`/res/drawable`**, przy użyciu węzła **`<drawable>`**. Identyfikatory obiektów są dostępne w przestrzeni nazw **`R.drawable.*`**

dowolny plik **`xml`**

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <drawable name="czerwony_prostokat">#ff0000</drawable>
    <drawable name="zielony_prostokat">#00ff00</drawable>
    <drawable name="niebieski_prostokat">#0000ff</drawable>
</resources>
```

kod Java

```
ColorDrawable cd;
cd = (Color Drawable)
activity.getResources().getDrawable(R.drawable.czerwony_prosotkat);
tv.setBackgroud(cd);  // tło widoku tekstu
```

plik **`main.xml`**

```xml
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textAlign="center"
    android:background="@drawable/czerwony_prostokat" />
```

odnośniki do zasobu

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Zasoby – własne pliki XML

Pliki umieszczone w katalogu **`/res/xml`** mogą zawierać dodatkowe zasoby.
Pliki te są kompilowane do postaci binarnej przez kompilator AAPT.
Identyfikatory zasobów są dostępne w przestrzeni nazw **`R.xml.*`**

plik **`test.xml`**

```
<rootelem1>
    <subelem1>
        Napis Witaj z podelementu XML
    </subelem1>
</rootelem1>
```

dostęp do pliku poprzez jego identyfikator zasobu

```
Resources res = activity.getResources();
XmlResourceParser xpp = res.getXml(R.xml.test);
xpp.next();
int ev = xpp.getEventType();
while (ev != XmlPullParser.END_DOCUMENT)
{
    switch (ev)  // typy zdarzeń z klasy XmlPullParser
    {
       // ... użycie dla obiektu xpp metod getName(), getText()
       // ... zapis do obiektu klasy StringBuffer metodą append
    }
    ev = xpp.next();
}
```

Zarys metody odczytu pliku XML

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Zasoby – własne pliki nieskompresowane (*raw*)

Pliki binarne lub tekstowe (nieskompresowane) umieszczone w katalogu **`/res/raw`** mogą zawierać dodatkowe zasoby (audio, video, tekstowe itp.). Każdy plik otrzymuje identyfikator zasobu. Pliki nie są kompilowane – są umieszczane w pakiecie APK w niezmienionej postaci.

Identyfikatory zasobów są dostępne w przestrzeni nazw **`R.raw.*`**

Metoda odczytu danych z tych plików zależy od ich zawartości.

```
// przykład odczytu pliku tekstowego
Resources res = activity.getResources();
InputStream is = res.openRawResource(R.raw.test);
String s = convertStreamToString(is);
is.close();
```

dostęp do pliku poprzez jego identyfikator zasobu

```
String convertStreamToString(InputStream is)
{
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    int i = is.read();
    while (i != -1)   {  baos.write(i);   i = is.read();   }
    return baos.toString();
}
```


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zasoby aplikacji

Pliki dodatkowe (*assets*)

Pliki nieskompresowane umieszczone w katalogu **`/assets`** mogą zawierać dodatkowe dane. Pliki te nie są traktowane jak zasoby, dlatego nie są dla nich generowane identyfikatory zasobów.

Możliwe jest tworzenie struktury podkatalogów w katalogu **`/assets`** (w przeciwieństwie do podkatalogów katalogu **`/res`**).

Dostęp do plików należy uzyskać podając ścieżkę relatywną względem katalogu **`/assets`**.

Odczyt plików realizowany jest przy pomocy klasy **`AssetManager`** i metody **`getAssets()`**.

```
// przykład odczytu pliku tekstowego
AssetManager am = activity.getAssets();
InputStream is = am.open(”test.txt”);
String s = convertStreamToString(is);  // patrz poprzedni slajd
is.close();
```

dostęp do pliku poprzez nazwę

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

- Koncepcja dostawcy treści
- Rejestrowanie dostawcy treści
- Identyfikatory URI
- Kursor treści
- Odczyt danych
- Zapis danych
- Pojęcie intencji
- Wykorzystanie intencji
- Intencje predefiniowane
- Kategorie i akcje intencji
- Adaptery

Dostawcy treści, intencje i adaptery

Koncepcja dostawcy treści

Dostawca treści to obiekt udostępniający źródło danych. Przykładem źródła danych umieszczonego w dostawcy treści jest baza danych SQLite (dostawca treści „opakowuje” bazę).

Dostawców treści używa sie do udostępniania danych na zewnątrz aplikacji lub do wymiany (współdzielenia) danych pomiędzy aplikacjami.

Przykłady dostawców treści (bazy danych SQLite `*.db`) w systemie Android:

- *Browser* (przeglądarka)
- *CallLog* (dziennik połączeń)
- *Contacts* (kontakty) – tabele *People*, *Phones*, *Photos*, *Groups*, …
- *Settings* (ustawienia)
- *MediaStore* (multimedia): – *Audio:* tabele *Albums*, *Artists*, *Genres*, *Playlists*
  – *Images*
  – *Video*
- *Calendar* (kalendarz) – od wersji 4.0

Dostawcy treści, intencje i adaptery

Przykład dostawcy treści

nowsze rozwiązanie: `ContactContracts`

Dostawca treści *Contacts*, tabela *Phones* (książka telefoniczna)

_ID	LABEL	NUMBER	NUMBER_KEY	TYPE
5	dom nad morzem	(601) 601 601	601 601 601	TYPE_HOME
11	biuro w Koszalinie	(602) 602-602	602 602 602	TYPE_WORK
14	służbowy	603.603.603	603 603 603	TYPE_MOBILE
21	domowy	(94) 123 456	94 123 456	TYPE_HOME

unikalne id w tabeli, służy także do łączenia danych z różnych tabel

Dostawca treści *Contacts*, tabela *People* (dane osobiste) – wybrane kolumny

_ID	NAME	NOTES	STARRED	TIMES_CONTACTED
5	Jan Kowalski	szwagier	true	10
11	Jan Nowak	kierownik biura	true	0
14	Jan Malinowski	agent ubezpieczeniowy	false	2
21	Jan Jankowski	znajomy z pracy	false	5

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Rejestrowanie dostawcy treści

Dostawca treści jest rejestrowany w aplikacji przy pomocy unikalnej nazwy (tzw. *upoważnienia*), podobnej do adresu strony www.

Rejestracja dostawcy treści odbywa się w pliku **`AndroidManifest.xml`** w węźle **`<provider>`**:

```
<provider android:name="NaszDostawca"
  android:authorities="com.politechnika.NaszDostawca" />
<provider android:name="NotePadProvider"
  android:authorities="com.google.provider.NotePad" />
```

upoważnienie

Dostęp do danych udostępnianych przez dostawcę treści uzyskuje się poprzez identyfikator URI (patrz strona 61).

Dostawca treści zwraca typ i podtyp MIME dla identyfikatora URI:

```
vnd.android.cursor.item/vnd.google.note
vnd.android.cursor.dir/vnd.google.note
```

dla pojedynczego rekordu

dla zbioru rekordów

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Identyfikatory URI

Po zarejestrowaniu dostawcy treści, dostępne są identyfikatory URI podobne do adresów URL:

```
content://com.politechnika.NaszDostawca/
content://com.google.provider.NotePad/
```

Identyfikatory URI pozwalają odwołać się do całej tabeli lub do wybranego rekordu:

```
content://com.google.provider.NotePad/Notes
content://com.google.provider.NotePad/Notes/5
```

identyfikator zbioru notatek

identyfikator notatki nr 5

Identyfikatory URI dla dostawców treści wbudowanych w system nie muszą posiadać pełnej struktury w nazwie:

```
content://media/internal/images
content://media/external/images
content://contacts/people
```

dostawca treści *MediaStore*

dostawca treści *Contacts*

URI typu **`String`**

```
MediaStore.Images.Media.INTERNAL_CONTENT_URI
MediaStore.Images.Media.EXTERNAL_CONTENT_URI
Contacts.People.CONTENT_URI
```

URI typu **`Uri`**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Kursor treści

Dostawca treści udostępnia dane do odczytu w postaci obiektu nazywanego kursorem treści. Jest to zbiór krotek (rekordów, wierszy) i kolumn.

Charakterystyka kursora treści:

- kursor jest wstępnie ustawiony przed pierwszą krotką – należy go ustawić na pierwszą krotkę metodą **`moveToFirst`**
- do odczytu danych konieczna jest znajomość nazw i typów kolumn w tabeli dostawcy treści (odczyt: **`getString`**, **`getInt`**, **`getFloat`**, …)
- metody przesuwania/ustawiania kursora wykorzystują indeksy kolumn – przed odczytem danych należy przekształcić nazwę kolumny w numer kolumny metodą **`getColumnIndex`** (metoda odwrotna: **`getColumnName`**)
- kursor można przesuwać do przodu, do tyłu lub ustawiać na dowolną krotkę (**`moveToNext`**, **`moveToPrevious`**, **`moveToFirst`**, **`moveToLast`**, **`moveToPosition`**, **`move`**)
- odczyt pozycji kursora: **`isFirst`**, **`isLast`**, **`isBeforeFirst`**, **`isAfterLast`**, **`isClosed`**
- odczyt liczby krotek: **`getCount`**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Odczyt danych (kwerenda)

Odczyt pojedynczego wiersza nr 5 (wszystkie kolumny):

odczyt rekordu nr 5

```
Uri ludzieUri = Contacts.People.CONTENT_URI;
Uri osobaUri = ludzieUri.withAppendedId(Contacts.People.CONTENT_URI, 5);
// lub: = ContentUris.withAppendedId(Contacts.People.CONTENT_URI, 5);
// lub: String osobaUri = "content://contacts/people/5";
// kwerenda dla rekordu (metoda 1, zalecana):
Cursor c = activity.managedQuery(osobaUri, null, null, null, null);
// kwerenda dla rekordu (metoda 2):
ContentResolver cr = activity.getContentResolver();
Cursor c = cr.query(osobaUri, null, null, null, null);
```

Odczyt wszystkich wierszy (tylko wybrane kolumny) :

możliwe łączenie tabel

```
String[] kol = new String[] { People._ID, People.NAME, People.NUMBER };
Uri ludzieUri = Contacts.People.CONTENT_URI;

Cursor c = activity.managedQuery(ludzieUri,
                                 kol,
                                 null,
                                 null,
                                 Contacts.People.NAME + " ASC");
```

identyfikator URI dostawcy treści

nazwy zwracanych kolumn (**null** to zwracanie wszystkich)

argumenty selekcji

klauzula **where** opisuje które zwracać wiersze (**null** to wszystkie)

kolejność sortowania wierszy (**ASC** to rosnąca)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Obsługa kursora treści

Do obsługi kursora treści najwygodniej jest wykorzystać pętlę **do-while**:

```
Cursor c = activity.managedQuery(People.CONTENT_URI, wynik, null, null,
           People.NAME + " ASC");

if (c.moveToFirst() == true) // kursor zawiera dane
{
   String nazwa, nrtel;
   int nazwakol = c.getColumnIndex(People.NAME);
   int nrtelkol = c.getColumnIndex(People.NUMBER);

   do
   {
      nazwa = c.getString(nazwakol);
      nrtel = c.getString(nrtelkol);
      // przetwarzanie odczytanych danych...
   } while (c.moveToNext());
}
else
   // kursor jest pusty, brak danych
```

utworzenie kursora treści

sprawdzenie, czy kursor jest niepusty

indeksy kolumn na podstawie ich nazw

możliwe łączenie tabel!

odczyt danych

przesuwanie kursora na kolejne rekordy

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Zapis danych

Kursor treści umożliwia jedynie odczyt danych. Zapis danych możliwy jest poprzez użycie klas **`ContentValues`** i **`ContentResolver`**.

Obiekt klasy **`ContentValues`** przechowuje pary klucz-wartość (nazwa kolumny-wartość kolumny). Obiekt taki jest wstawiany do bazy przy użyciu identyfikatora URI za pomocą obiektu **`ContentResolver`**.

