

# KMI / TMA

## TVORBA MOBILNÍCH APLIKACÍ

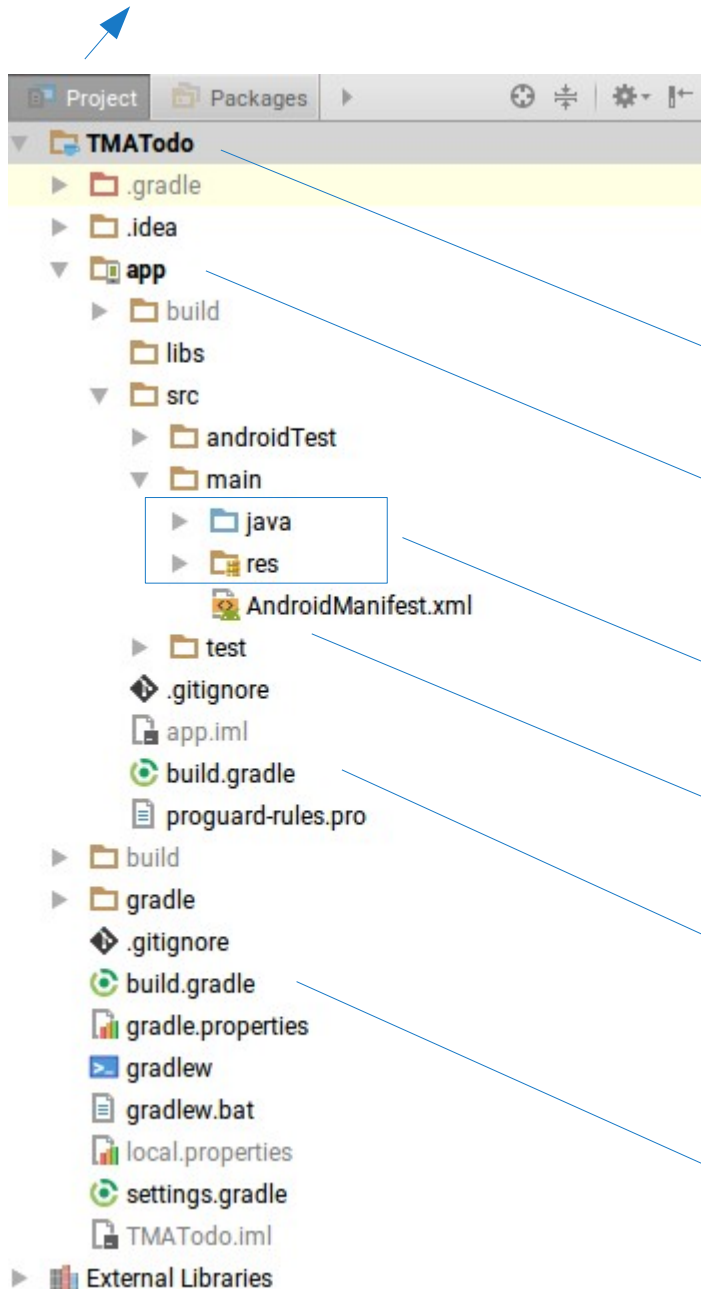
2. SEMINÁŘ | 5.10.2018  
ZS 2017/2018 | STŘEDA 13:15-15:45

# OBSAH SEMINÁŘE

KONFIGURAČNÍ SOUBORY PROJEKTU,  
AKTIVITY, ZÁKLADNÍ UDÁLOSTI,  
ŽIVOTNÍ CYKLUS APLIKACE,  
INTENTY A PRÁCE S NIMI

**NAINSTALOVÁNO?**  
**VYTVOŘENO ZAŘÍZENÍ?**  
**VYTVOŘEN PROJEKT?**  
**OTÁZKY?**

přepnout na „Project“



# PROJEKT

## KONFIGURAČNÍ SOUBORY

adresář projektu

adresář modulu

kód aplikace (to hlavní)

konfigurační soubor aplikace

konfigurační soubor gradle modulu

konfigurační soubor gradle projektu (top-level)

