



## MOBILNE APLIKACIJE

### Vežbe 2

## Sadržaj

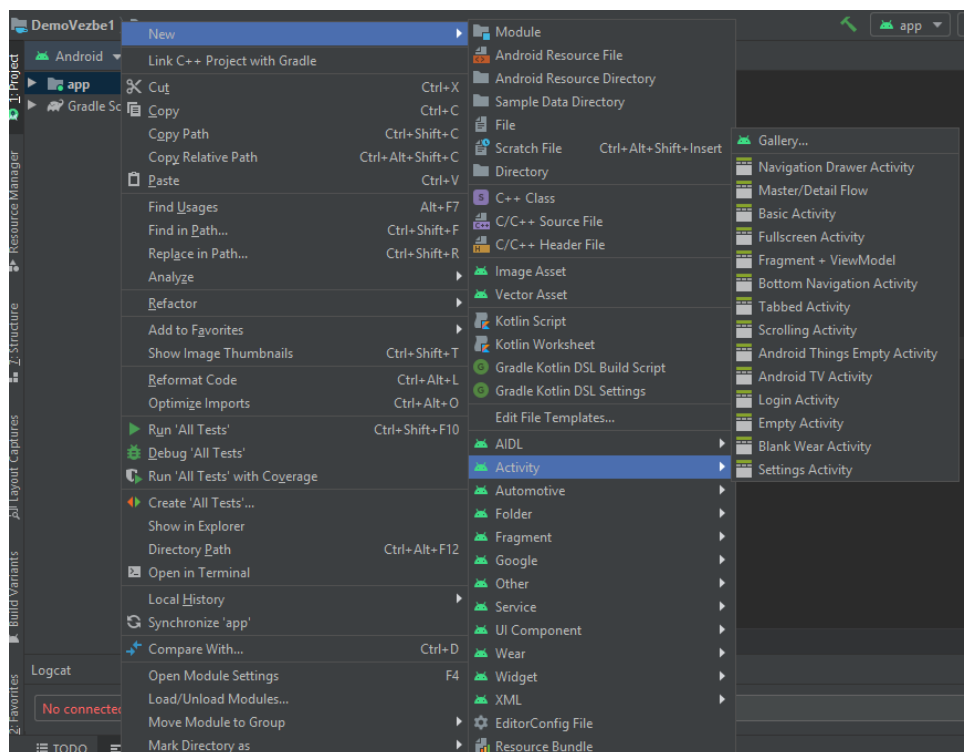
1. Aktivnosti .....	3
1.1 Kreiranje aktivnosti .....	3
1.2 Životni ciklus aktivnosti .....	6
1.3 Namere .....	10
2. Fragmenti .....	12
2.1 Kreiranje fragmenta .....	12
2.2 Životni ciklus fragmenta .....	14
3. Domaći .....	16

# 1. Aktivnosti

Aktivnosti predstavljaju osnovne gradivne blokove Android aplikacija. Aktivnost je pojedinačan ekran Android aplikacije.

## 1.1 Kreiranje aktivnosti

Nova aktivnost se kreira odabirom stavke iz menija: File > New > Activity (slika 1). U primeru, koji sledi, odabrali smo *Empty Activity*.