```
ContentValues wartosc = new ContentValues();
wartosc.put(People.NAME, "Albert Einstein");
wartosc.put(People.STARRED, 1);  // do ulubionych
ContentResolver cr = activity.getContentResolver();

// dodanie nowego wiersza, uri to identyfikator nowego rekordu
Uri uri = cr.insert(People.CONTENT_URI, wartosc);

// aktualizacja rekordów, n to liczba zaktualizowanych rekordów
int n = cr.update(People.CONTENT_URI, wartosc, null, null);

// usuwanie rekordów, n - liczba usuniętych rekordów
n = cr.delete(People.CONTENT_URI, null, null);
Uri uri = Uri.withAppendedPath(People.CONTENT_URI, "5"); // rekord nr 5
n = cr.delete(uri, null, null);
```

przygotowany obiekt klasy **`ContentValues`**

klauzula **`where`**:
`null` – wszystkie krotki
`new String[]{5,11}`

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Pojęcie intencji

Intencja to zamiar wykonania pewnej czynności, akcja i powiązane z tą akcją dane.

Intencje są wykorzystywane do wywoływania aplikacji z poziomu innej aplikacji (wywoływanie aktywności, wywoływanie wewnętrznych lub zewnętrznych składników aplikacji). Innym zastosowaniem jest generowanie zdarzeń.

Podczas przydzielania intencji do aktywności (określania która aktywność będzie wywołana) brane są pod uwagę kolejno:

- składnik aktywności (obiekt `ComponentName` aktywności)
- nazwa akcji intencji
- dane dla intencji (identyfikator URI danych)
- kategoria intencji
- w przypadku braku decyzji system wyświetla listę aktywności i umożliwia wybór

Do uruchamiania aktywności służą metody:

- `startActivity` – uruchomienie aktywności w oddzielnym wątku, w postaci modalnego okna (wywołanie asynchroniczne, bez wywołań zwrotnych)
- `startActivityForResult` – j.w. ale z wywołaniem zwrotnym (onActivityResult)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Wykorzystanie intencji

Intencje definiowane są w tzw. filtrach intencji dla poszczególnych składników aplikacji (patrz plik deskryptora, strony 27-28).
Przykładowy filtr intencji dla aktywności:

```
<activity
    android:label="Testowanie aktywności"
    android:name="Hello" >

    <intent-filter >
        <action android:name="org.przyklad.intent.action.PokazWidok" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

pakiet

stały fragment

nazwa akcji

```
android.intent.action.MAIN
android.intent.category.LAUNCHER
```

Użycie intencji do wywołania aktywności:

```
public static void wywolajAplikacje(Activity parentActivity)
{
    String akcja = "org.przyklad.intent.action.PokazWidok";
    Intent intencja = new Intent(akcja);
    parentActivity.startActivity(intencja);
}
```

uruchomienie aktywności

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Wykorzystanie intencji

Istnieją również inne sposoby wykorzystania intencji do uruchamiania aktywności, nie wymagające stosowania filtrów intencji (ale wymagające rejestracji aktywności w pliku manifestu).

Przykład 1: uruchomienie aktywności poprzez określenie jej obiektu **`ComponentName`** (tzw. obiekt składnika)

```
// uruchomienie aktywności Contacts
Intent intencja = new Intent();
intencja.setComponent(new ComponentName(
    "com.android.contacts",
    "com.android.contacts.DialContactsEntryActivity");
startActivity(intencja);
```

obiekt **`ComponentName`** łączy nazwę pakietu z nazwą klasy

Przykład 2: uruchomienie aktywności za pomocą nazwy klasy aktywności

```
// tzw. intencja bezpośrednia:
Intent intencja = new Intent(activity, Hello.class);
activity.startActivity(intencja);
```

aktywność bieżąca, np. **`this`**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Intencje predefiniowane

W zależności od wersji platformy Android, dostępny jest zestaw predefiniowanych intencji umożliwiających uruchamianie aplikacji systemowych.

```
public static void wywolajAplikacje(Activity activity)
{
    Intent intencja = new Intent(Intent.AKCJA);
    intencja.setData(Uri.parse("dane"));
    activity.startActivity(intencja);
}
```

nazwa akcji, działania związanego z intencją

dane dla intencji

AKCJA	dane (identyfikator URI)	opis
ACTION_VIEW	"http://www.google.com"	uruchomienie przeglądarki www
ACTION_WEB_SEARCH	"http://www.google.com"	uruchomienie wyszukiwania www
ACTION_DIAL	brak danych i metody setData	klawiatura do wpisywania numeru
ACTION_CALL	"tel:601-601-601"	połączenie z numerem telefonu
ACTION_VIEW	"geo:lat,long?z=zoom"	wyświetlenie mapy dla współ. lat, long

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany"

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Kategorie intencji

Intencje podzielone są na kategorie, które pozwalają decydować o uruchamianej aktywności. Podczas uruchamiania aplikacji wyszukiwane są aktywności o kategorii **`CATEGORY_LAUNCHER`** (**`android.intent.category.LAUNCHER`**).

wybrane kategorie	opis aktywności
`CATEGORY_BROWSABLE`	aktywność możliwa do przeglądania (nie narusza reguł bezpieczeństwa przeglądarki)
`CATEGORY_DEFUALT`	aktywność domyślna
`CATEGORY_GADGET`	możliwa do osadzenia w aktywności nadrzędnej
`CATEGORY_HOME`	aktywność jest ekranem początkowym
`CATEGORY_LAUNCHER`	wyświetlana na ekranie początkowym systemu
`CATEGORY_PREFERENCE`	aktywność odpowiedzialna za ustawienia, wyświetlana na ekranie ustawień
`CATEGORY_SELECTED_ALTERNATIVE`	aktywność alternatywna, do wyboru z listy dostępnych aktywności (np. wybór edytora, przeglądarki itp.)
`CATEGORY_TAB`	możliwa do osadzenia w określonej aktyw. nadrzędnej
`CATEGORY_TEST`	aktywność testowa (dawniej **`CATEGORY_EMBED`**)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Akcje dla intencji

Intencje podzielone są również ze względu na akcje.

Podczas uruchamiania aplikacji wyszukiwane są aktywności o akcji **`ACTION_MAIN`** (**`android.intent.action.MAIN`**).

akcja	opis
`ACTION_CALL`	połączenie z numerem telefonu
`ACTION_DIAL`	wyświetlenie klawiatury do wpisywania numeru
`ACTION_EDIT`	
`ACTION_GET_CONTENT`	jak **`ACTION_PICK`**, ale zwracany jest element określonego typu MIME
`ACTION_MAIN`	aktywność początkowa (aktywność szczytowego poziomu)
`ACTION_PICK`	aktywność, która zwraca wynik (wyświetla listę elementów, umożliwia wybór jednego z nich i zwraca identyfikator URI wybranego elementu)
`ACTION_VIEW`	uruchomienie przeglądarki www, wyświetlenie mapy itp.
`ACTION_WEB_SEARCH`	uruchomienie wyszukiwania www

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Adaptery

Adaptery dziedziczą po klasie **`android.widget.Adapter`**.

Służą do powiązania kontrolek-pojemników dziedziczących po klasie **`AdapterView`** (**`ListView`**, **`GridView`**, **`Gallery`**, **`Spinner`**) z danymi umieszczanymi w tych kontrolkach. Odpowiadają również za utworzenie widoku dla każdego elementu danych (za dostarczenie widoków potomnych do pojemnika).

adapter	opis
`ArrayAdapter<T>`	najprostszy adapter, stosowany dla pojemników **`ListView`**
`CursorAdapter`	j.w., dostarcza dane dla listy poprzez tzw. kursor treści
`SimpleAdapter`	używany do zapełniania listy danymi statycznymi (także danymi pochodzącymi z zasobów)
`ResourceCursorAdapter`	dziedziczy po **`CursorAdapter`**, tworzy widoki z zasobów
`SimpleCursorAdapter`	dziedziczy po **`ResourceCursorAdapter`**, tworzy widoki **`TextView`** i **`ImageView`** zdefiniowane w zasobach z kolumn w kursorze treści
`BaseAdapter`	klasa abstrakcyjna, dziedzicząc po niej można tworzyć adaptery niestandardowe

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Adapter `ArrayAdapter`

Tworzenie adaptera (przykład 1):

```
String s[] = {"Kowalski", "Nowak"};
ArrayAdapter<String> adapter =
    new ArrayAdapter<String>(this,
                             android.R.layout.simple_list_item_1,
                             s);
```

dane kontrolki **`TextView`** są reprezentowane przez ciągi znaków

kontrolka **`TextView`** zdefiniowana przez Android SDK

Tworzenie adaptera (przykład 2):

```
Spinner sp = (Spinner) findViewById(R.id.spinner1);
adapter = ArrayAdapter.createFromResource(this, R.array.planety,
                                          android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
sp.setAdapter(adapter);
```

identyfikator zasobów

```
<string-array name="planety">
    <item>Merkury</item>
    <item>Wenus</item>
    <item>Ziemia</item>
    <item>Mars</item>
    <item>Jowisz</item>
    <item>Saturn</item>
    <item>Uran</item>
    <item>Neptun</item>
</string-array>
```

dane kontrolki **`Spinner`** pochodzą z zasobów **`string-array`**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Adapter `SimpleCursorAdapter`

Adapter **`SimpleCursorAdapter`** przekształca krotkę w kursorze na widok podrzędny w kontrolce-pojemniku (np. na listę **`ListView`** zawierającą elementy **`TextView`**).

