Android @ Szczecin 2011

Konrad Malawski
konrad.malawski@java.pl

31 lipca 2011

Let's move the Java world!

GeeCON

lunar logic
polska

@ktosopl

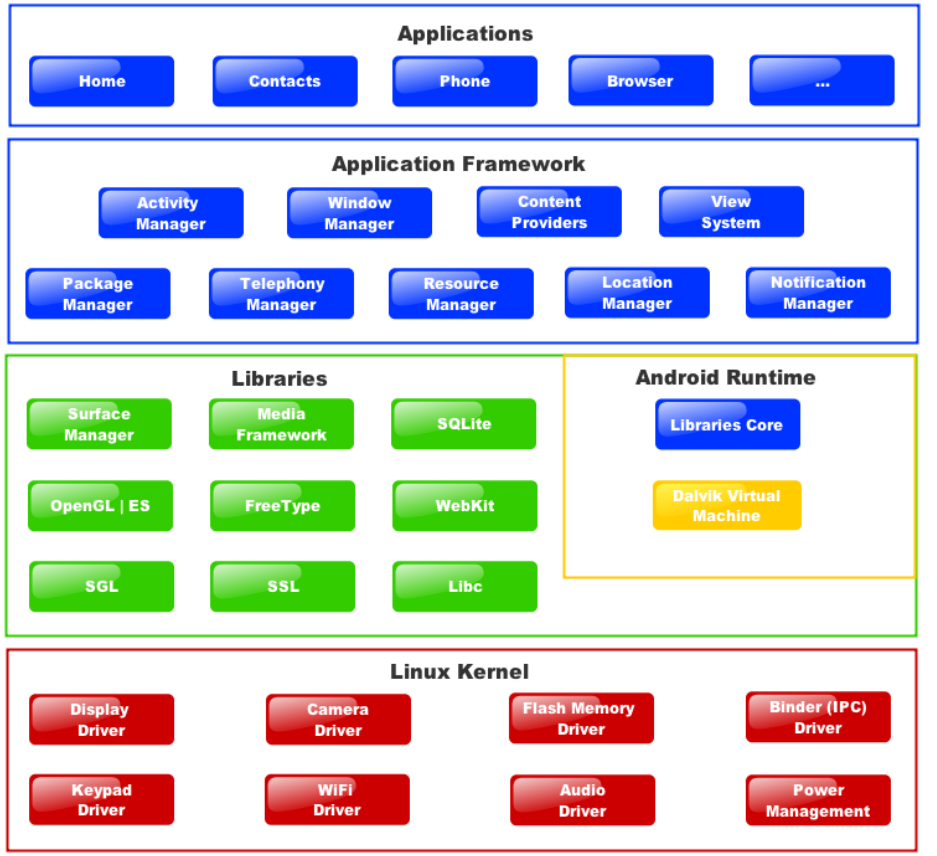
blog.project13.pl

github.com/ktoso

CODERETREAT

KRAKÓW - APRIL 2nd 2011 - START: 08:00

Architektura



Rysunek: Architektura systemu Android

Słowa na dziś:

- Activity
- View
- Service
- Intent
- PendingIntent
- _____Manager

Przykłady Managerów

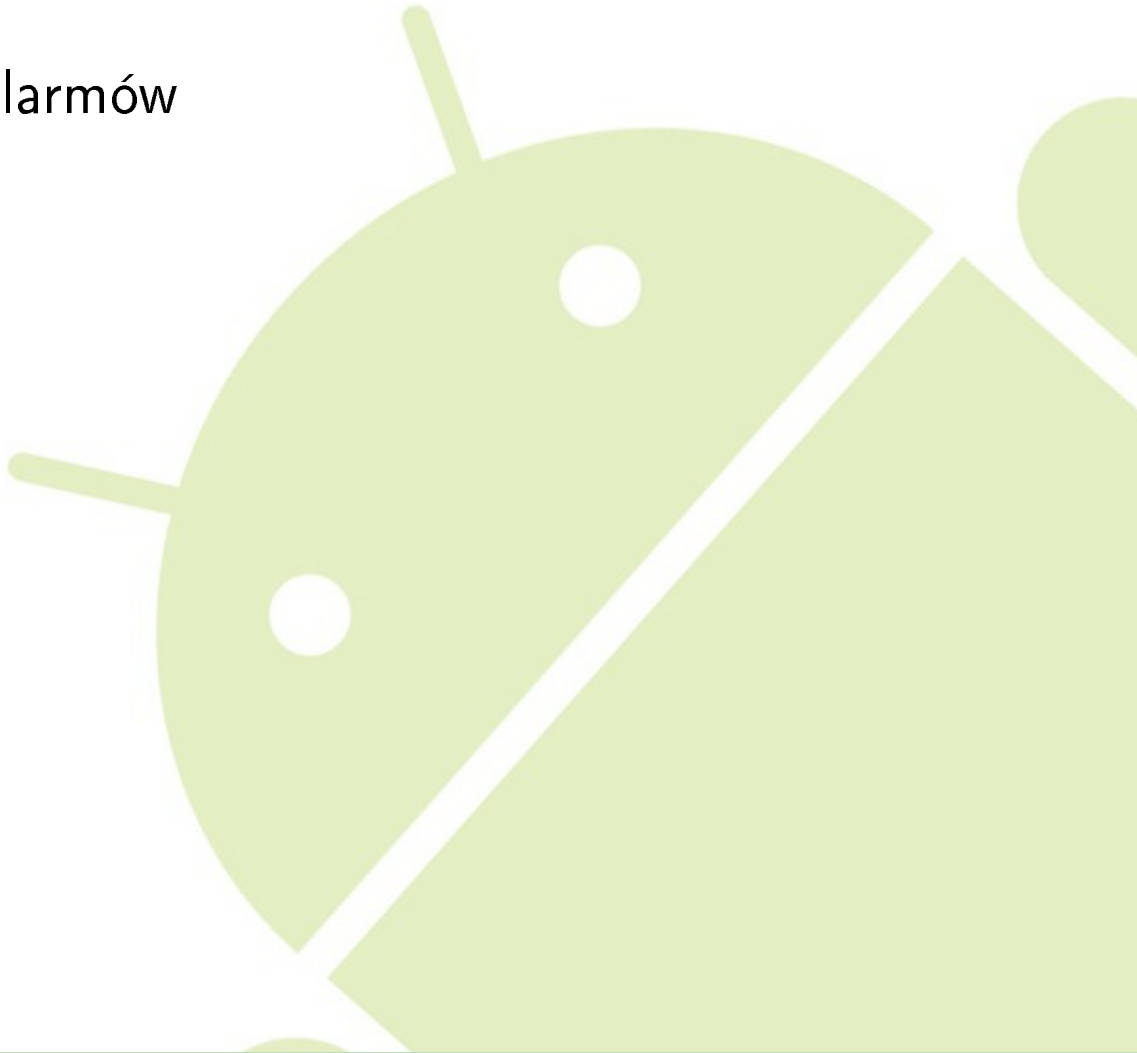
- **NotificationManager** - dla powiadomień

Przykłady Managerów

- **NotificationManager** - dla powiadomień
- **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)