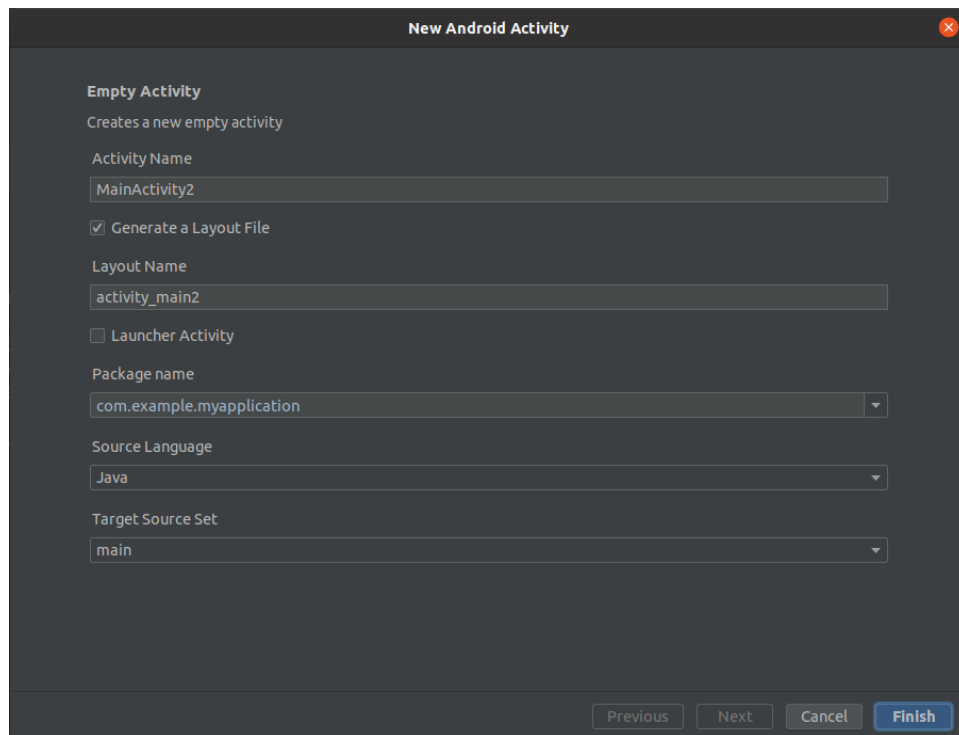
Slika 1. Kreiranje nove aktivnosti

Nakon što se odabere vrsta aktivnosti prikazuje se forma *New Android Activity*, koju treba da popunite sa osnovnim podacima same aktivnosti:

- **Activity Name**  
U ovo polje unosite naziv aktivnosti (Java klase).
- **Layout Name**  
U ovo polje unosite naziv izgleda ekrana (xml datoteke).
- **Package Name**  
*Package Name* je naziv paketa u kom će se nalaziti kreirana aktivnost.
- **Select Language**  
Za jezik biramo *Java*.
- **Target Source Set**

U ovom polju ostaje main, koji je podrazumevano kreiran kada je kreiran i projekat.

Klikom na dugme *Finish* kreira se nova aktivnost.



Slika 2. *Configure Activity*

Nakon što smo uspešno kreirali novu aktivnost, prvo što primećujemo jeste da se u paketu, koji je odabran prilikom kreiranja aktivnosti, nalazi nova klasa *SecondActivity*. Ova klasa nasledjuje *AppCompatActivity* klasu i na taj način dobijamo metode životnog ciklusa aktivnosti (više o ovim metodama u sledećem potpoglavlju).