Konstruktor adaptera:

```
SimpleCursorAdapter(Context context, int layout,
                    Cursor c, String[] from, int[] to)
```

`context`	kontekst aktywności
`layout`	identyfikator zasobu widoku potomnego (XML)
`c`	kursor treści
`from`	tablica zawierająca nazwy kolumn, które mają zostać wczytane (każda kolumna jest mapowana do kontrolki **`TextView`**)
`to`	tablica zawierająca identyfikatory zasobów kontrolek **`TextView`**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

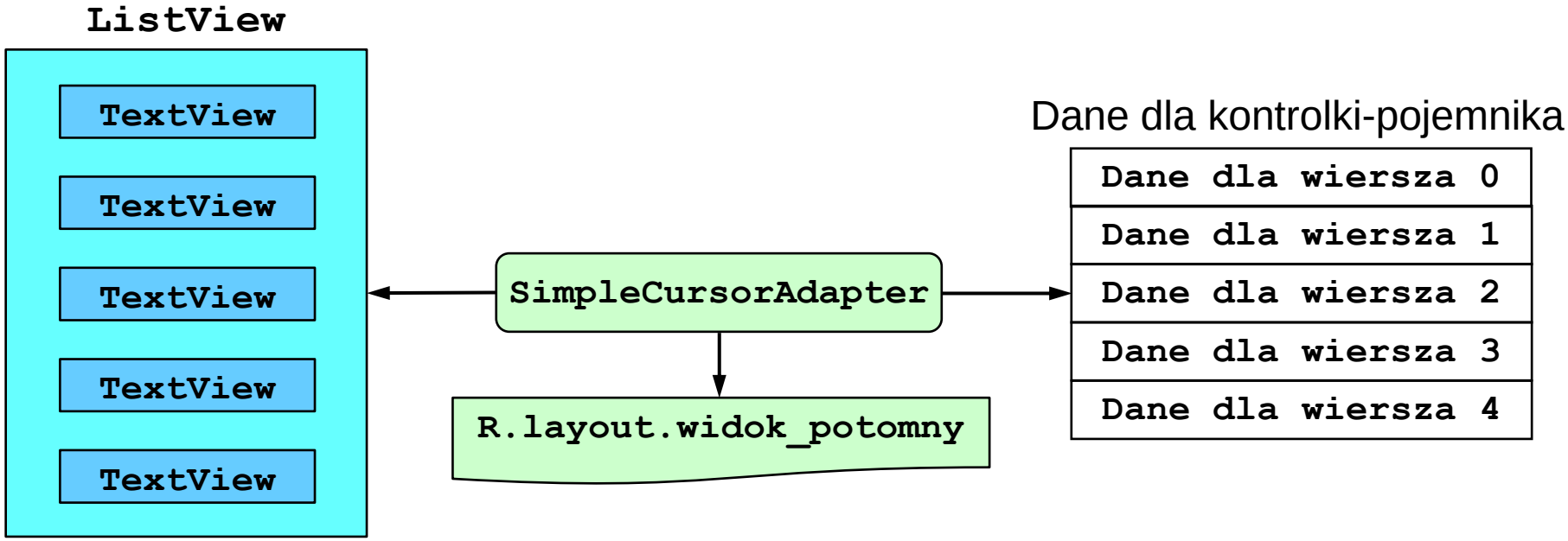
Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Dostawcy treści, intencje i adaptery

Adapter `SimpleCursorAdapter`

Adapter **`SimpleCursorAdapter`** przekształcający dane na listę **`ListView`** zawierającą elementy **`TextView`**.



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

- Koncepcja interfejsu użytkownika
- Hierarchia klas kontrolek
- Kontrolki tekstu
- Kontrolki przycisków
- Kontrolki pól wyboru
- Kontrolka listy
- Kontrolka siatki
- Kontrolki daty i czasu
- Dodatkowe kontrolki
  - Toast
  - MapView
  - Gallery
  - Spinner

Interfejs użytkownika – elementy

Koncepcja interfejsu użytkownika

Podstawowe składniki budulcowe interfejsu użytkownika:

- widok (*view*) = widget = kontrolka
  - podstawowy element interfejsu użytkownika
  - niewielki zestaw predefiniowanych kontrolek
- pojemnik = kontener
  - widok służący do przechowywania innych widoków
- układ graficzny (*layout*)
  - plik XML opisujący widok
  - układ graficzny zarządza rozmieszczeniem kontrolek wewnątrz pojemnika

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Koncepcja interfejsu użytkownika

Interfejs użytkownika (UI) może być tworzony dwoma metodami:

- metoda programowa (niezalecana)
  - UI definiowany bezpośrednio w kodzie źródłowym Java
  - małe zmiany w UI mogą pociągać za sobą konieczność dużych zmian w kodzie źródłowym
  - trudności w wiązaniu i kojarzeniu ze sobą elementów UI
- metoda deklaracyjna (deklaratywna) (zalecana)
  - UI definiowany w oparciu o pliki XML zapisywane w urządzeniu w postaci binarnej (kompilowane narzędziem AAPT – *Android Asset Packaging Tool*)
  - struktura pliku XML ma postać drzewa elementów, w którym każdy węzeł opisuje właściwość elementu UI
  - metoda inspirowana modelem tworzenia aplikacji MVC
  - separacja warstwy prezentacji (widoku) od warstwy logiki (kodu): można zmieniać wygląd i zachowanie aplikacji poprzez edycję plików XML

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

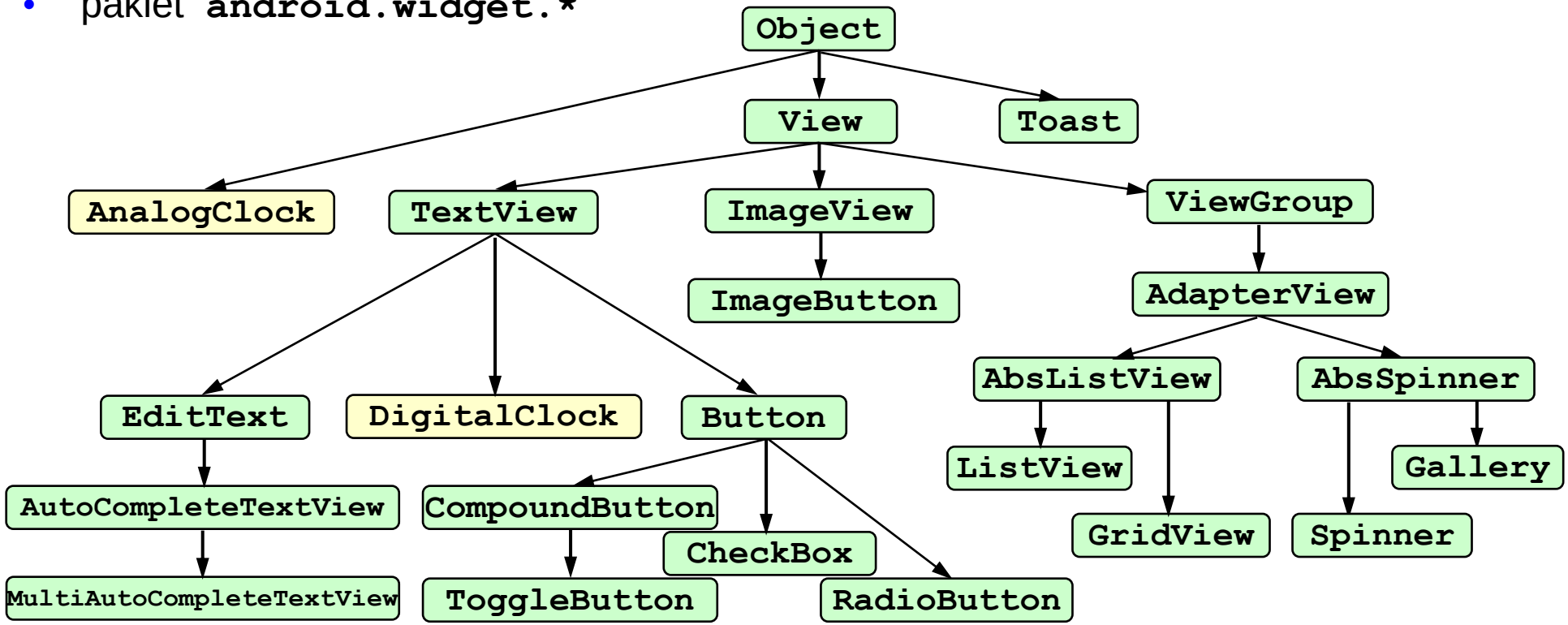
UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Hierarchia klas kontrolek

Podstawę kontrolek stanowią:

- klasa **`android.view.View`** (widok ogólnego przeznaczenia)
- klasa **`android.widget.ViewGroup`** (klasa bazowa układów graficznych)
- pakiet **`android.widget.*`**



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolki tekstu

kontrolka	opis
`TextView`	etykieta tekstowa (nieedytowalna); w zależności od rodzaju tekstu (adres URL, e-mail, nr telefonu) zostaje on podświetlony, a jego kliknięcie wywołuje odpowiednią akcję

`EditText`
`CharSequence getText()`
`setText(CharSequence)`
	pole edycyjne; wybrane właściwości:
	`inputType` – rodzaj wprowadzanego tekstu (**`number`**, **`numberSigned`**, **`numberDecimal`**)
	`autoText` – poprawianie błędów w trakcie pisania
	`capitalize` – autom. duże litery na początku zdań i wyrazów
	`phoneNumber` – akceptowanie numeru telefonu
	`password` – akceptowanie hasła
	`singleLine` – tekst wprowadzany w jednym wierszu
	style treści HTML: znaczniki **`<b> <i> <u>`** w zasobach kontrolki
	style treści programowe: j.w. + przekreślenie, indeksy, tło

`AutoCompleteTextView`	pole edycyjne z podpowiedziami (uzupełnianie całego tekstu); do kontrolki przypisana jest lista sugestii (metoda **`setAdapter`**) i sposób ich wyświetlania
`MultiAutoCompleteTextView`	pole edycyjne z podpowiedziami (uzupełnianie poszczególnych wyrazów wprowadzanego tekstu); dodatkowo określa się miejsce wyświetlania sugestii (metoda **`setTokenizer`**)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolki tekstu

```
// programowe tworzenie kontrolki TextView
TextView tv = new TextView(this);
tv.setText("Witaj świecie!");

// pobranie opisu kontrolki z zasobów
TextView tv = (TextView) this.findViewById(R.id.tv1);

// użycie klasy Linkify - tworzenie łącza do treści kontrolki
// kliknięcie łącza wywołuje odpowiednią intencję
TextView tv = (TextView) this.findViewById(R.id.tv1);
tv.setText("Odwiedź stronę http://onet.pl");  // uruchom. przeglądarki
tv.setText("Zadzwoń pod 112");                // wybieranie numeru
Linkify.addLinks(tv, Linkify.ALL);
```

```
// pobranie opisu kontrolki EditText z zasobów i utworzenie stylu
EditText et = (EditText) this.findViewById(R.id.et1);
et.setText("Michael \"Air\" Jordan");
Spannable span = et.getText();
span.setSpan(new BackgroundColorSpan(Color.RED, 10, 12,
             Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
span.setSpan(new StyleSpan(android.graphics.Typeface.BOLD_ITALIC),
             10, 12, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
```

czerwone tło

dla znaków 10÷12

pogrubienie i kursywa

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolki tekstu

```
// tworzenie podpowiedzi dla kontrolki AutoCompleteTextView
AutoCompleteTextView actv;
actv = (AutoCompleteTextView) this.findViewById(R.id.actv1);

ArrayAdapter<String> aa = new ArrayAdapter<String>(this,
   android.R.layout.simple_dropdown_item_1line,
   new String[] { "Audi", "Mercedes", "Opel", "Volkswagen" });

actv.setAdapter(aa);
```

lista podpowiedzi umieszczona w adapterze

powiązanie kontrolki **actv** z podpowiedziami

Tworzenie podpowiedzi dla kontrolki **MultiAutoCompleteTextView** jest analogiczne. Należy dodatkowo określić miejsce wyświetlania sugestii:

```
// tworzenie podpowiedzi dla kontrolki MultiAutoCompleteTextView
mactv.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
```

podpowiedzi wyświetlane po wpisaniu przecinka

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolki przycisków

kontrolka	opis
`Button`	przycisk podstawowy, zawiera napis
`ImageButton`	przycisk obrazkowy, zawiera obraz
`CompoundButton`	przycisk dwustanowy, przełącznik
`ToggleButton`	przycisk dwustanowy, przełącznik z dodatkową sygnalizacją: włączony – przycisk zawiera zielony pasek i napis "On” wyłączony – przycisk zawiera szary pasek i napis "Off”

