Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Elektronsko in mobilno poslovanje

VAJE 3

Jan Meznarič
Simon Vrhovec

Ljubljana
9. 11. 2016

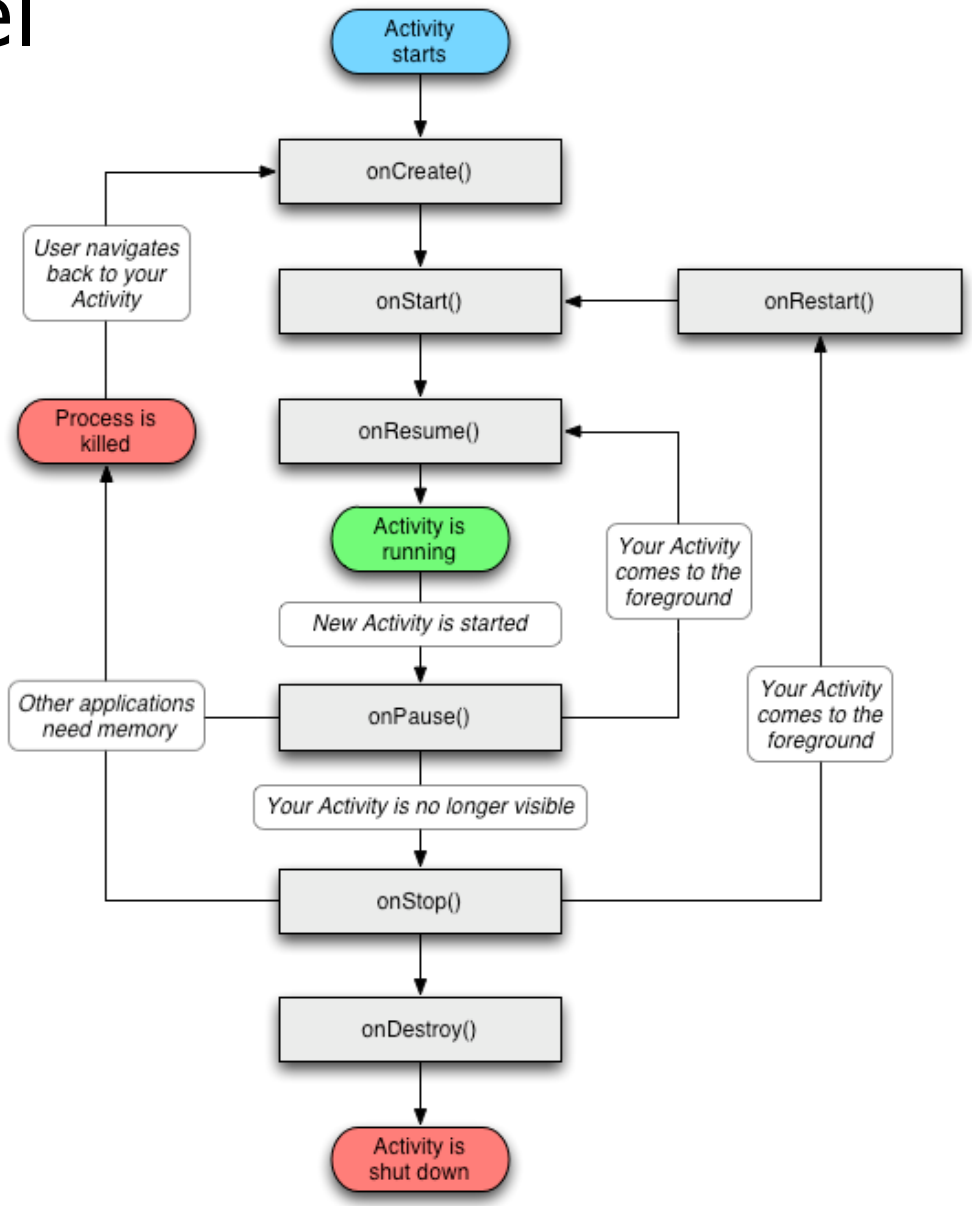
Tipična zgradba aplikacij v Androidu

- **Activity** – osnovna izvedbena enota, ki navadno vključuje tudi UI; aktivnosti se lahko kličejo med seboj.

- **Intent** – skrbi za komunikacijo med aktivnostmi – abstrakten opis funkcije, ki jo neka aktivnost zahteva od druge aktivnosti.