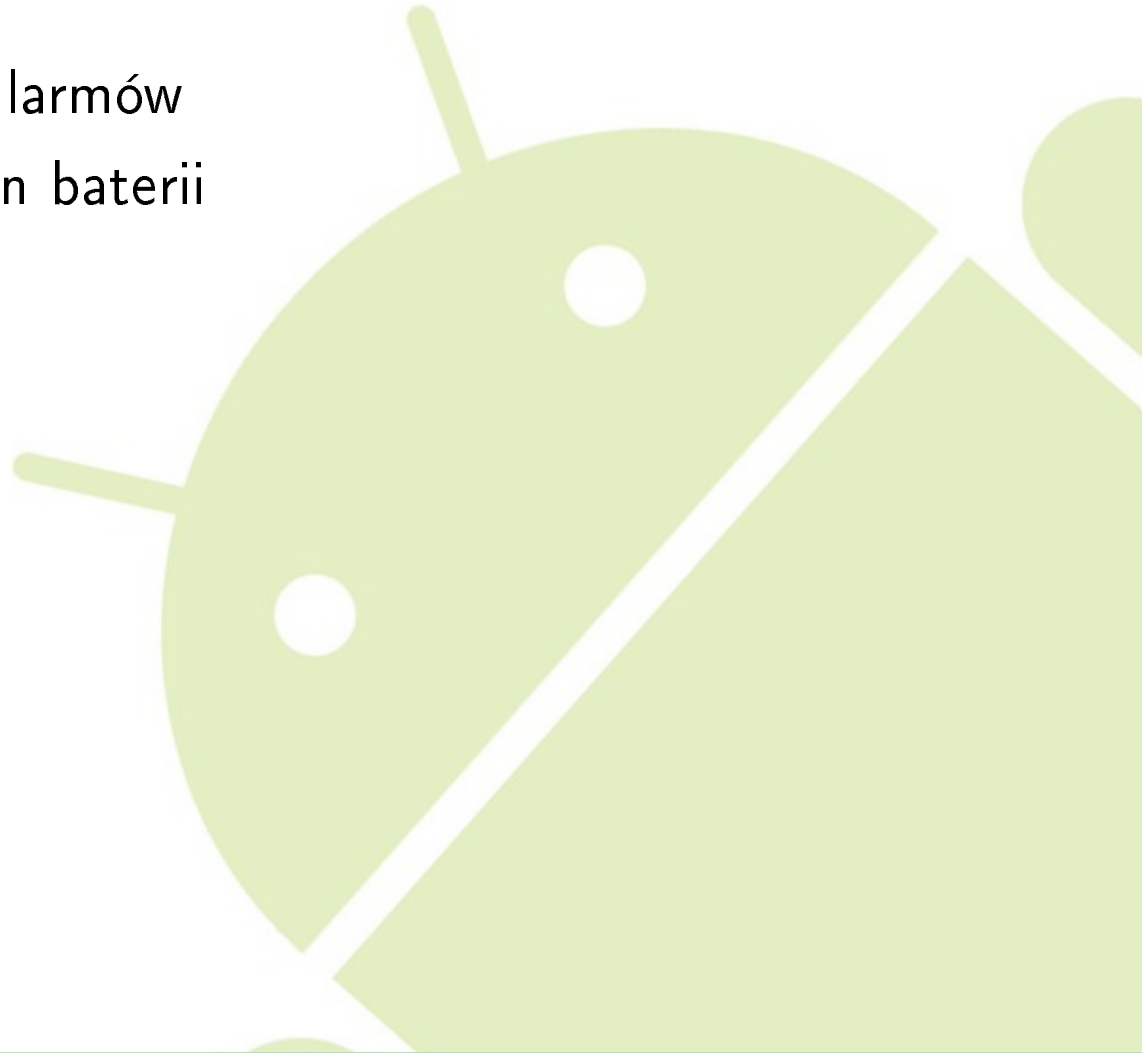
Przykłady Managerów

- **NotificationManager** - dla powiadomień
- **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)
- **AlarmManager** - dla alarmów



Przykłady Managerów

- **NotificationManager** - dla powiadomień
- **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)
- **AlarmManager** - dla alarmów
- **BatteryManager** - stan baterii



Przykłady Managerów

- **NotificationManager** - dla powiadomień
- **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)
- **AlarmManager** - dla alarmów
- **BatteryManager** - stan baterii
- **ClipboardManager** - schowek nie jest taki trywialny, ma całe własne API

Przykłady Managerów

- **NotificationManager** - dla powiadomień
- **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)
- **AlarmManager** - dla alarmów
- **BatteryManager** - stan baterii
- **ClipboardManager** - schowek nie jest taki trywialny, ma całe własne API
- **ConnectivityManager** - możemy sprawdzić czy Internet jest dostępny. Via 3G czy WiFi?

Przykłady Managerów

- **NotificationManager** - dla powiadomień
- **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)
- **AlarmManager** - dla alarmów
- **BatteryManager** - stan baterii
- **ClipboardManager** - schowek nie jest taki trywialny, ma całe własne API
- **ConnectivityManager** - możemy sprawdzić czy Internet jest dostępny. Via 3G czy WiFi?
- **LocationManager** - dostęp do pozycji (również GPS)