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka przycisku **Button** – obsługa kliknięcia (metoda 1)

```
<Button
    android:id="@+id/przycisk1"
    android:text="@+string/napis_przycisku"
    android:typeface="serif"
    android:textstyle="bold"
    android:layout_width="fill_parent"
    android:layotu_height="wrap_content" />
```

```
public class Test extends Activity implements OnClickListener
{
    protected void onCreate(Bundle savedValues)
    {
        ...
        Button btn = (Button) this.findViewById(R.id.przycisk1);
        btn.setOnClickListener(this);
    }

    public void onClick(View v)
    {
        // wykonaj czynności gdy naciśnięto przycisk
        this.finish(); // zakończenie aktywności
    }
}
```

interfejs nasłuchu

metoda **setOnClickListener** ustala obiekt nasłuchiwacza: tutaj **this**

metoda **onClick** nasłuchiwacza

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka przycisku **Button** – obsługa kliknięcia (metoda 2)

```
<Button
    android:id="@+id/przycisk1"
    android:text="@+string/napis_przycisku"
    android:typeface="serif"
    android:textstyle="bold"
    android:layout_width="fill_parent"
    android:layotu_height="wrap_content" />
```

anonimowy obiekt nasłuchiwacza

```
Button btn = (Button) this.findViewById(R.id.przycisk1);
btn.setOnClickListener(new OnClickListener()
{
    public void onClick(View v)
    {
        // wykonaj czynności gdy naciśnięto przycisk
        // switch (v.getId()) .....
        this.finish();
    }
});
```

metoda **setOnClickListener** ustala obiekt nasłuchiwacza: tutaj tworzony dynamicznie

metoda **onClick** nasłuchiwacza

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka przycisku **Button** – obsługa kliknięcia (metoda 3)

```
<Button
    android:id="@+id/przycisk1"
    android:text="@+string/napis_przycisku"
    android:typeface="serif"
    android:textstyle="bold"
    android:layout_width="fill_parent"
    android:layotu_height="wrap_content"
    android:onClick="naszClickHandler" />
```

nazwa metody obsługi kliknięcia (od Android 1.6)

```
Button btn = (Button) this.findViewById(R.id.przycisk1);

public void naszClickHandler(View v)
{
    switch(v.getId())
    {
        case R.id.przycisk1: ..... ; break;
        case .....
    }
}
```

metoda **naszClickHandler** może również obsługiwać inne kontrolki

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka przycisku **ImageButton** – wstawianie obrazka

Umieszczanie obrazka na przycisku w zasobach (właściwość **src**):

```
<ImageButton
    android:id="@+id/przycisk2"
    android:layout_width="wrap_content"
    android:layotu_height="wrap_content"
    android:src="@drawable/obrazek" />
```

identyfikator zasobu obrazka

Umieszczanie obrazka na przycisku dynamicznie (metodą **setImageResource**):

```
ImageButton btn;
btn = (ImageButton) this.findViewById(R.id.przycisk2);
btn.setImageResource(R.drawable.obrazek);
```

identyfikator zasobu obrazka

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka przycisku **`ToggleButton`** – zmiana stanu przycisku

Tworzenie przycisku:

```
<ToggleButton
    android:id="@+id/przycisk3"
    android:layout_width="wrap_content"
    android:layotu_height="wrap_content"
    android:text="Przycisk przełączania"
    android:textOff="Wyłączone"
    android:textOn="Włączone" />
```

napis niewykorzystywany!

napisy na przycisku

Zmiana stanu przycisku po jego kliknięciu sygnalizowana jest na przycisku odpowiednim napisem i zielonym lub szarym paskiem.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka pól wyboru **CheckBox**

Kontrolka **CheckBox** to dwustanowe pole wyboru.

Tworzenie pól wyboru:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <CheckBox android:text="Śniadanie"
        android:layout_width="wrap_content"
        android:layotu_height="wrap_content" />
    <CheckBox android:text="Obiad"
        android:layout_width="wrap_content"
        android:layotu_height="wrap_content" />
    <CheckBox android:text="Kolacja"
        android:layout_width="wrap_content"
        android:layotu_height="wrap_content" />
</LinearLayout>
```

pola wyboru będą układane liniowo w pionie

opis pól wyboru

Obsługa w metodzie **onClick** jak przycisk **Button** (strona 84)

Stanem pola wyboru zarządzają metody **setChecked()** i **toggle()**.

Metoda **isChecked()** zwraca stan pola.

Zdarzenie związane z zaznaczeniem lub odznaczeniem pola rejestruje się metodą **setOnCheckedChangeListener()** oraz implementując interfejs **OnCheckedChangeListener** i metodę **onCheckedChange()**.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka pól wyboru **RadioButton**

Kontrolka **RadioButton** to pole jednokrotnego wyboru (klasa **android.widget.RadioButton**).

Pola jednokrotnego wyboru są grupowane przy pomocy elementu **RadioGroup** (klasa **android.widget.RadioGroup**) . Element ten może również grupować inne kontrolki.

Uwaga: zaznaczenia nie można usunąć przez ponowne kliknięcie pola wyboru – można jedynie usunąć zaznaczenie wszystkich pól wyboru metodą **clearCheck()**.

Domyślnie, żadne pole wyboru nie jest zaznaczone. Wybraną opcję można zaznaczyć programowo metodą **setChecked** lub **toggle**. Metoda **isChecked()** zwraca stan pola.

```
RadioButton rbtn;
rbtn = (RadioButton) this.findViewById(R.id.pole1);
rbtn.setChecked(true);
```

zaznaczenie pola o identyfikatorze zasobu **pole1**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka pól wyboru **RadioButton**

Obsługa w metodzie **onClick** jak przycisk **Button** (strona 84)

Tworzenie pól jednokrotnego wyboru:

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

   <RadioGroup android:id="@+id/grupa_wzrost"
      android:layout_width="wrap_content"
      android:layotu_height="wrap_content"
      android:orientation="vertical" >

      <RadioButton android:id="@+id/pole1" android:text="Niski"
         android:layout_width="wrap_content"
         android:layotu_height="wrap_content" />

      <RadioButton android:id="@+id/pole2" android:text="Średni"
         android:layout_width="wrap_content"
         android:layotu_height="wrap_content" />

      <RadioButton android:id="@+id/pole3" android:text="Wysoki"
         android:layout_width="wrap_content"
         android:layotu_height="wrap_content" />
   </RadioGroup>
</LinearLayout>
```

pola jednokrotnego wyboru będą układane liniowo w pionie

opis pól wyboru

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka listy **ListView**

Kontrolka **ListView** to pionowa lista elementów.

Tworzenie listy polega na utworzeniu pliku **/res/layout/list_item.xml** o przykładowej zawartości:

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

    <CheckBox xmlns:android="http://schemas.android.com/apk/res/android"
       android:id="@+id/row_chbox"
       android:layout_width="wrap_content"
       android:layotu_height="wrap_content" />

    <TextView android:id="@+id/row_tv"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content" />

</LinearLayout>
```

lista zawiera elementy **CheckBox** i **TextView**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka listy `ListView`

Wypełnianie listy kontaktami:

klasa aktywności dziedzicząca po **`ListActivity`** (jej obiekt zawiera kontrolkę **`ListView`**)

```
public class ListViewActivity extends ListActivity
{
   private SimpleCursorAdapter adapter;

   @Override
   protected void onCreate(Bundle savedInstaceState)
   {
      super.onCreate(savedInstanceState);
      Cursor c = getContentResolver().query(People.CONTENT_URI,
                 null, null, null, null);
      startManagingCursor(c);
      String[] cols = new String[] { People.NAME };
      int[] names = new int[] { R.id.row_tv };
      adapter = new SimpleCursorAdapter(this, R.layout.list_item,
                c, cols, names);
      this.setListAdapter();
   }
}
```

lista kontaktów pobrana z urządzenia

wybór kolumny **`NAME`**

mapowanie nazw kontaktów na elementy **`TextView`**

tworzenie adaptera dla kursora treści **`c`**

metoda **`setListAdapter`** umieszcza dane na liście (każdy wiersz listy ma zdefiniowany układ graficzny XML)

Aplikacja wymaga nadania w deskryptorze uprawnienia READ_CONTACTS (dostęp do listy kontaktów urządzenia).

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka siatki **GridView**

Kontrolka **GridView** umożliwia wyświetlanie danych (tekst, rysunki itp.) w siatce. Siatkę definiuje się w pliku XML układu graficznego.

Tworzenie siatki polega na utworzeniu pliku **/res/layout/gridview.xml** o przykładowej zawartości:

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/siatka"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10px"
    android:vertical_spacing="10px"
    android:horizontal_spacing="10px"
    android:numColumns="auto_fit"
    android:columnWidth="100px"
    android:stretchMode="columnWidth"
    android:gravity="center" />
```


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka siatki **GridView**

Wypełnianie siatki nazwami kontaktów (analogicznie do strony 93):

```
protected void onCreate(Bundle savedInstaceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.gridview);
    GridView gv = (GridView) this.findViewById(R.id.siatka);

    Cursor c = getContentResolver().query(People.CONTENT_URI,
               null, null, null, null);
    startManagingCursor(c);
    String[] cols = new String[] { People.NAME };
    int[] names = new int[] { android.R.id.text1 };

    SimpleCursorAdapter adapter;
    adapter = new SimpleCursorAdapter(this,
              android.R.layout.simple_list_item_1, c, cols, names);
    gv.setListAdapter(adapter);
}
```

wczytanie siatki do widoku aktywności

powiązanie danych z siatką za pomocą klasy **android.widget.ListAdapter**

Aplikacja wymaga nadania w deskryptorze uprawnienia READ_CONTACTS (dostęp do listy kontaktów urządzenia).

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu „Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolki daty i czasu

kontrolka	opis
DatePicker	ustawienie lub wybór daty
TimePicker	ustawienie lub wybór czasu
DatePickerDialog	odpowiednik kontrolki daty w postaci okna dialogowego
TimePickerDialog	odpowiednik kontrolki czasu w postaci okna dialogowego
AnalogClock	zegar analogowy (wskazówka godzin i minut) – tylko wyświetlanie
DigitalClock	zegar cyfrowy (godziny, minuty, sekundy) – tylko wyświetlanie

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TimePicker android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

kontrolka daty

kontrolka czasu

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolki daty i czasu

Brak inicjalizacji kontrolek w kodzie powoduje ustawienie w nich bieżącej daty i bieżącego czasu.