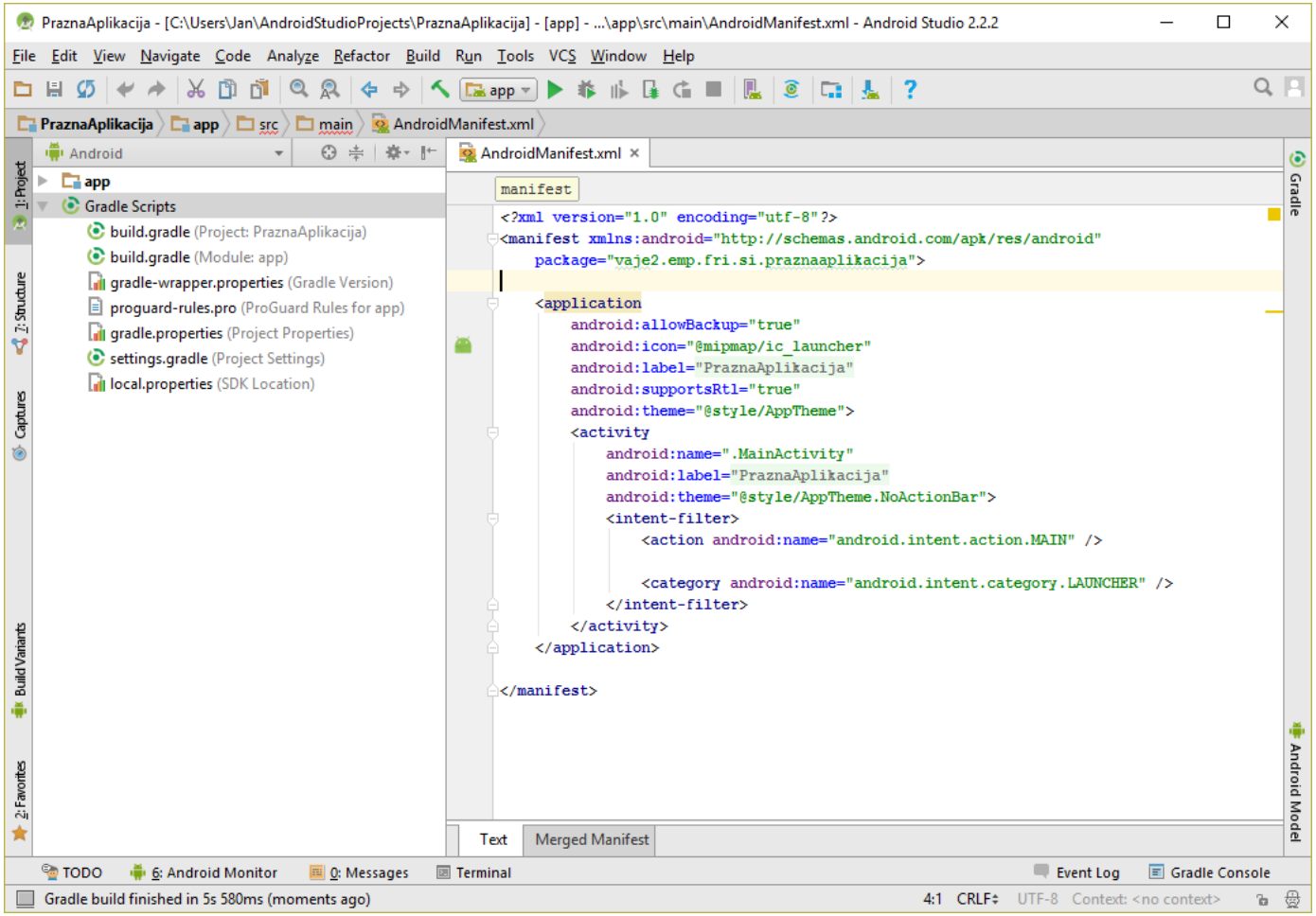
Življenjski cikel aktivnosti

V življenjskem ciklu aktivnosti se prožijo različne *call-back* metode.



Aktivnosti

- Vse aktivnosti moramo obvezno prijaviti v *AndroidManifest.xml.*



Vaja 3

- Dodajte novo aktivnost *(HelpActivity).*
- Preverite manifest; če aktivnosti nismo izdelali s čarovnikom, jo moramo ročno dodati v manifest.
- Na prvo aktivnost dodajte tipko *(@+id/btnHelp)* s katero boste poklicali drugo aktivnost.
- Izdelajte ustrezen *OnClickListener* in vanj zapišite:

```
Intent intent = new Intent(MainActivity.this,
                           HelpActivity.class);
startActivity(intent);
```

Klici aktivnosti

- Z razredom *Intent* zgradimo klic.
- V klic lahko dodamo parametre, preko katerih se aktivnosti sporazumevata.
- Klic lahko pričakuje tudi povratne podatke.

Klici aktivnosti

- Enostaven klic brez prenosa podatkov:


```
Intent intent = new Intent(MainActivity.this,
                           HelpActivity.class);
startActivity(intent);
```

Klici aktivnosti

- Enostaven klic s prenosom vsebine:


```
Intent intent = new Intent(MainActivity.this,
                           HelpActivity.class);
Bundle bundle = new Bundle();
bundle.putString("emp.vaja3.poslanString", "Moj
                                          string");
intent.putExtras(bundle);
startActivity(intent);
```

Z uporabo *Bundle* lahko prenašamo vse primitivne tipe, objekte pa s pomočjo serializacije.

Poimenovanje vsebine naj se začne z imenom paketa!