Przykłady Managerów

- **NotificationManager** - dla powiadomień
- **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)
- **AlarmManager** - dla alarmów
- **BatteryManager** - stan baterii
- **ClipboardManager** - schowek nie jest taki trywialny, ma całe własne API
- **ConnectivityManager** - możemy sprawdzić czy Internet jest dostępny. Via 3G czy WiFi?
- **LocationManager** - dostęp do pozycji (również GPS)
- nowe: **FragmentManager** - nowe api do umieszczania wiele activity na jednym ekranie (upraszczając)

Przykłady Managerów

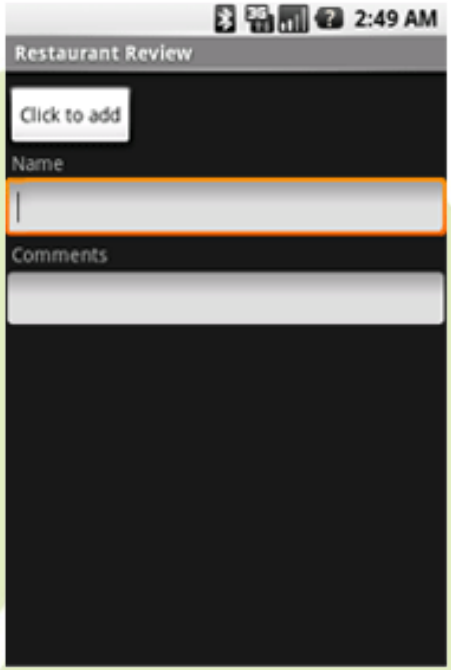
- **NotificationManager** - dla powiadomień
- **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)
- **AlarmManager** - dla alarmów
- **BatteryManager** - stan baterii
- **ClipboardManager** - schowek nie jest taki trywialny, ma całe własne API
- **ConnectivityManager** - możemy sprawdzić czy Internet jest dostępny. Via 3G czy WiFi?
- **LocationManager** - dostęp do pozycji (również GPS)
- nowe: **FragmentManager** - nowe api do umieszczania wiele activity na jednym ekranie (upraszczając)
- nowe: **DownloadManager** - zaawansowany menadżer pobierania plików w tle (w **Honeycomb**)

Good News, nie będzie tsunami wyliczeń nazw klas

Obiecuję, że nie będzie już slajdów takich jak poprzedni :-)
Konkrety, obrazki, kodzenie!

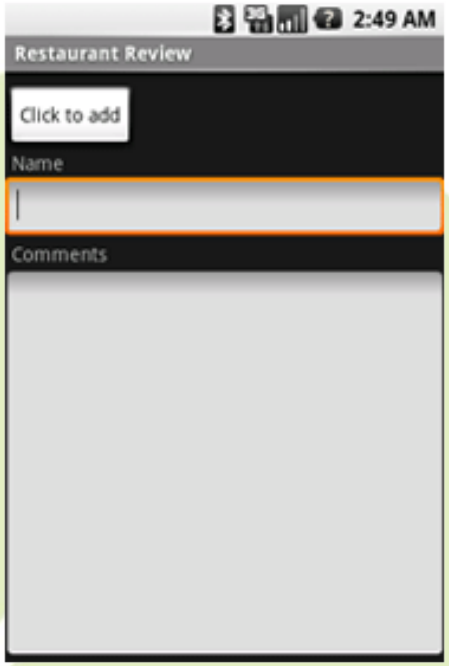
LinearLayout + EditText + wrap_content

```
<LinearLayout
 android:id="@+id/root_layout"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical">
  <!-- ... -->

  <TextView android:id="..." ... />
  <EditText android:id="..."
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
  />
</LinearLayout>
```



LinearLayout + TextView z fill_parent

```
<LinearLayout
 android:id="@+id/root_layout"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical">
  <!-- ... -->

  <TextView android:id="..." ... />
  <EditText android:id="..."
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
  />
</LinearLayout>
```



Panowanie nad Layoutami

- Istnieje jeszcze **match_parent**, pojawił się w nowszych wersjach Android, jest analogiczny do fill_parent.
- W przypadku gdy chcemy ustalać "jaką część widoku ma zajmować pewien element", korzystamy z **android:layout_weight**, przyjmuje on liczby *(default = 1)*.