Programowa inicjalizacja kontrolek na wymagane wartości:

```
protected void onCreate(Bundle savedInstaceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.dateTime);

    DatePicker dp = (DatePicker) this.findViewById(R.id.datePicker);
    // data 2011-12-01
    dp.init(2011, 12, 1, null);

    TimePicker tp = (TimePicker) this.findViewById(R.id.timePicker);
    // godzina 9:15
    tp.setIs24HourView(true);
    tp.setCurrentHour(new Integer(9));
    tp.setCurrentMinute(new Integer(15));
}
```

```
int getDayOfMonth()
int getMonth()
int getYear()
setCalendarViewShown(boolean)
```

```
Integer getCurrentHour()
Integer getCurrentMinute()
```


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka **Toast**

Kontrolka **Toast** umożliwia wyświetlenie na ekranie komunikatu (klasa **android.widget.Toast**).

```
Context kontekst = getApplicationContext();
CharSequence komunikat = "Nacisnąłeś przycisk";
Toast toast;
toast = Toast.makeText(kontekst, komunikat, Toast.LENGTH_SHORT);
toast.show();
```

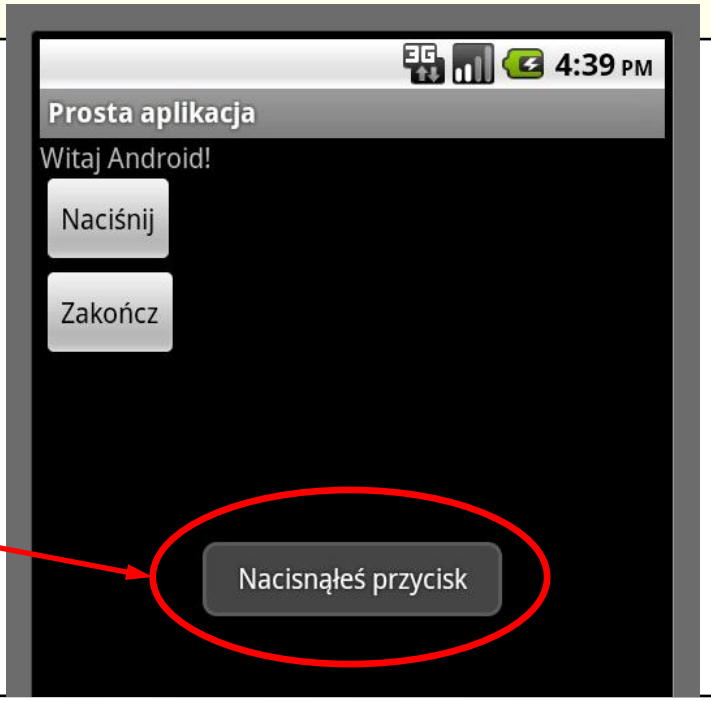
kontekst aplikacji

treść komunikatu jako:
– napis **CharSequence**
– identyfikator zasobu np. **R.string.napis**

utworzenie kontrolki

wyświetlenie kontrolki

czas wyświetlania kontrolki:
LENGTH_SHORT – krótki
LENGTH_LONG – długi



kontrolka **Toast**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka **MapView**

Kontrolka **MapView** umożliwia wyświetlanie mapy (klasa **com.google.android.maps.MapView**).

Tworzenie kontrolki:

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

    <com.google.android.maps.MapView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:enabled="true"
        android:clickable="true"
        android:apiKey="myAPIKey" />

</LinearLayout>
```

interakcja użytkownika z mapą

Maps API Key – certyfikat dla aplikacji otrzymywany po bezpłatnym zarejestrowaniu się w usłudze Google Maps

Aktywność korzystająca z kontrolki musi dziedziczyć po klasie **MapActivity** (przetwarzanie wielowątkowych żądań ładowania mapy, buforowanie itp.).

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka **`Gallery`**

Kontrolka **`Gallery`** to lista przewijana w poziomie (skupiona na środku listy) zawierająca np. galerie obrazów obsługiwane w trybie dotykowym.

Tworzenie kontrolki:

```
<Gallery
    android:id="@+id/galeria"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

Obsługa kontrolki jest podobna do obsługi kontrolki listy (strona 93):

- uzyskujemy odniesienie do galerii,
- wywołujemy metodę **`setAdapter()`** aby umieścić dane w galerii,
- rejestrujemy zdarzenia związane z wybieraniem elementów galerii.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – elementy

Kontrolka **Spinner**

Kontrolka **Spinner** to rozwijane menu.

Tworzenie kontrolki:

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Obsługa kontrolki jest podobna do obsługi kontrolki listy (strona 93).

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

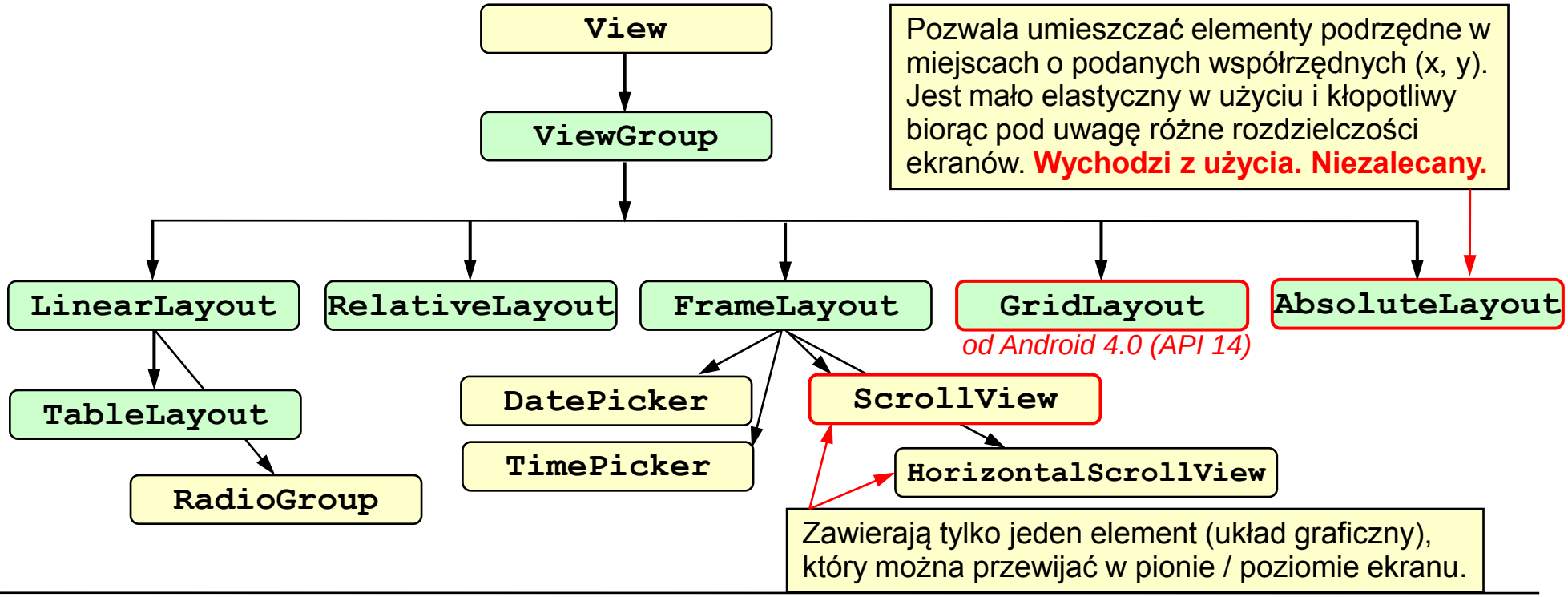
- Menadżery układu interfejsu użytkownika
- Układ LinearLayout
- Układ RelativeLayout
- Układ TableLayout
- Układ FrameLayout
- Układ z zakładkami
- Menadżery układu a orientacja urządzenia
- Narzędzie Hierarchy Viewer

Interfejs użytkownika – układy

Menadżery układu interfejsu użytkownika

Menadżery układu graficznego (*layout manager*) lub inaczej układy graficzne (*layout*) to kontenerowe klasy widoku, pełniące rolę pojemników na widoki. Zarządzają rozmiarami, rozmieszczeniem i pozycjonowaniem elementów podrzędnych (kontrolek lub innych układów). Idea podobna jak w Swing.

Klasy układów graficznych dziedziczą po klasie **`android.view.ViewGroup`**.



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Układ `LinearLayout`

Układ **`LinearLayout`** (najprostszy z dostępnych) rozmieszcza elementy podrzędne liniowo w poziomie lub w pionie. Jest to domyślny układ ustalany podczas tworzenia interfejsu użytkownika.

Wybrane właściwości układu:

dotyczy układu kontrolek w pojemniku

dotyczy widoku kontrolek

właściwość	znaczenie
`orientation`	orientacja (kierunek) rozmieszczania elementów w pojemniku: **`horizontal, vertical`**
`layout_weight`	ciężar; określa stopień ważności kontrolki w odniesieniu do innych kontrolek w pojemniku, maks. wartość to 1, dom. 0; kontrolka o ciężarze 1 zajmuje całą niezajętą przestrzeń pojemnika, 0 => zajmuje domyślny obszar
`gravity`	grawitacja; określa pozycjonowanie w obrębie widoku kontrolki: **`center`**, **`left`** (dom.), **`right`**, **`top`**, **`bottom`**, **`fill`**, **`center_horizontal`**, **`clip_horizontal`** itd.
`layout_gravity`	j.w. ale dotyczy pozycji kontrolki w pojemniku

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Układ `LinearLayout`

Plik XML definiujący układ:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/napis1" />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/napis2" />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/napis3" />

</LinearLayout>
```

orientacja pozioma

elementy potomne (podrzędne)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

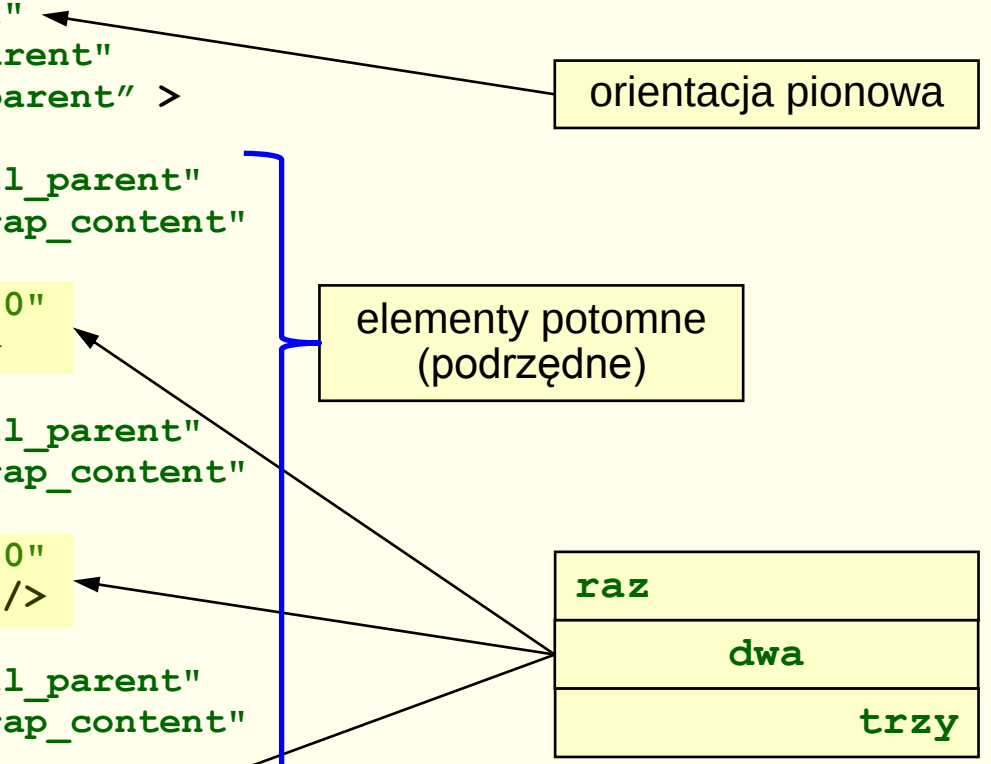
UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Układ `LinearLayout`

Plik XML definiujący układ:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="raz"
        android:layout_weight="0.0"
        android:gravity="left" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="dwa"
        android:layout_weight="1.0"
        android:gravity="center" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="trzy"
        android:layout_weight="0.0"
        android:gravity="right" />
</LinearLayout>
```