Klici aktivnosti

- Branje podatkov v klicani aktivnosti: 

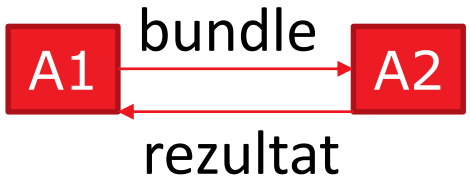

```
String myString = "";

final Intent intent = getIntent();

if (intent != null) {
    final Bundle bundle = intent.getExtras();

    if (bundle != null) {
        myString = bundle.getString("emp.vaja3.poslanString");

        if (myString == null) {
            myString = "Privzet niz";
        }
    }
}
```

Vaja 4

- Na prvo aktivnost dodajte še eno tipko.

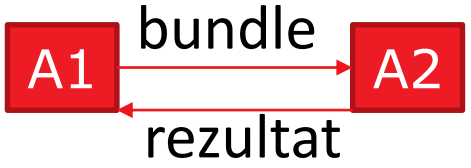
- Ta tipka naj pokliče drugo aktivnost in pri tem prenese niz, ki naj se na drugi aktivnosti izpiše.

Klici aktivnosti



- Klic s povratnim rezultatom:
  1. A1: Kličemo *startActivityForResult* (zraven pošljemo *requestCode*, lahko pa tudi bundle).
  2. A2: Prestrežemo *bundle (extras)*.
  3. A2: Pred izhodom iz aktivnosti kreiramo *Intent* v katerem je *Bundle* s povratnim rezultatom in kličemo *setResult.*
  4. A1: rezultat prestrežemo v *onActivityResult* (dogodek na aktivnosti A1):
     - Najprej preverimo *requestCode*.
     - Nato preverimo, če je rezultat ustrezen (RESULT_OK, RESULT_CANCELED).

Klici aktivnosti



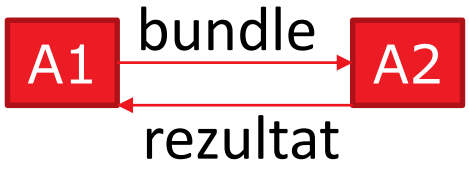
- 1 (A1):

```
Intent intent = new Intent(MainActivity.this, HelpActivity.class);
Bundle bundle = new Bundle();
bundle.putString("emp.vaja3.poslanString", "Moj string");
intent.putExtras(bundle);
startActivityForResult(intent, 1234);
```

- 2 (A2):

```
String myString = "";
final Intent intent = getIntent();

if (intent != null) {
    final Bundle bundle = intent.getExtras();
    if (bundle != null) {
        myString = bundle.getString("emp.vaja3.poslanString");
        if (myString == null) {
            myString = "Privzeta vrednost";
        }
    }
}
```

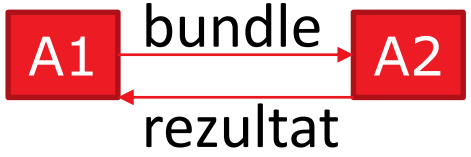
Klici aktivnosti



- 3 (A2):

```
Intent returnIntent = new Intent();
Bundle bundle = new Bundle();
bundle.putString("emp.vaja3.vrnjenString", "Ta
                                    string vracamo");
returnIntent.putExtras(bundle);
setResult(RESULT_OK, returnIntent);
finish();
```

Klici aktivnosti



- 4 (A1):

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == 1234) {
        if (resultCode == RESULT_OK) {
            final String result = data.getStringExtra("emp.vaja3.vrnjenString");
            Log.i(TAG, "Prejet rezultat: " + result);
        } else {
            Log.w(TAG, "Neuspesen rezultat.");
        }
    }
}
```

Vaja 5

- Izdelajte klic aktivnosti s povratnim rezultatom.

Ohranjanje stanja ob uničenju in ponovnem zagonu aplikacije

Za ohranjanje stanja imamo na razpolago 2 namenska dogodka:

- **onSaveInstanceState(Bundle)**: Android pokliče to metodo za shranjevanje stanja aktivnosti preden je aktivnost uničena.

- **onRestoreInstanceState(Bundle)**: Metoda, ki je poklicana za metodo onStart(), ko je aktivnost ponovno poklicana iz stanja, ki je bilo predhodno shranjeno z metodo onSaveInstanceState(Bundle).

Ohranjanje stanja ob uničenju in ponovnem zagonu aplikacije

- Stanje ohranjamo s shranjevanjem/branjem iz razreda *Bundle*, ki je parameter metod:
  - onCreate(Bundle),
  - onSaveInstanceState(Bundle) in
  - onRestoreInstanceState(Bundle).

Ohranjanje stanja ob uničenju in ponovnem zagonu aplikacije

Primer:

```
@Override
public void onRestoreInstanceState(Bundle
savedState)
{
    super.onRestoreInstanceState(savedState);
    if (i==0) i=savedState.getInt("ii");
}

@Override
public void onSaveInstanceState(Bundle outState) {
     super.onSaveInstanceState(outState);
     outState.putInt("ii", i);
}
```

Ohranjanje stanja ob uničenju in ponovnem zagonu aplikacije

- *OnCreate* se bo ob ponovnem zagonu sprožil tako kot ob startu. V primeru, da določenih stvari ne želimo ponovno zagnati lahko preverimo je *Bundle* enak *null*.

```
if (savedInstanceState==null)
{
    //koda, ki se izvede samo ob prvem zagonu
}
```