Ważne ViewGroups:

- **FrameLayout** - Layout dla tylko jednego elementu, najprostszy
- **LinearLayout** - Wystarczający dla prostych widoków, kombinowanie kilku LinearLayout daje przyjemne efekty
- **ListView** - Sam dba o scrollowanie
- **TabHost** - Może mieć zakładki

Ważne ViewGroups:

- **FrameLayout** - Layout dla tylko jednego elementu, najprostszy
- **LinearLayout** - Wystarczający dla prostych widoków, kombinowanie kilku LinearLayout daje przyjemne efekty
- **ListView** - Sam dba o scrollowanie
- **TabHost** - Może mieć zakładki
- **Spinner**, **Gallery**, **GridView**, **RelativeLayout**, **ScrollView**, **SurfaceView**, **TableLayout**, **ViewFlipper**, **ViewSwitcher**...

R, as in Resource

R, as in Resource

```
package pl.project13;

public final class R {
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int login=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
    }
}
```

R, as in Resource

Dodanie elementu id **wewnątrz widoku**, poprzez **+@id/**

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

R, as in Resource

Dodanie elementu id **wewnątrz widoku**, poprzez **+@id/**

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

Spowoduje wygenerowanie pola w klasie **R**:

```
public final class R {
  public static final class id {
    public static final int login = 0x7f050000;
  }
}
```

R, as in Resource

Dodanie elementu id **wewnątrz widoku**, poprzez **+@id/**

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

Spowoduje wygenerowanie pola w klasie **R**:

```
public final class R {
  public static final class id {
    public static final int login = 0x7f050000;
  }
}
```

A skorzystamy z niego w np. **Activity**:

```
EditText mLogin = findById(pl.project13.R.id.login);
```

R, tips and tricks

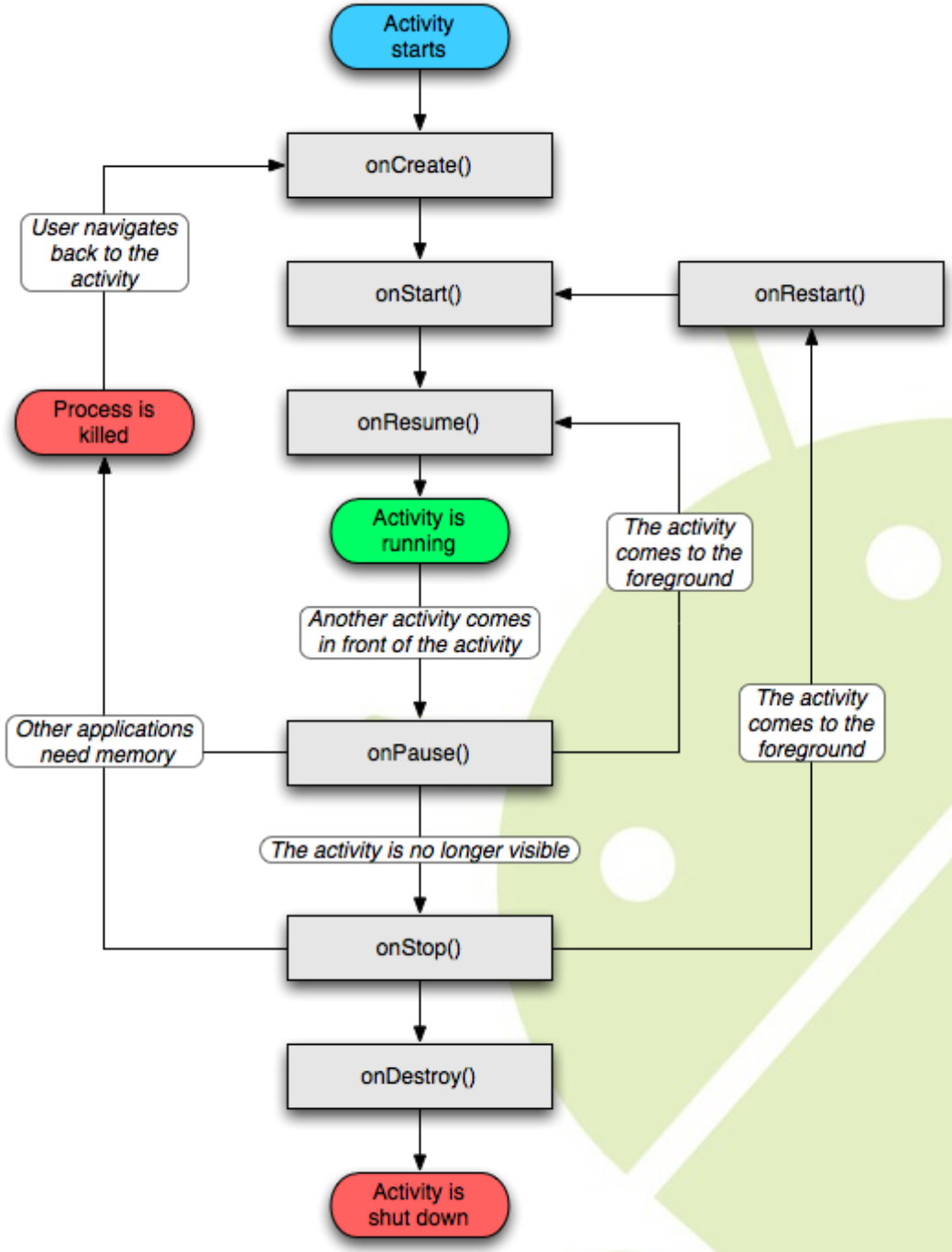
- nazwy wykorzystywane dla np. ID **nie muszą** być unikalne, @+id/login raz może oznaczać ten EditText a raz TextView. Rozwiązywane jest do wedle 'na czym wołany jest findById'.