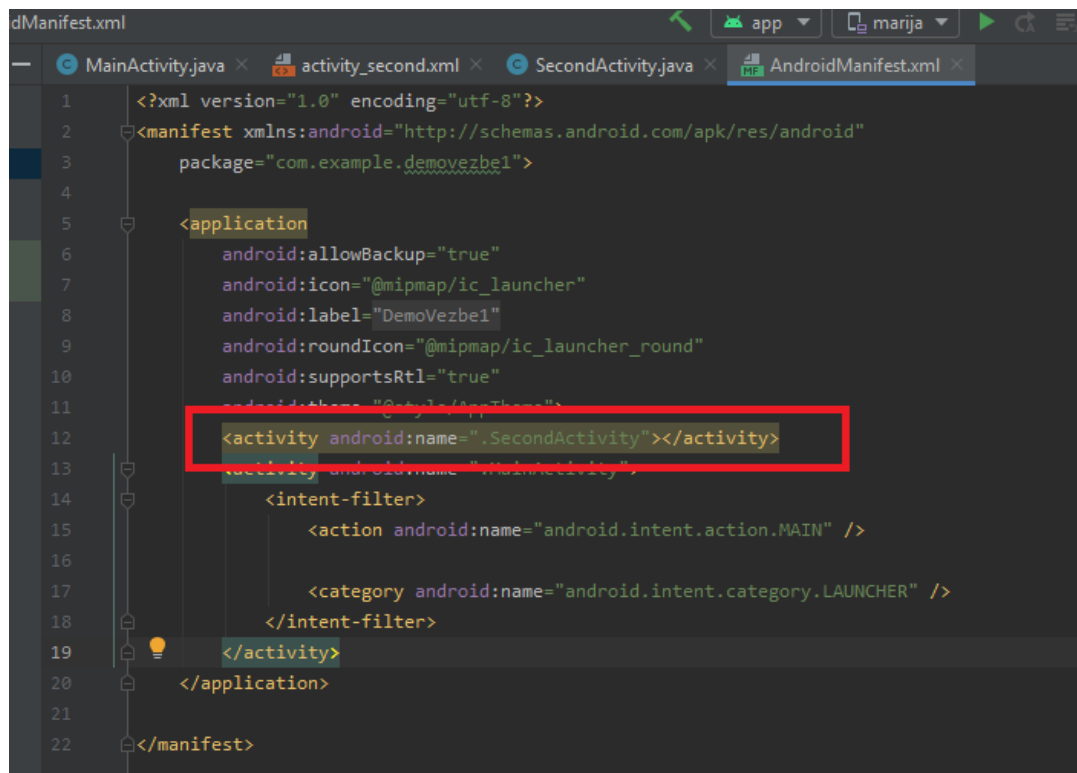
*setContentView* metoda vrši povezivanje aktivnosti sa njenim izgledom (*layout*) koji se nalazi na slici 5.

```
2
3
21
22 import ...
23
24 public class SecondActivity extends AppCompatActivity {
25
26     @Override
27     protected void onCreate(@Nullable Bundle savedInstanceState) {
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_second);
30     }
31 }
```

Slika 3. Nova aktivnost *SecondActivity*

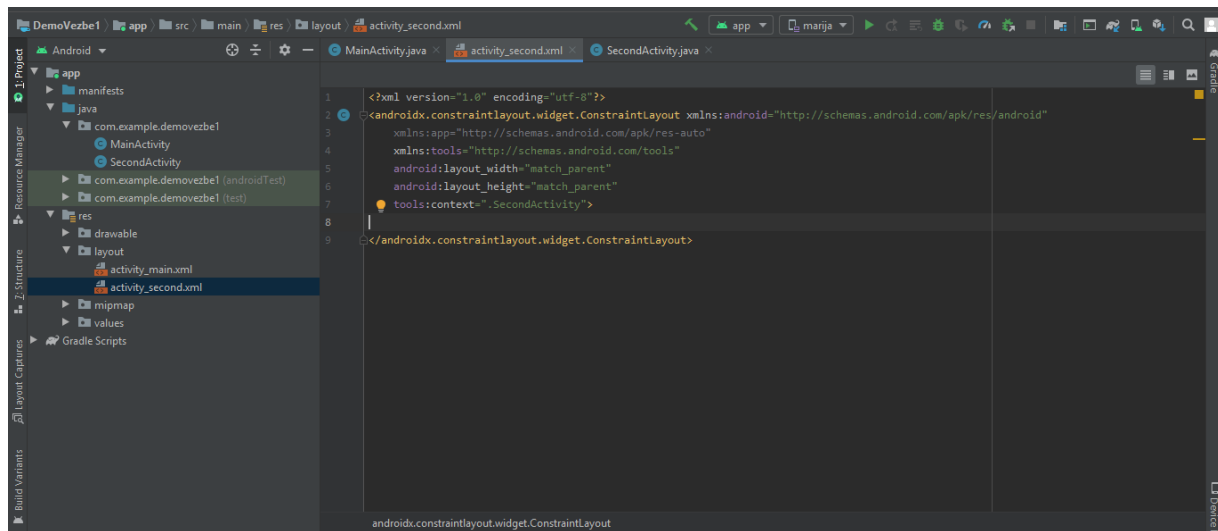
Aktivnosti se deklarišu u XML datoteci *AndroidManifest*, koja se nalazi unutar *manifests* direktorijuma. U *AndroidManifest* datoteku dodaje se element `<activity>`. Ovaj element minimalno treba da ima atribut *android:name*, čija vrednost treba da bude putanja do klase koja nasleđuje neku od *Activity* klasa. Kada otvorimo *AndroidManifest* datoteku primećujemo da je AS već dodao ovaj element za nas (slika 4), ali to neće biti slučaj ako sami ručno kreirate novu klasu za aktivnost.