# GRADLE

## CO JE TO?

- » „nástroj pro automatizaci sestavování programu“
- » zdrojáky > APK pro distribuci
- » podobné nástroje: ant, make, rake, cake, ...?
- » napsaný v jazyce Groovy  
(konfigurační soubory v Groovy)

# PROJEKT? MODUL?

## CO TO JE?

- » **gradle modul**
  - » „část aplikace“
  - » vlastní knihovna
  - » konfigurace v samostatném build.gradle
  - » pro Android Wear: 2 moduly (phone+wear)
- » **gradle projekt**
  - » aplikace včetně všech modulů potřebných pro sestavení
  - » konfigurace v build.gradle se týká celého projektu

# BUILD.GRADLE PROJEKTU

## CO OBSAHUJE?

- » definice Android pluginu pro Gradle
  - » `com.android.tools.build:gradle:x.y.z`
- » globální proměnné, funkce (tasky), apod.
- » nemusí obsahovat nic

# BUILD.GRADLE MODULU

## CO OBSAHUJE?

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion "24.0.2"
    defaultConfig {
        applicationId "cz.upol.inf.kma.tmatodo"
        minSdkVersion 14
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:24.2.1'
    testCompile 'junit:junit:4.12'
}
```



# INTERMEZZO

## ANDROID API LEVELS

- » uživatelům jsou známy verze systému např. 6.0 Marshmallow, 7.0 Nougat, 8.0 Oreo, ...
- » pro vývojáře jsou důležitější čísla úrovně API, tj. verze rozhraní pro práci se systémem
- » 6.0 – API Level **23**, 7.0 – API Level **24**, 7.1 – API Level **25**, 8.0 – API Level **26**, ...

# MODULE BUILD.GRADLE

## CO OBSAHUJE?

- » **buildToolsVersion**
  - » verze utilit pro sestavení aplikace
  - » jaké verze jsou k dispozici viz SDK Manager
  - » doporučeno použít nejnovější
- » **compileSdkVersion**
  - » API úroveň oproti které se bude kompilovat
  - » na vyšších verzích možné využívat nové featury
  - » koresponduje s SDK Platform v SDK Manager
  - » doporučeno použít nejnovější

# MODULE BUILD.GRADLE

## CO OBSAHUJE?

- » `minSdkVersion`
  - » nejmenší možná API úroveň, na které půjde aplikace nainstalovat
  - » dnes (2017) se doporučuje API 15 (Android 4.0.3) nebo API 16 (Android 4.1)
- » `targetSdkVersion`
  - » ~ číslo API úrovně na které byla aplikace otestována
  - » systém na základě tohoto čísla uplatňuje různé změny kompatibility napříč verzemi, např. nový model oprávnění pouze pro  $\geq 23$

# MODULE BUILD.GRADLE

## CO OBSAHUJE?

- » `applicationId`
  - » unikátní identifikátor aplikace v rámci Google Play
  - » uživatel se s ním setká zpravidla pouze v URL odkazu na Google Play, vývojář mnohem častěji
  - » `applicationId`  $\neq$  `java package`, ale doporučuje se stejná konvence
- » `versionCode`
  - » celé číslo vyjadřující verzi, pro aktualizaci v zařízení musí být vždy stejná nebo vyšší
- » `versionName`
  - » řetězec, cokoliv, nemusí korespondovat s `versionCode`

# MODULE BUILD.GRADLE

## CO OBSAHUJE?

- » konfigurace testů, obfuskování kódu (proguard), podepsání balíčku pro upload na Google Play a další pokročilé možnosti Gradle Android pluginu
- » gradle úkoly (tasks), např. co provést po sestavení aplikace (upload na FTP?)
- » externí knihovny/závislosti
  - » podpůrné knihovny od Google
  - » knihovny třetích stran

# ANDROIDMANIFEST.XML

## CO OBSAHUJE?