R, tips and tricks

- nazwy wykorzystywane dla np. ID **nie muszą** być unikalne, @+id/login raz może oznaczać ten EditText a raz TextView. Rozwiązywane jest do wedle 'na czym wołany jest findById'.
- Korzystamy raczej z 'notacji_z_podkresleniami_tutaj'

R, tips and tricks

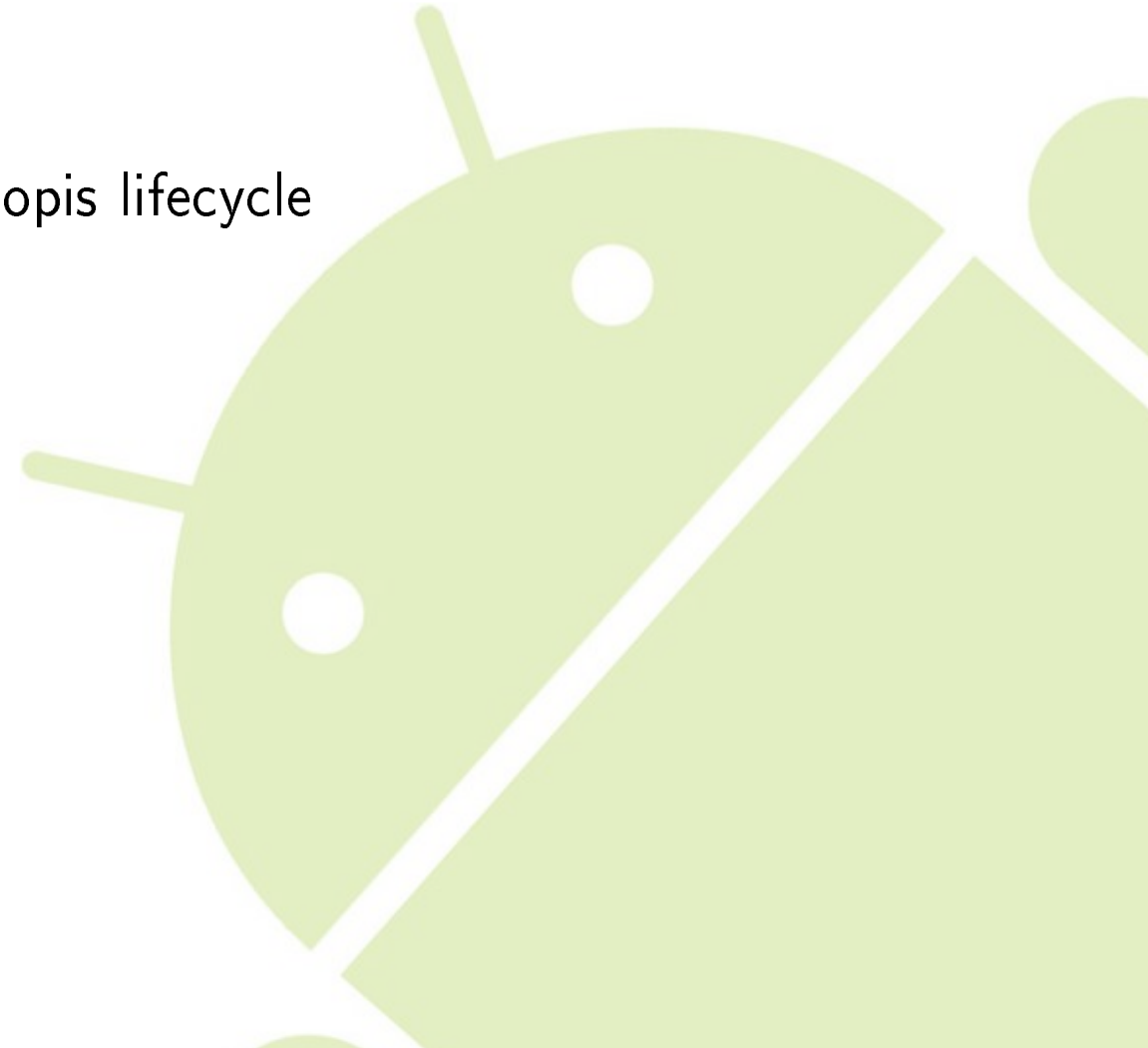
- nazwy wykorzystywane dla np. ID **nie muszą** być unikalne, @+id/login raz może oznaczać ten EditText a raz TextView. Rozwiązywane jest do wedle 'na czym wołany jest findById'.
- Korzystamy raczej z 'notacji_z_podkresleniami_tutaj'
- W kontekście gdzie nie masz **findById**, skorzystaj z **android.content.res.Resources.getSystem().get_____()**