Na slici 4 se ispod nove aktivnosti nalazi i aktivnost koja je inicijalno kreirana prilikom pravljenja projekta. Unutar nje se nalazi element `<intent-filter>` kojim označavamo da je ta aktivnost glavna (*main*) aktivnost za našu aplikaciju. Samo jedna aktivnost treba da bude glavna aktivnost u okviru naše aplikacije, poput *main* metode u bilo kom programskom jeziku.



Slika 4. *AndroidManifest* datoteka

Osim *Java* klase, za nas je kreirana i xml datoteka koja se nalazi unutar direktorijuma *res/layout*. Našu aktivnost smo uz pomoć *setContentView* metode (slika 3) povezali sa ovom datotekom, tj. sa njenim izgledom.



Slika 5. *activity\_second.xml*

## 1.2 Životni ciklus aktivnosti

Aktivnost može da bude:

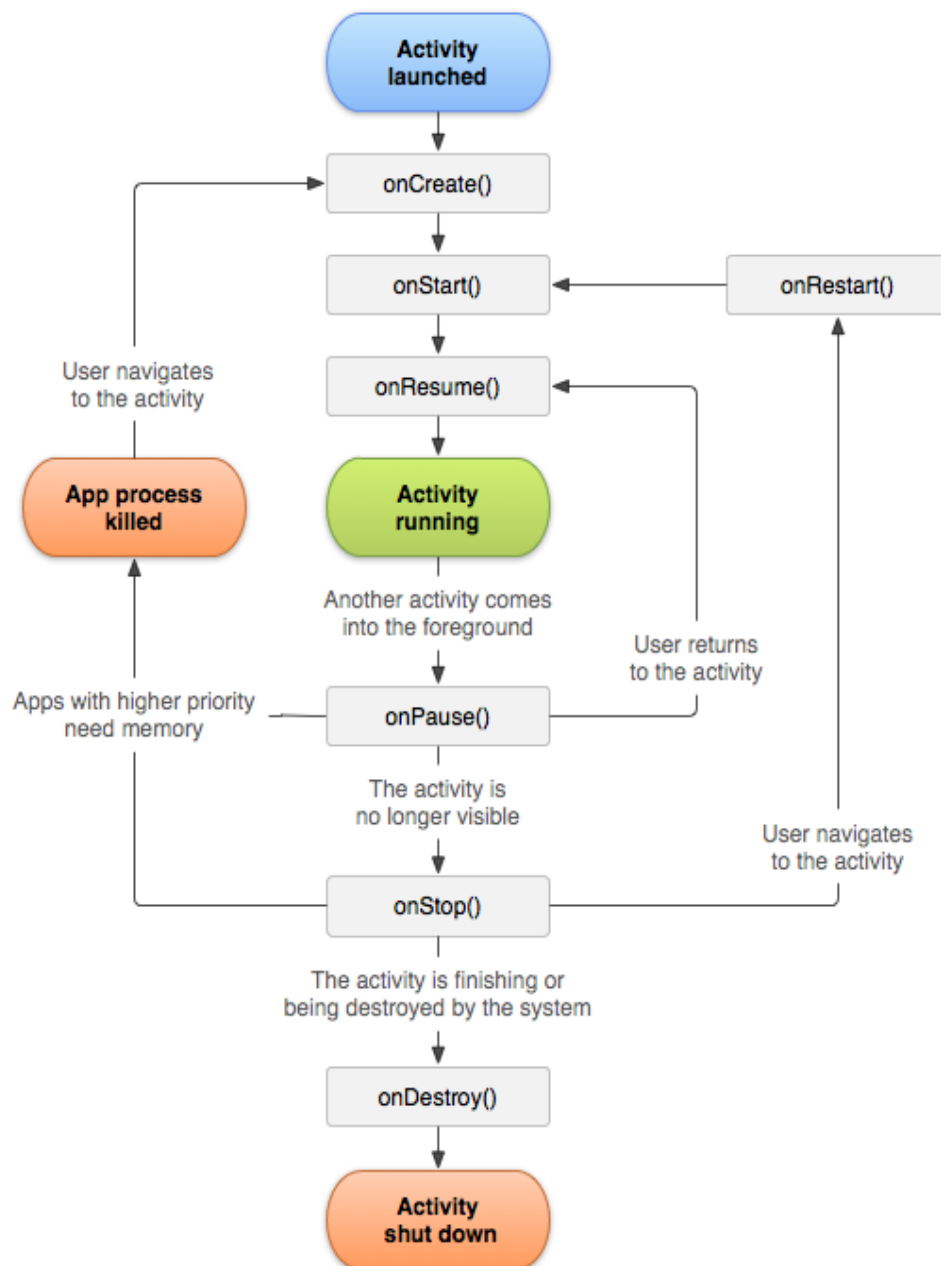
1. Aktivna
2. Pauzirana ili
3. Zaustavljena.