```
<?xml version="1.0" encoding="utf-8"?>
<manifest package="cz.upol.inf.kma.tmatodo"
    xmlns:android="http://schemas.android.com/apk/res/android">

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

</manifest>
```

# ANDROIDMANIFEST.XML

## CO OBSAHUJE?

- » manifest
  - » package – java package
  - » application
    - » icon – ikona aplikace
    - » label – popis aplikace
    - » activity/service/...
      - » definice jednotlivých částí aplikace
    - » další pokročilá konfigurace

# ZÁKLADNÍ KAMENY

## JAVA TŘÍDY

- » a) Java třídy dědící z tříd v Android SDK, např. **Activity**, **Service**, **Button**, **SQLiteOpenHelper**, ...
- » b) ostatní Java třídy, např. POJO, model, utility třídy, ...



# ZÁKLADNÍ KAMENY

## AKTIVITY

- » aktivita ~ „obrazovka“
- » aktivita se vytváří zděděním třídy `android.app.Activity`
- » aplikace jsou zpravidla tvořeny několika aktivitami
- » počet obrazovek = počet aktivit (zjednodušeně)

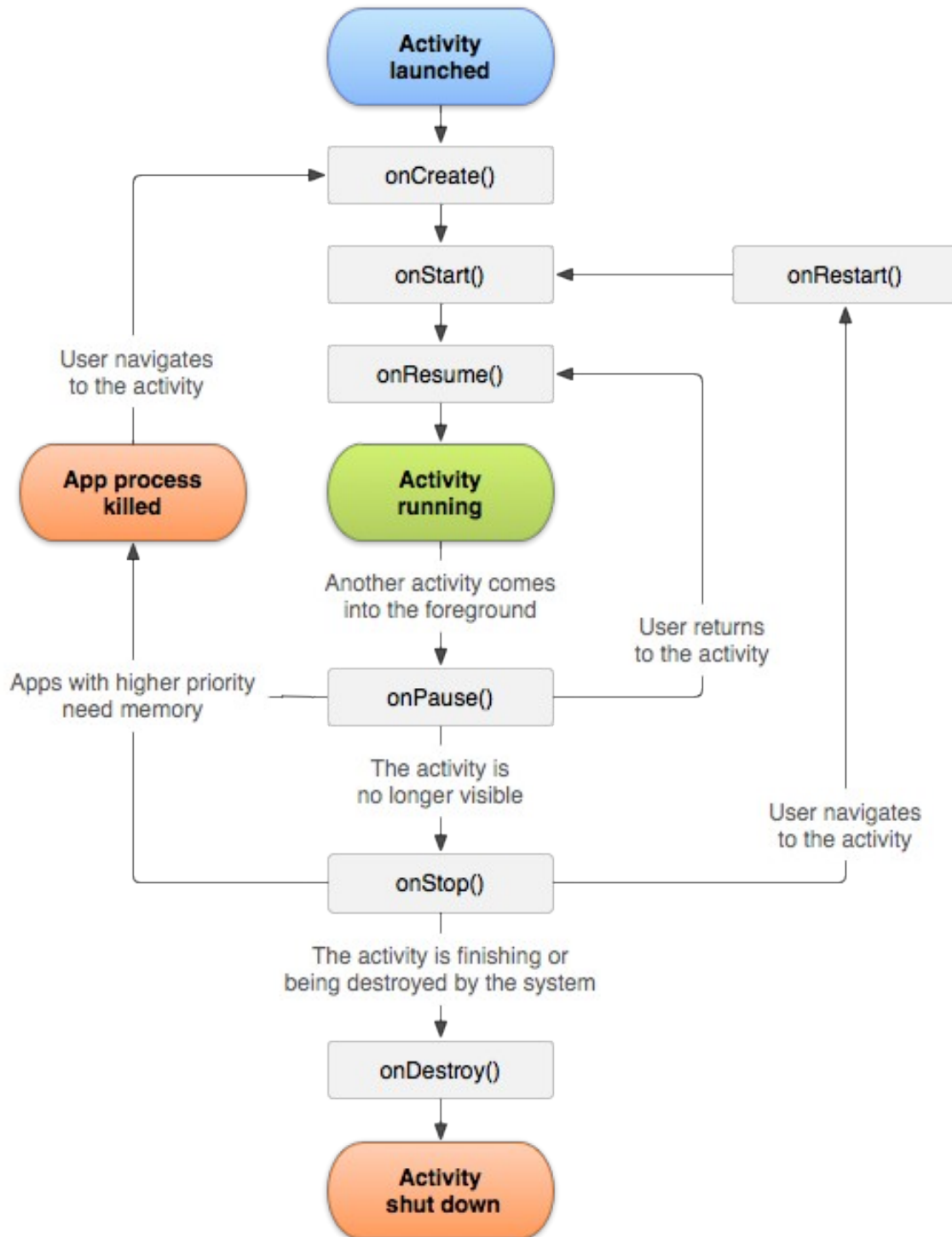
# ŽIVOTNÍ CYKLUS APLIKACE

CO ZNAMENÁ, ŽE JE APLIKACE „SPUŠTĚNÁ“?

- » je spuštěná alespoň jedna její aktivita nebo služba na pozadí
- » nejsme na desktopu, máme omezené systémové prostředky, systém může aplikaci, resp. aktivity kdykoliv zabít
- » musíme na to dávat pozor  $\Rightarrow$  eventy

# AKTIVITA

## ŽIVOTNÍ CYKLUS



- » created
- » started
- » resumed
- » paused
- » stopped
- » destroyed

# AKTIVITA

SRC/MAIN/JAVA/{PACKAGE}/MAINACTIVITY.JAVA