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Układ `LinearLayout`

Plik XML definiujący układ:

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

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="raz"
        android:layout_gravity="right"
        android:gravity="center" />
</LinearLayout>
```

orientacja pionowa

element potomny (podrzędny)

raz

wyśrodkowanie tekstu w kontrolce `EditText`

umieszczenie kontrolki w pojemniku z prawej strony

Interfejs użytkownika – układy

Układ `RelativeLayout`

Układ **`RelativeLayout`** (najczęściej stosowany z dostępnych) rozmieszcza elementy podrzędne względem innych kontrolek lub względem pojemnika nadrzędnego. Właściwości ustalające pozycję elementów w układzie:

wyrównuje element w stosunku do elementu **`id`**	
wartość: **`id`**	opis
`layout_above`	umieszcza ponad
`layout_below`	umieszcza pod
`layout_alignBaseline`	wyrównuje środki w poziomie
`layout_alignBottom`	wyrównuje dolne krawędzie
`layout_alignLeft`	wyrównuje lewe krawędzie
`layout_alignRight`	wyrównuje prawe krawędzie
`layout_alighTop`	wyrównuje górne krawędzie
`layout_toLeftOf`	umieszcza na lewo*
`layout_toRightOf`	umieszcza na prawo*

wyrównuje element w stosunku do pojemnika	
wartość: **`true`** / **`false`**	opis
`layout_alignParentBottom`	dolne krawędzie
`layout_alignParentLeft`	lewe krawędzie
`layout_alignParentRight`	prawe krawędzie
`layout_alignParentTop`	górne krawędzie
`layout_alignWithParentIfMissing` (patrz *)	gdy brak lewego lub prawego el. to przyjmowany jest nadrzędny
`layout_centerHorizontal`	wyśrod. poziomo
`layout_centerInParent`	wyśrodkowuje
`layout_centerVertical`	wyśrod. pionowo

* wymagają użycia właściwości **`align`**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Układ `RelativeLayout`

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"  android:layout_height="wrap_content" >
    <TextView android:id="@+id/tv_login"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Nazwa użytkownika"
        android:layout_alignParentTop="true" />
    <EditText android:id="@+id/et_login"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/tv_login" />
    <TextView android:id="@+id/tv_haslo"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hasło"
        android:layout_below="@id/et_login" />
    <EditText android:id="@+id/et_haslo"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/tv_haslo" />
    <TextView android:id="@+id/tv_uwaga"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="Uzyskanie dostępu wymaga podania loginu i hasła..." />
</RelativeLayout>
```

etykieta tekstowa wyrównana do górnej krawędzi pojemnika

poniżej poprzedniego elementu

etykieta tekstowa wyrównana do dolnej krawędzi pojemnika

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Układ `TableLayout`

Układ **`TableLayout`** rozmieszcza elementy podrzędne w wierszach i kolumnach (jak w tabeli). Liczba kolumn tabeli jest określania na podstawie wiersza zawierającego największą liczbę elementów. Kolumny numerowane są od zera. Układ dziedziczy właściwości po układzie **`LinearLayout`**, uzupełniając je o nowe właściwości:

właściwość	znaczenie
`layout_width`	może przyjąć tylko wartość **`fill_parent`**, nie może przyjmować wartości **`wrap_content`**
`stretchColumns="0,1,2"`	rozciągnięcie kolumn o podanych numerach
`shrinkColumns="0,1,2"`	zwężenie kolumn o podanych numerach, gdy inne kolumny wymagają więcej miejsca
`collapseColumns="1,2"`	wybrane kolumny stają się niewidoczne
`layout_span`	rozciągnięcie komórki na wiele kolumn
`padding="40px"`	dotyczy widoku, określa odstęp wewnątrz zawartości komórki lub kontrolki (kontrola przestrzeni pomiędzy zewn. granicami widoku, a jego treścią) (**`leftPadding`, `rightPadding`, `topPadding`, `bottomPadding`**)
`layout_margin`	j.w. ale dotyczy pojemnika

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Układ **TableLayout**

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Imię:" />
        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Jan" />
    </TableRow>
    <TableRow>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Nazwisko:" />
        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Kowalski" />
    </TableRow>
</TableLayout>
```

węzły **TableRow** opisują poszczególne komórki tabeli

pierwszy wiersz tabeli

drugi wiersz tabeli

Imię:	**Jan**
Nazwisko:	**Kowalski**

Interfejs użytkownika – układy

Układ **TableLayout**

Plik XML definiujący układ:

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="0,1,2" >

    <EditText android:text="Imię i nazwisko:" />

    <TableRow>
        <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="Jan" />
        <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="Kowalski" />
    </TableRow>

</TableLayout>
```

Jako element podrzędny tabeli, zamiast elementu **TableRow** można umieścić dowolny element **View**, np. **EditText**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Układ `FrameLayout`

Układ **`FrameLayout`** służy do dynamicznego wyświetlania (zmieniania) pojedynczego elementu. Jeżeli pojemnik zawiera wiele elementów, to:

- jeden element jest widoczny, a pozostałe są niewidoczne
- elementy tworzą stos – widoczny jest element z wierzchołka stosu (**`start`**).

Położeniem elementu w kontenerze steruje właściwość **`android:layout_gravity`**:

wartość właściwości	opis
`top`	na górze
`bottom`	na dole
`left`	z lewej strony
`right`	z prawej strony
`center_vertical`	na środku w poziomie
`center_horizontal`	na środku w pionie
`center`	na środku w pionie i poziomie
`start`	na wierzchołku stosu
`end`	na dnie stosu

wartość właściwości	opis
`fill`	wypełnia cały kontener
`fill_vertical`	wypełnia kontener w poziomie
`fill_horizontal`	wypełnia kontener w pionie

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Układ **FrameLayout**

Plik XML definiujący układ:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/frmLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <ImageView
        android:id="@+id/obrazek1"
        android:src="@drawable/plik1"
        android:scaleType="fitCenter"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
    <ImageView
        android:id="@+id/obrazek2"
        android:src="@drawable/plik2"
        android:scaleType="fitCenter"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:visibility="gone" />

</FrameLayout>
```

identyfikator pliku **plik1.png**

element domyślnie widoczny

element niewidoczny

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Przykład wykorzystania układu **FrameLayout**

```
protected void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   ImageView img1 = (ImageView) this.findViewById(R.id.obrazek1);
   ImageView img2 = (ImageView) this.findViewById(R.id.obrazek2);

   img1.setOnClickListener(new OnClickListener() {
      public void onClick(View view) {
         ImageView img2 =
            (ImageView) FrameLayoutActivity.this.findViewById(R.id.obrazek2);
         img2.setVisibility(View.VISIBLE);
         view.setVisibility(View.GONE);
      }
   });

   img2.setOnClickListener(new OnClickListener() {
      public void onClick(View view) {
         ImageView img1 =
            (ImageView) FrameLayoutActivity.this.findViewById(R.id.obrazek1);
         img1.setVisibility(View.VISIBLE);
         view.setVisibility(View.GONE);
      }
   });
}
```

załadowanie układu ze strony 114

tutaj nie używać przestrzeni nazw **R.drawable**

obiekty nasłuchujące reagują na kliknięcie elementów **ImageView**

Jednocześnie widoczny jest tylko jeden z elementów **ImageView**. Kliknięcie obrazka (**onClick**) powoduje jego ukrycie i wyświetlenie drugiego w tym samym miejscu.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

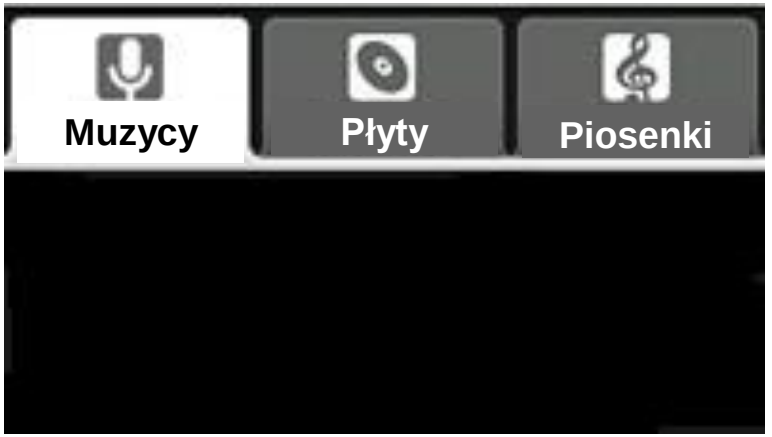
Interfejs użytkownika – układy

Układ z zakładkami

Układ graficzny z zakładkami można utworzyć przy użyciu węzłów **`TabHost`** i **`TabWidget`**. Węzeł **`TabHost`** powinien zawierać wewnątrz węzły **`TabWidget`** i **`FrameLayout`** (patrz strona 117).

Dla każdej zakładki wymagane jest przygotowanie dwóch ikon, symbolizujących stan aktywnej i nieaktywnej zakładki.

Ze względu na elastyczność wykorzystania układu, korzystnie jest przyjąć, że z każdą zakładką będzie związana oddzielna aktywność. Główna aktywność aplikacji dziedziczy po klasie **`TabActivity`**.



Wygląd przykładowego układu zdefiniowanego na stronach 117 i 118.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Układ z zakładkami

Plik XML definiujący układ:

```
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="5dp">
        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp" />
    </LinearLayout>
</TabHost>
```

układ **LinearLayout** zawiera dwa elementy ułożone poziomo:
– element **TabWidget**
– układ **FrameLayout**

wymagane nazwy identyfikatorów!

w układzie **FrameLayout** wyświetlana jest dynamicznie zawartość poszczególnych zakładek

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Zarys obsługi układu z zakładkami

```
public void onCreate(Bundle savedInstanceState)
{
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);