**Aktivna aktivnost** – aktivnost je u prvom planu i ima fokus.

**Pauzirana aktivnost** – aktivnost je pauzirana kada se neka druga aktivnost nalazi u prvom planu i ima fokus, ali ta pauzirana aktivnost je još uvek vidljiva, jer se na primer ispred nje nalazi aktivnost koja ne pokriva ceo ekran.

**Zaustavljena aktivnost** – aktivnost koja se nalazi u pozadini, ali je potpuno prekrivena nekom drugom aktivnošću.

Na slici 6 se nalazi životni ciklus aktivnosti.



Slika 6. Životni ciklus aktivnosti

## onCreate

Sistem poziva onCreate metodu kada startuje aktivnost. Ovde se zauzimaju resursi i najosnovnije operacije da bi aktivnost mogla da se izvrši.

Unutar onCreate metode, pozivanjem setContentView metode iscrtavamo korisnički interfejs i to je mesto gde spajamo layout sa aktivnošću (slika 7).

U ovoj metodi ne sme da se nalazi kod koji će blokirati prelazak aktivnosti u naredne metode, tj. sve operacije koje mogu dosta vremena da oduzmu ne treba pisati u ovoj metodi

```
2
3 import ...
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14 }
15
```

Slika 7. Primer *onCreate* metode

## onStart

Sistem će pozvati *onStart* metodu neposredno pre nego što aktivnost bude vidljiva korisniku (slika 8).

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

Slika 8. Primer *onStart* metode

## onResume

Metoda *onResume* se poziva neposredno pre nego što aktivnost počne interakciju sa korisnikom. Aktivnost je ovde vidljiva i nalazi se u fokusu (slika 9). Na ovom mestu je zgodno raditi neke duže aktivnosti, kao što su pristupi ka bazi, internetu itd.

```
@Override
protected void onResume() {
    super.onResume();
    Toast.makeText( context: this, text: "onResume()", Toast.LENGTH_SHORT).show();
}
```

Slika 9. Primer *onResume* metode

## onPause

Sistem će pozvati ovu metodu neposredno pre nego što pauzira izvršavanje aktivnosti. Ova metode se obično koristi za snimanje podataka i zaustavljanje svih procesa, koji zauzimaju procesor (slika 10).



```

@Override
protected void onPause() {
    super.onPause();
    Toast.makeText( context: this, text: "onPause()", Toast.LENGTH_SHORT).show();
}

```

Slika 10. Primer *onPause* metode

Aktivnost je sve vreme aktivna kada je interfejs dostupan za korisnika i to traje od metode *onResume* do metode *onPause*. Ako je druga aktivnost prešla u prvi plan, ali ne remeti prethodnu aktivnost, onda će ta prethodna aktivnost ostati u stanju *paused*, dok se nova aktivnost ne završi. Potom će se pozvati *onResume* metoda, koja će staru aktivnost ponovo vratiti u prvi plan i nastaviće se njeno izvršavanje.