```
package cz.upol.inf.kma.tmatodo;
```

```
import ...
```

```
public class MainActivity extends AppCompatActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

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

```
    }
```

```
}
```

rozšířená Activity pro zpětnou  
kompatibilitu

metoda spuštěná  
vždy při startu  
aktivity

metoda pro nastavení  
uživatelského rozhraní

# UŽIVATELSKÉ ROZHRAŇÍ

SRC/MAIN/RES/LAYOUT/ACTIVITY\_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/activity_main"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="cz.upol.inf.kma.tmatodo.MainActivity">
```

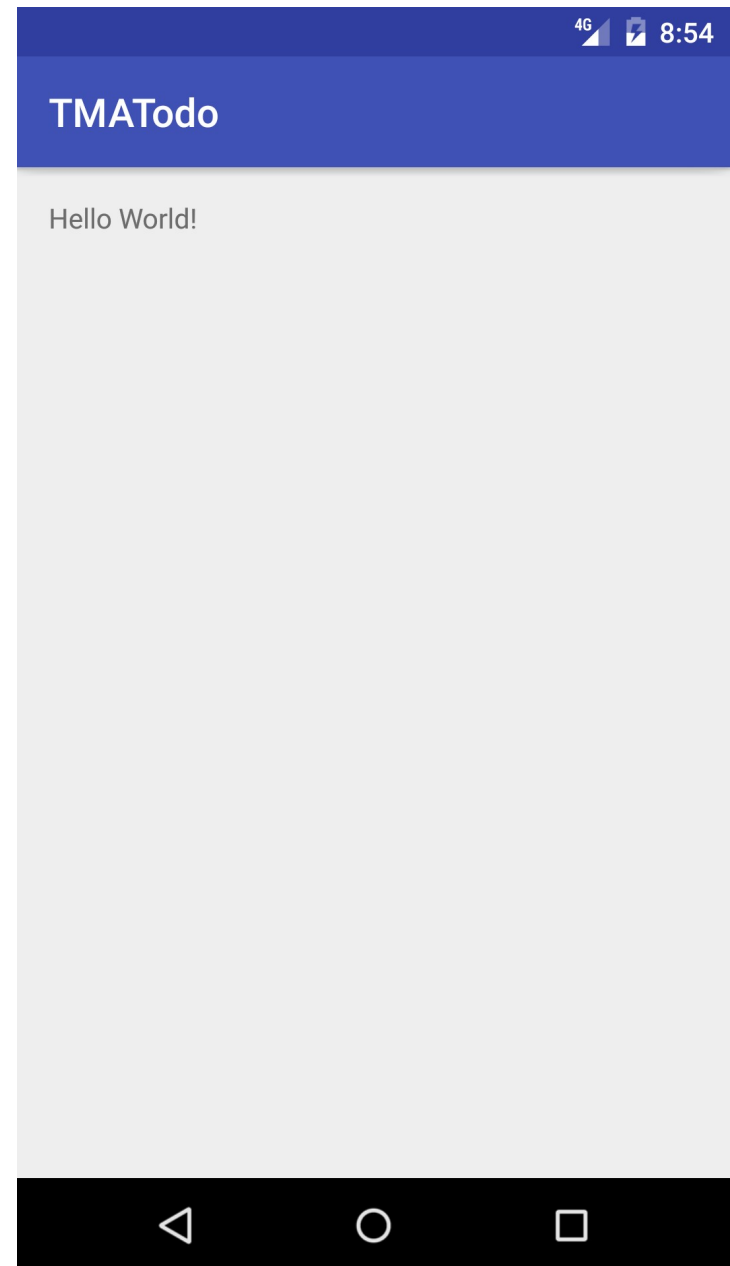
kontejner  
sdužující  
více prvků

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"/>
```