   Resources res = getResources();  // obiekt zasobów
   TabHost tabHost = getTabHost();  // TabHost dla aktywności
   TabHost.TabSpec spec;  // TabSpec używany dla każdej zakładki
   Intent intencja;       // intencja używana dla każdej zakładki

   // utworzenie intencji do uruchamiania aktywności zakładki
   intencja = new Intent().setClass(this, Aktywnosc1.class);

   // inicjalizacja TabSpec dla zakładki
   spec = tabHost.newTabSpec("muzycy").setIndicator("Muzycy",
      res.getDrawable(R.drawable.ic_tab_nazwiska))
      .setContent(intencja);  // id, tekst na zakładce, ikona, intencja
   tabHost.addTab(spec); // dodanie TabSpec do TabHost i zw. z TabWidget

   // .....

   tabHost.setCurrentTab(0);  // domyślnie wyświetlana pierwsza zakładka
}                             // (numeracja od 0)
```

kod powtarzany dla każdej zakładki

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Menadżery układu graficznego a orientacja urządzenia

Poprawne wyświetlanie układów graficznych we wszystkich konfiguracjach wymaga, aby utworzyć dla nich oddzielne katalogi przechowujące dostosowane definicje układu. System automatycznie wykrywa orientację urządzenia i wczytuje układ z odpowiedniego folderu. Ta sama reguła dotyczy zawartości katalogu **`drawable`**.

Urządzenie może znajdować się w jednej z trzech konfiguracji:

- pionowa (*portrait*) → zasoby w katalogu **`/res/layout-port`**
- pozioma (*landscape*) → zasoby w katalogu **`/res/layout-land`**
- wyświetlanie kwadratowego obrazu (*square*) → zasoby w katalogu **`/res/layout-square`**

Zawsze powinien być dostępny domyślny katalog zasobów **`/res/layout`**.

Programowe wymuszenie działania aplikacji w wybranej orientacji:

```
// kod dla metody onCreate aktywności
import android.content.pm.ActivityInfo;
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
```


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Interfejs użytkownika – układy

Narzędzie Hierarchy Viewer

Narzędzie Hierarchy Viewer służy do optymalizacji układów graficznych i usuwania błędów w interfejsie użytkownika z poziomu układu graficznego.

Narzędzie to wczytuje układ graficzny i pokazuje hierarchię zdefiniowanych widoków w postaci drzewa.

Układ graficzny jest analizowany pod względem określenia możliwych problemów oraz jest optymalizowany pod kątem zmniejszenia liczby widoków (poprawienie wydajności działania aplikacji).

Narzędzie Hierarchy Viewer jest dostępne w katalogu **`/tools/hierarchyviewer.bat`**.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

- Koncepcja menu
- Tworzenie menu
- Tworzenie grup menu
- Odpowiedź na wybór elementu menu
- Inne rodzaje menu
- Menu kontekstowe
- Tworzenie menu w XML

Tworzenie i obsługa menu

Koncepcja menu

Menu są widziane jako zasoby wczytywane z plików XML. Dla każdego elementu menu jest tworzony identyfikator zasobów.

Do obsługi menu służy klasa **`android.view.Menu`**. Elementy menu są reprezentowane przez obiekty klasy **`android.view.MenuItem`**, a elementy podmenu **`android.view.SubMenu`**.

Rodzaje menu:

- menu standardowe
- podmenu
- menu kontekstowe
- menu w postaci ikon
- menu drugorzędne (**`Menu.CATEGORY_SECONDARY`**) **`0x30000`**
- menu alternatywne (**`Menu.CATEGORY_ALTERNATIVE`**) **`0x40000`**
- menu systemowe (**`Menu.CATEGORY_SYSTEM`**) **`0x20000`** – elementy tego menu są umieszczane automatycznie przez system
- menu kontenerowe (**`Menu.CATEGORY_CONTAINER`**) **`0x10000`**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

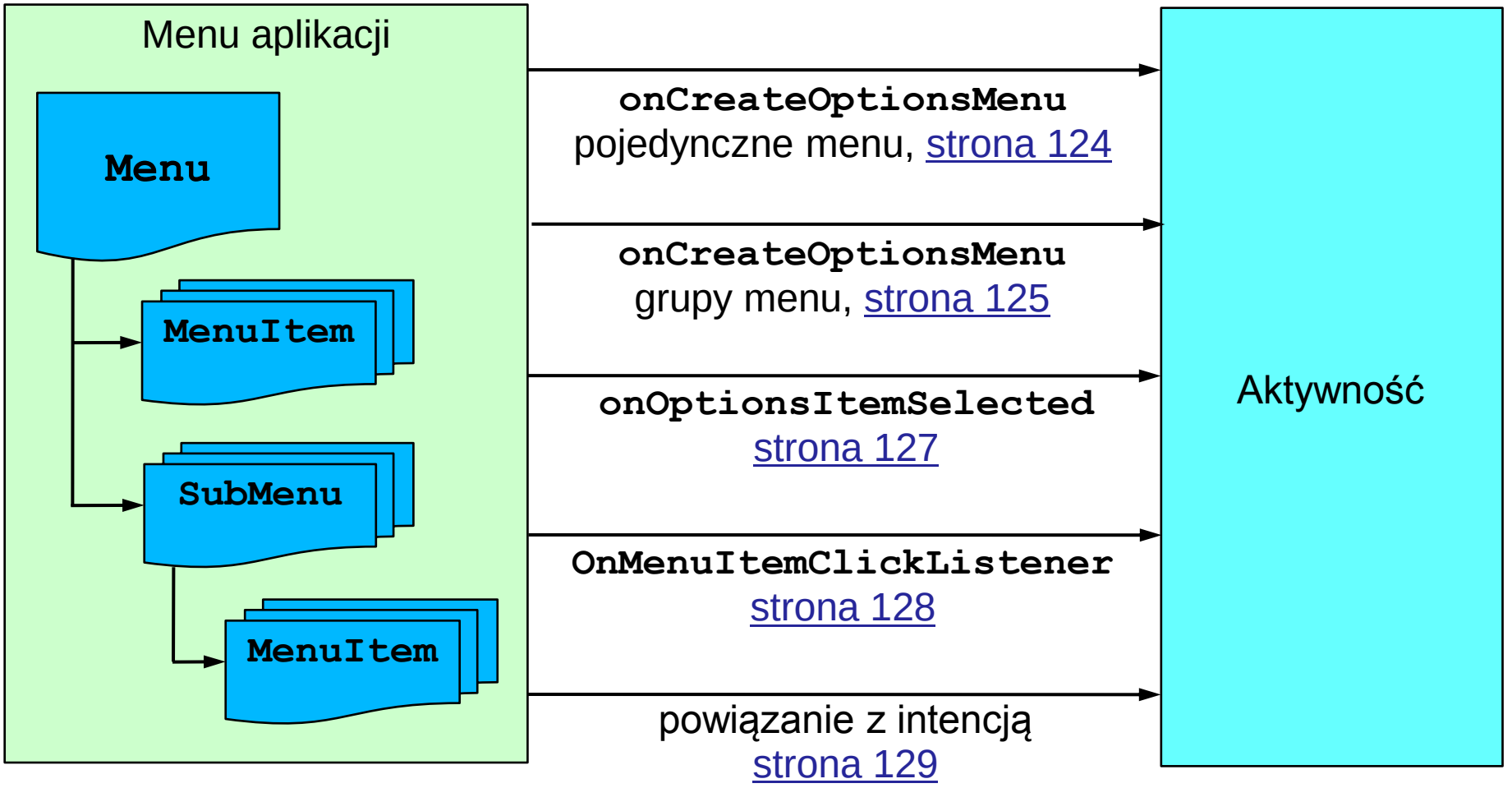
Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Schemat obsługi menu

Powiązania pomiędzy klasami i funkcjami dotyczącymi menu



Tworzenie i obsługa menu

Tworzenie menu

Aktywność jest powiązana z pojedynczym menu (obiektem klasy **`Menu`**).
Menu jest tworzone dla aktywności i przekazywane jako argument do metody **`onCreateOptionsMenu`**. Metoda ta wypełnia menu zestawem elementów menu – zwrócenie wartości **`true`** oznacza uwidocznienie menu (**`false`** oznacza menu niewidoczne).

Element menu posiada: identyfikator swojej grupy elementów, własny unikalny identyfikator, numer pozycji w menu, nazwę (tytuł).

```
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
    // wywołanie metody nadrzędnej – systemowe elementy menu
    super.onCreateOptionMenu(menu);

    menu.add(0, 1, 1, "Polecenie 1");
    menu.add(0, 2, 2, "Polecenie 2");
    menu.add(0, 3, 3, "Polecenie 3");

    return true;  // powoduje wyświetlenie menu
}
```

```
id grupy
id elementu   } opcjonalne (Menu.NONE)
nr pozycji
nazwa         - może pochodzić z zasobów
```


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Tworzenie grup menu

Przykład utworzenia dwóch grup menu:

```
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
   // wywołanie metody nadrzędnej – systemowe elementy menu
   super.onCreateOptionMenu(menu);

   int grupa1 = 1; // grupa 1
   menu.add(grupa1, 1, 1, "Polecenie 1.1");
   menu.add(grupa1, 2, 2, "Polecenie 1.2");
   menu.add(grupa1, 3, 3, "Polecenie 1.3");

   int grupa2 = 2; // grupa 2
   menu.add(grupa2, 4, 4, "Polecenie 2.1");
   menu.add(grupa2, 5, 5, "Polecenie 2.2");
   menu.add(grupa2, 6, 6, "Polecenie 2.3");

   return true;  // powoduje wyświetlenie menu
}
```

Numeracja **`id elementu`** i **`nr pozycji`** są niezależne i zachowują ciągłość.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Obsługa grup menu

Grupowanie elementów menu pozwala wykorzystywać metody odwołujące się do identyfikatorów grup i kontrolować elementy menu w grupie:

- **`removeGroup(int id)`** – usuwa wszystkie elementy z grupy **`id`**
- **`setGroupEnabled(int id, boolean enabled)`** – włącza lub wyłącza dostępność wszystkich elementów menu z grupy **`id`**
- **`setGroupVisible(int id, boolean visible)`** – ustala widoczność wszystkich elementów menu z grupy **`id`**
- **`setGroupCheckable(int id, boolean checkable, boolean exclusive)`** – wstawia znak zaznaczenia obok wszystkich elementów menu z grupy **`id`** (jeżeli flaga **`exclusive`** jest ustawiona na **`true`** to flagę **`checkable`** będzie można ustawiać dla tylko jednego elementu z grupy metodą **`setCheckable`** z klasy **`MenuItem`**)

Zarządzanie poszczególnymi elementami **`MenuItem`**:

```
setCheckable(boolean checkable)
setChecked(boolean checked)
setEnabled(boolean enabled)
setVisible(boolean visible)
```


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Odpowiedź na wybór elementu menu – metoda 1

Zalecanym sposobem odpowiedzi na wybór (kliknięcie) elementu menu jest przesłonięcie metody **`onOptionsItemSelected`** w klasie aktywności. Wybór elementu menu powoduje wywołanie tej metody i przekazanie do niej identyfikatora wybranego elementu menu.

Metoda zalecana

```
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    switch (item.getItemID()) // identyfikacja wybranego elementu
    {
        case ...
            return true;  // dla kliknietych (przetwarzanych)
                          // elementów menu
        case ...
            return true;
        default:
            // dla nieprzetwarzanych elementów menu
            return super.onOptionsItemSelected(item);
    }
}
```


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Odpowiedź na wybór elementu menu – metoda 2

Innym sposobem odpowiedzi na wybór elementu menu jest odpowiedź za pomocą obiektu nasłuchującego (nasłuchiwacza). Zarejestrowanie nasłuchiwacza pełni rolę tzw. wywołania zwrotnego. Po wybraniu elementu menu wywołana zostaje metoda **`onMenuItemClick`** interfejsu **`OnMenuItemClickListener`**.