## onStop

Ova metoda se poziva kada aktivnost više nije vidljiva korisniku (slika 11).

```

@Override
protected void onStop() {
    super.onStop();
    Toast.makeText( context: this, text: "onResume()", Toast.LENGTH_SHORT).show();
}

```

Slika 11. Primer *onStop* metode

Kada ostane malo prostora u memoriji OS će ukloniti aktivnosti iz memorije koje se nalaze u stanju *paused* ili *stopped*.

## onDestroy

*onDestroy* je metoda koja se poziva pre nego što se aktivnost uništi. Ovde više aktivnost ne postoji i svi resursi su oslobođeni.

Kada želite da isključite aktivnost to možete da uradite tako što pozovete metodu *finish()*, koja će pozvati metodu *onDestroy()*.

Ako treba da proverite da li je metoda zaustavljena ili samo pauzirana to možete proveriti pozivanjem metode *isFinishing*.

## onRestart

Kada je naša aktivnost u potpunosti pokrivena, ona prvo ide u stanje *onPause*, pa potom u stanje *onStop*. Ako želimo da se ponovo vratimo na tu aktivnost to radimo uz pomoć metode *onRestart*.

## onSaveInstanceState

Svaki put kada nešto preklopi našu aktivnost, njeno stanje treba da bude sačuvano, da bismo mogli ponovo da se vratimo u to stanje. To stanje se ponovo inicijalizuje u *onCreate* metodi ili u metodi *onRestoreInstanceState*.

Jasan primer je kada ukucamo neki tekst u *input* polje i potom izađemo i vratimo se nazad, to *input* polje i dalje prikazuje tekst koji smo prethodno uneli.

## 1.3 Namere

Namera (*Intent*) služi za povezivanje komponenti aplikacije. *Intent* je glavna klasa unutar Android-a za prelazak na druge delove aplikacije. Postoje 2 vrste namere:

1. Eksplicitna
2. Implicitna

**1. Eksplicitna** – eksplicitno navodimo sa koje aktivnosti na koju prelazimo. Primer: sa *MainActivity* prelazimo na *SecondActivity*. Pozivom *startActivity* metode, šaljemo poruku Android-u da on za nas pokrene drugu aktivnost, nakon čega se i sam korisnik prebacuje na novu aktivnost (slika 12).

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toast.makeText(context, this, "onCreate()", Toast.LENGTH_SHORT).show();

    /*...*/
    Button button = (Button) findViewById(R.id.button);
    button.setOnClickListener((v) -> {
        /*...*/
        Intent intent = new Intent(packageContext, MainActivity.this, SecondActivity.class);

        /*...*/
        startActivity(intent);
    });
}
```

Slika 12. Primer eksplicitne namere

**2. Implicitna** – ne navodimo eksplicitno na koju aktivnost želimo da pređemo, nego specificiramo parametre na osnovu kojih nam Android nudi sve moguće opcije, od kojih mi biramo onu koja je najbolja. Primer: Želimo da otvorimo <http://www.google.com>. Korisnik ima više instaliranih browser-a na uređaju. Android će nam ponuditi sve browser-e koji mogu da završe tu akciju. Ako imamo samo 1 browser instaliran on će biti pokrenut (slika 13).

```
Button open_google = (Button) findViewById(R.id.open_google);
open_google.setOnClickListener((v) -> {
    /*...*/
    Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
    startActivity(i);
});
```

Slika 13. Primer implicitne namere

Kada prelazimo sa jedne na drugu aktivnost, možemo da pošaljemo i neke podatke u tu aktivnost, ako postoji potreba za tim (slika 14).

```
@Override
public void onItemClick(ListView l, View v, int position, long id) {
    super.onItemClick(l, v, position, id);

    Cinema cinema = Mokap.getCinemas().get(position);

    /*...*/
    Intent intent = new Intent(getActivity(), DetailActivity.class);
    intent.putExtra( name: "name", cinema.getName());
    intent.putExtra( name: "descr", cinema.getDescription());
    startActivity(intent);
}
```

Slika 14. Primer slanja podataka u aktivnost

Na slici 15 se nalazi kod koji prikazuje kako se podaci dobavljaju iz aktivnosti.