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

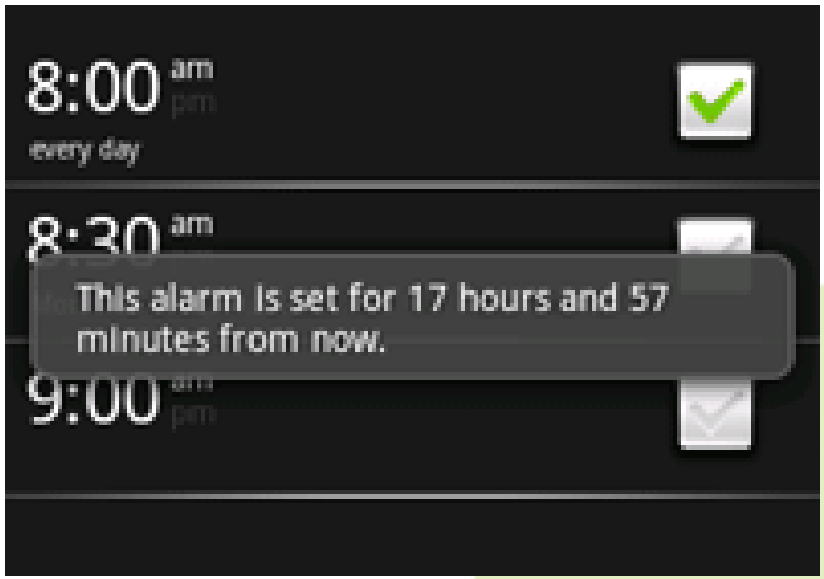
Activity

```
public class MainActivity extends Activity {
  public void onCreate(Bundle savedState) {
    /**/
  }
}
```

todo, informacje o service i opis lifecycle



Pyszne tosty z masełkiem (android.widget.Toast)



Przykład użycia:

```
Toast.makeText(getApplicationContext(),
               "Halo Szczecin!",
               Toast.LENGTH_LONG)
     .show();
```

Co więcej potrafi Toast?

```
Toast t = Toast.makeText(this, txt, LENGTH_SHORT);
```

Co więcej potrafi Toast?

```
Toast t = Toast.makeText(this, txt, LENGTH_SHORT);
```

Można mu zmienić pozycję:

```
t.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Co więcej potrafi Toast?