```
public class MyResponse implements OnMenuItemClickListener
{
    @Override
    boolean onMenuItemClick(MenuItem item)
    {
        // czynności związane
        // z poleceniem menu...
        return true;
    }
}
```

implementacja interfejsu **`OnMenuClickListener`**

kliknięcie elementu menu powoduje wywołanie kodu nasłuchiwacza: ma on pierwszeństwo przed metodą **`onOptionsItemSelected`**

```
MyReponse myResponse = new MyResponse();
menuItem.setOnMenuItemClickListener(myResponse);
```

ustalenie nasłuchiwacza

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Odpowiedź na wybór elementu menu – metoda 3

Kolejnym sposobem odpowiedzi na wybór elementu menu jest użycie intencji.

Element menu należy powiązać z intencją za pomocą metody **`setIntent(intent)`** klasy **`MenuItem`**.

Jeżeli element menu nie jest obsługiwany żadną dwóch omówionych wcześniej metod (strona 127 i 128), to domyślnie nastąpi przywołanie intencji i wywołanie metody **`startActivity(intent)`**.

Warunkiem zadziałania tej metody odpowiedzi jest:

- nieprzesłanianie metody **`onOptionsItemSelected`**
- jeżeli metoda **`onOptionsItemSelected`** została przesłonięta, konieczne jest umieszczenie w jej kodzie instrukcji

```
return super.onOptionsItemSelected(item);
```

dla nieprzetwarzanych elementów menu

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Inne rodzaje menu

- menu rozszerzone
  Element menu zatytułowany "**`Więcej`**" / "**`More`**" - pojawia się automatycznie, jeżeli menu zawiera więcej elementów niż mieści się na ekranie. Ukryte pod nim menu rozszerzone nie może zawierać ikon (jeśli zawiera, to ikony nie są wyświetlane).
- menu w postaci ikon
  Elementy menu zawierające ikony nie obsługują funkcji ich zaznaczania (*checkable*). Przypisanie ikony do elementu menu:

```
// utworzenie standardowego elementu menu
Menu menu = null;
MenuItem item = menu.add(0, 2, 2, "Uruchom");

// ustalenie ikony dla elementu menu
item.setIcon(R.drawable.balony);  // id zasobu pliku balony.png
```

metoda **`setIcon`** klasy **`MenuItem`** ustala jako ikonę obrazek o podanym identyfikatorze zasobu z katalogu **`/res/drawable`**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Inne rodzaje menu

- podmenu

  W obiekcie klasy **`Menu`** może znajdować się wiele obiektów klasy **`SubMenu`** dodawanych metodą **`Menu.addSubMenu`** (podobnie jak w menu standardowym). Podmenu nie może zawierać kolejnego podmenu. Ikony przypisane do elementów podmenu są ignorowane.

```
Menu menu = null;
SubMenu sm = menu.addSubMenu(100, 101, Menu.NONE, "podmenu");
sm.add(100, 102, 102, "podelement 1");
sm.add(100, 103, 103, "podelement 2");
sm.add(100, 104, 104, "podelement 3");
```

- menu systemowe

  Elementy menu systemowego są dołączane do tworzonego menu dzięki wywołaniu metody **`onCreateOptionsMenu`** z klasy nadrzędnej. Menu systemowe może być zaimplementowane w przyszłych wersjach systemu Android – obecnie nie występuje.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Inne rodzaje menu

- menu kontekstowe
  Menu kontekstowe jest uruchamiane poprzez tzw. długie kliknięcie (przytrzymanie przycisku). Nie obsługuje ono ikon, podmenu ani skrótów reprezentowane przez klasę **`ContextMenu`**. Elementy dodawane są takimi samymi metodami jak w klasie **`Menu`** (metody **`add`**).
  Różnica w stosunku do klasy **`Menu`** to obiekt-właściciel: menu standardowe należy do aktywności, a menu kontekstowe należy do widoku.

  Implementacja menu kontekstowego:

1. Zarejestrowanie widoku dla menu kontekstowego w metodzie **`onCreate`** aktywności.
2. Wypełnienie menu poleceniami w metodzie **`onCreateContextMenu`**.
3. Zdefiniowanie odpowiedzi na kliknięcia poszczególnych elementów menu kontekstowego.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Implementacja menu kontekstowego

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    TextViev tv = (TextView) this.findViewById(R.id.tv1);
    registerForContextMenu(this.getTextView());
}

@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo mi)
{
    menu.setHeaderTitle("Przykładowe menu kontekstowe");
    menu.add(200, 200, 200, "element 1");
}

@Override
public boolean onContextItemSelected(MenuItem item)
{
    // podobnie jak w menu standardowym
    // patrz strona 127
}
```

krok 1: rejestrowanie widoku dla menu kontekstowego (rejestrowanie widoku **TextView** jako właściciela menu)

domyślnie tworzony obiekt menu i widok który wygenerował wywołanie zwrotne

krok 2: zapełnianie menu kontekstowego elementami

krok 3: tworzenie odpowiedzi na kliknięcia elementów menu kontekstowego

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Inne rodzaje menu

- menu alternatywne
  Elementy menu alternatywnego mogą stanowić część menu standardowego, podmenu i menu kontekstowego.
  Menu alternatywne to część systemu komunikacji pomiędzy aplikacjami (np. menu jednej aplikacji może być umieszczone wewnątrz drugiej aplikacji). Wybranie menu alternatywnego powoduje uruchomienie docelowej aplikacji lub aktywności za pośrednictwem adresu URL (adres ten przekazuje dane). Aktywność wykorzysta adres URL używając do tego intencji.

- menu dynamiczne
  Zawartość menu dynamicznego może się zmieniać (inne rodzaje menu to menu statyczne).
  Menu dynamiczne tworzone jest metodą **`onPrepareOptionMenu`**, która jest wywoływana za każdym razem gdy wybierane jest menu dynamiczne.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Tworzenie menu za pomocą plików XML

Menu traktowane są jako zasoby – dlatego można je definiować w plikach XML. Metoda taka ma zalety w stosunku do tworzenia menu w kodzie Java: możliwość nadania nazwy menu, automatyczne tworzenie kolejności menu, automatyczne przyznawanie identyfikatorów menu, lokalizacja tekstu menu (wersje językowe).

Tworzenie menu w oparciu o plik XML:

1. Zdefiniowanie w katalogu **`/res/menu`** pliku XML ze znacznikami menu (nazwa pliku dowolna). Automatycznie zostanie wygenerowany identyfikator zasobu dla tego pliku.
2. Wczytanie pliku do menu za pomocą identyfikatora zasobu pliku.
3. Zdefiniowanie odpowiedzi na kliknięcia poszczególnych elementów menu za pomocą identyfikatorów zasobów wygenerowanych dla poszczególnych elementów menu.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Plik XML z zasobami menu – krok 1

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   <group android:id="@+id/menuGroup_Main" >
      <item android:id="@+id/menu_testPick"
         android:orderInCategory="5"
         android:title="Część testowa" />
      <item android:id="@+id/menu_testGetContent"
         android:orderInCategory="5"
         android:title="Test pobierania treści" />
      <item android:id="@+id/menu_clear"
         android:orderInCategory="10"
         android:title="Wyczyść" />
      <item android:id="@+id/menu_dial"
         android:orderInCategory="7"
         android:title="Dzwoń" />
      <item android:id="@+id/menu_test"
         android:orderInCategory="4"
         android:title="@+string/test" />
      <item android:id="@+id/menu_show_browser"
         android:orderInCategory="5"
         android:title="Wyświetl przeglądarkę" />
   </group>
</menu>
```

plik **/res/menu/moje_menu.xml**

Plik rozpoczyna się od znacznika **<menu>**, po którym występuje seria znaczników **<group>** opisujących grupy elementów menu.
Jeżeli występuje tylko jedna grupa, to otrzyma ona identyfikator **menuGroup_main**.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Wczytanie plików XML zasobów menu – krok 2

Dla pliku o nazwie **`/res/menu/moje_menu.xml`** zostanie wygenerowany identyfikator zasobu pliku **`R.menu.moje_menu`** oraz identyfikatory elementów menu.

Klasa **`android.view.MenuInflater`** służy do umieszczania obiektów klasy **`Menu`** z plików XML.

```
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
    // pobranie obiektu MenuInflater z aktywności
    MenuInflater inflater = getMenuInflater();

    // przesłanie zawartość pliku XML do menu
    inflater.inflate(R.menu.moje_menu, menu);

    return true;  // powoduje że menu jest widoczne
}
```


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Tworzenie i obsługa menu

Tworzenie odpowiedzi dla elementów menu XML – krok 3

Nazwy elementów menu z pliku XML spowodowały wygenerowanie identyfikatorów elementów menu w przestrzeni nazw **`R.id`**.

```
public void onOptionsItemSelected(MenuItem item)
{
    this.appendMenuItemText(item);

    switch (item.getItemId())
    {
        case R.id.menu_clear:
            this.emptyText(); break;
        case R.id.menu_dial:
            this.dial(); break;
        case R.id.menu_testPick:
            IntentsUtils.invokePick(this); break;
        case R.id.menu_testGetContent:
            IntentsUtils.invokeGetContent(this); break;
        case R.id.menu_show_browser:
            IntentsUtils.invokeBrowser(this); break;
    }
}
```


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Materiały przygotowano w ramach projektu
„Inżynier pilnie poszukiwany”

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Literatura i źródła

1. S. Hashimi, S. Komatineni, D. MacLean: „*Android 2. Tworzenie aplikacji*”, Helion, Gliwice 2010.
2. E. Burnette: „*Hello Android. Programowanie na platformę Google dla urządzeń mobilnych. Wydanie III*”, Helion, Gliwice 2011.
3. S. Conder, L. Darcey: „*Android. Programowanie aplikacji na urządzenia przenośne. Wydanie II*”, Helion, Gliwice 2011.
4. W.F. Ableson, R. Sen, C. King: „*Android w akcji. Wydanie II*”, Helion, Gliwice 2011.
5. dokumentacja SDK on-line http://developer.android.com
6. strona internetowa Android.com http://www.android.com
7. strona internetowa Eclipse http://www.eclipse.org
8. strona internetowa Oracle Java http://www.oracle.com/java
9. strona internetowa Android Market https://market.android.com
10. kod źródłowy systemu Android http://source.android.com http://android.git.kernel.org