jednoduché  
textové  
pole

```
</RelativeLayout>
```

**SPUŠŤTME TO**  
ZATÍM MÁME TOHLE



# VÍCE OBRAZOVEK

## VYTVOŘME DALŠÍ

- » Zkopírovat `res/layout/activity_main.xml` a vytvořit soubor `res/layout/activity_second.xml` s identickým obsahem
- » File > New > Java Class
  - » Name: **SecondActivity**
  - » Superclass: **AppCompatActivity**
- » V **SecondActivity** načíst nový layout

# VÍCE OBRAZOVEK

## PŘIDÁNÍ TLAČÍTKA V ACTIVITY\_MAIN.XML

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Hello World!"/>
```



```
<Button  
    android:id="@+id/button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Hello World!"/>
```



# VÍCE OBRAZOVEK

## PŘIDÁNÍ INTERAKCE V MAINACTIVITY.JAVA

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Button button = (Button) findViewById(R.id.button);

    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Log.d("TODO", "Clicked on button!");
        }
    });
}
```

získání instance tlačítka dle identifikátoru v layoutu

nastavení akce po kliknutí na tlačítko

# VÍCE OBRAZOVEK

## SPUŠTĚNÍ NOVÉ AKTIVITY

- » Swing
  - » `new JFrame(...);`
- » Android
  - » `new SecondActivity(...);`



# VÍCE OBRAZOVEK

## SPUŠTĚNÍ NOVÉ AKTIVITY

» Swing

```
new JFrame(...);
```

» Android

```
new SecondActivity(...);
```

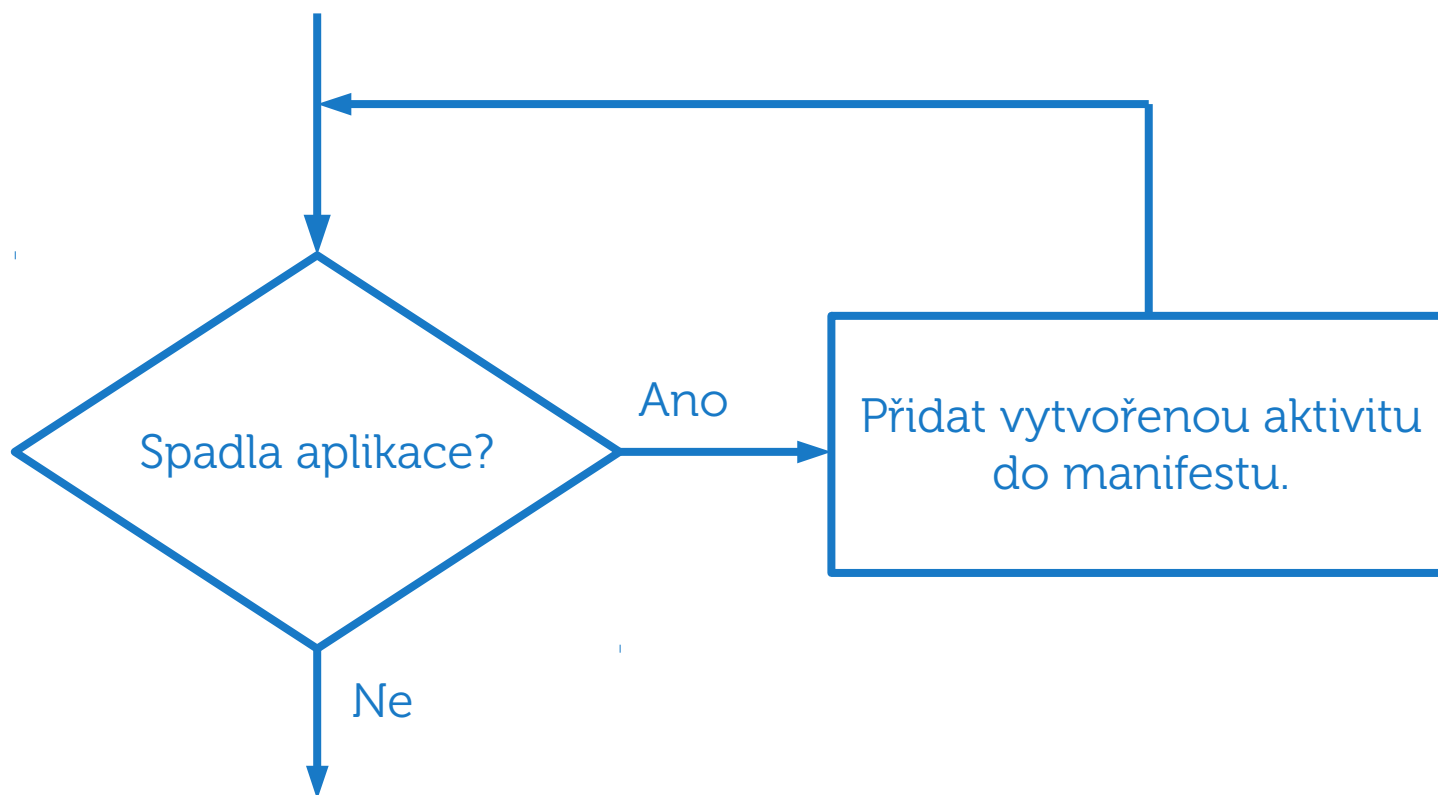
```
Intent i = new Intent(
```

```
    this, SecondActivity.class);
```

```
startActivity(i);
```

# VÍCE OBRAZOVEK

## DEFINICE NOVÉ OBRAZOVKY V MANIFESTU



# VÍCE OBRAZOVEK

## DEFINICE NOVÉ OBRAZOVKY V MANIFESTU

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="TMATodo"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>

    <activity android:name=".SecondActivity" />

</application>
```

# ÚKOL 2. SEMINÁŘE

## LOGOVÁNÍ ŽIVOTNÍCH CYKLŮ AKTIVIT

- 1) Zprovoznit prostředí pro vývoj.
- 2) Vytvořit projekt TODO aplikace.
- 3) Vytvořit druhou aktivitu.
- 4) Vypsát do logu základní metody životního cyklu při přepínání aktivit (6 metod z diagramu) a zhodnotit, zda to souhlasí s diagramem.
- 5) Ukázat do konce semináře nebo na následujícím semináři.

# OTÁZKY

PTEJTE SE!