```
Toast t = Toast.makeText(this, txt, LENGTH_SHORT);
```

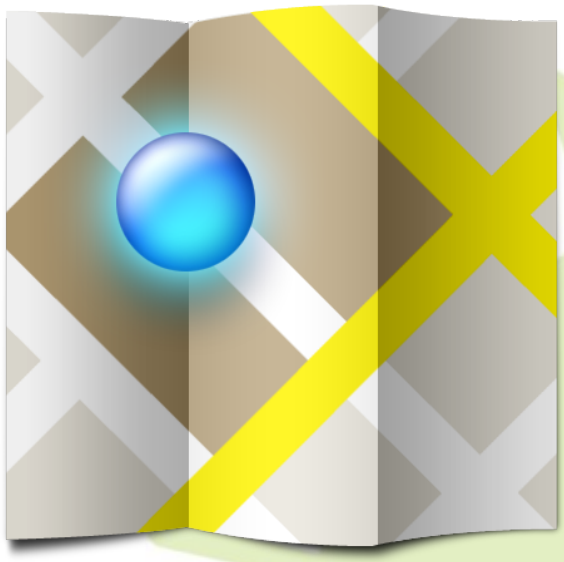
Można mu zmienić pozycję:

```
t.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

lub podmienić widok:

```
View customView = findViewById(R.id.custom_view);
/**/
t.setView(customView)
```

Google Maps



Istnieje pewien "problem" z Google Maps oraz niektórymi innymi API.

Nie są one dostępne bez odpowiedniego klucza oraz podpisania swojej aplikacji!

MapsAPI key sign-up

Rejestrujemy są po klucz na:
http://code.google.com/intl/pl-PL/android/maps-api-signup.html
BitLy: **http://bit.ly/mapsapiandroid**

Zdobywanie MD5 klucza 'debug'

```
keytool -list -alias androiddebugkey \
-keystore <path_to_debug_keystore>.keystore \
-storepass android -keypass android
```

Zdobywanie Md5 klucza 'release'

keytool -list -keystore /android.keystore

```
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry
android-key, Jul 3, 2011, PrivateKeyEntry,
Certificate fingerprint (MD5): AA:AA:AA:AA...
```

Oto co dostaniemy:

Twój klucz to:

```
0X8wqBZbEMsNFtsoHXzufZdWTKBlcUvTWq20fEg
```

Klucz jest przeznaczony dla wszystkich aplikacji podpisanych Twoim certyfikatem, którego „odciskiem pal

```
A2:D6:8F:12:A6:81:CA:C1:1B:8F:78:8A:E4:FB:A7:FC
```

Poniżej przedstawiono przykład układu xml, który ułatwi rozpoczęcie przygody z mapami:

```
<com.google.android.maps.MapView
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:apiKey="0X8wqBZbEMsNFtsoHXzufZdWTKBlcUvTWq20fEg"
   />
```

Więcej informacji znajduje się w dokumentacji interfejsu API.

Permissions

W tym przypadku interesują następujące <uses-permission/>:

- android.permission.ACCESS_COARSE_LOCATION
- android.permission.ACCESS_FINE_LOCATION

oraz (skoro chcemy wyświetlić mapkę)

- android.permission.INTERNET

Permissions

W tym przypadku interesują następujące <uses-permission/>:

- android.permission.ACCESS_COARSE_LOCATION
- android.permission.ACCESS_FINE_LOCATION

oraz (skoro chcemy wyświetlić mapkę)

- android.permission.INTERNET

Dodatkowo jeszcze deklarujemy wykorzystanie biblioteki maps:

```
<uses-library android:name="com.google.android.maps" />
```

Uwaga!

```
<application>
  <uses-library/>
</application>
<uses-permission/>
```

Co na pewno się przyda?

- **LocationManager**
- **MapView**
- bardzo wygodny jest **MapActivity**
- tip: dostępny jest **GPS** i **NETWORK** location provider

Zadanie: Google Maps App

- mapka, wycentrowana na obecnym położeniu telefonu
- podczas odświeżenia lokalizacji ma pojawiać się Toast z nową lokalizacją (oraz recentrujemy mapkę)
- obecne położenie ma być zaznaczone markerem: **http://bit.ly/gmapmark**
- dodatkowe: w przypadku oddalenia się od miejsca X (dowolne) należy odpalić '**alarm**'
- dodatkowe: zaskocz nas czymś!