```
public class DetailActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detail);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        TextView tvName = findViewById(R.id.tvName);
        TextView tvDescr = findViewById(R.id.tvDescr);

        tvName.setText(getIntent().getStringExtra( name: "name"));
        tvDescr.setText(getIntent().getStringExtra( name: "descr"));
    }
}
```

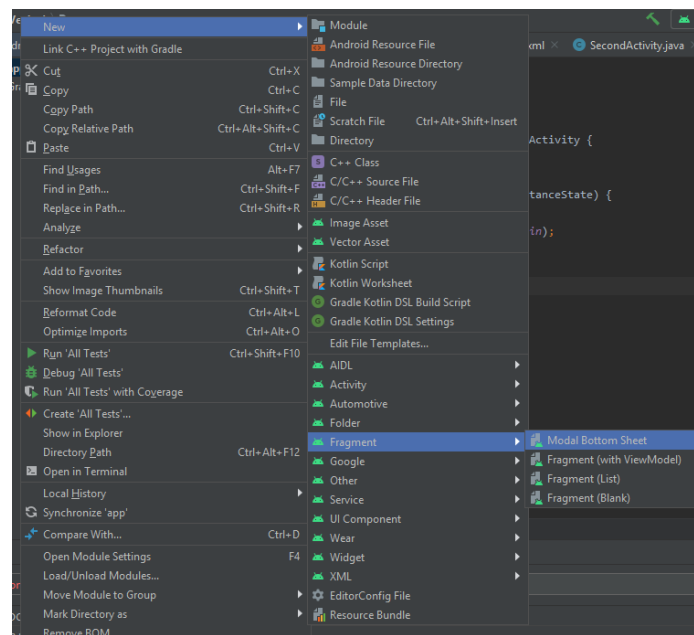
Slika 15. Dobavljanje podataka iz aktivnosti.

## 2. Fragmenti

Fragment možemo posmatrati kao podaktivnost. To je deo ponašanja ili GUI-a aktivnosti. Jedna aktivnost može da sadrži više fragmenata i jedan fragment može da bude sadržan u više aktivnosti.

### 2.1 Kreiranje fragmenta

Novi fragment se kreira odabirom stavke iz menija: File > New > Fragment (slika 16) ili jednostavnim kreiranjem Java klase koja nasleđuje klasu *Fragment*. U ovom primeru odabrali smo *Fragment (Blank)*.



Slika 16. Kreiranje novog fragmenta

Za razliku od aktivnosti, fragmente ne moramo da specificiramo unutar *Manifest* datoteke.

Na slikama 17 i 18 se vide klasa, *BlankFragment.java*, i izgled, *fragment\_blank.xml*, koji su kreirani za nas.

```

/**
 * A simple {@link Fragment} subclass.
 * Use the {@link BlankFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class BlankFragment extends Fragment {
    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public BlankFragment() {
        // Required empty public constructor
    }
}

```

Slika 17. Primer dela klase predefinisano fragmenta *BlankFragment*

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout 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"
    tools:context=".BlankFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Hello blank fragment" />

</FrameLayout>

```

Slika 18. *BlankFragment* layout

Isto kao i aktivnost, fragment ima specifičnu metodu za postavljanje layout-a fragmenta (slika 19). Takođe, na istoj slici se može videti i kako se vade podaci koji su poslani u fragment.

Kako se fragment postavlja na aktivnost možete pogledati u primeru za vežbe 2 na Git-u.

```

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup vg, Bundle data) {
    View view = inflater.inflate(R.layout.mysecondfragment_layout, vg, attachToRoot: false);

    Bundle bundle = getArguments();
    if (bundle != null){
        String param = bundle.getString( key: "SOME_PARAM_KEY", defaultValue: "DEFAULT");
        TextView textView = view.findViewById(R.id.textView2);
        textView.setText(param);
    }

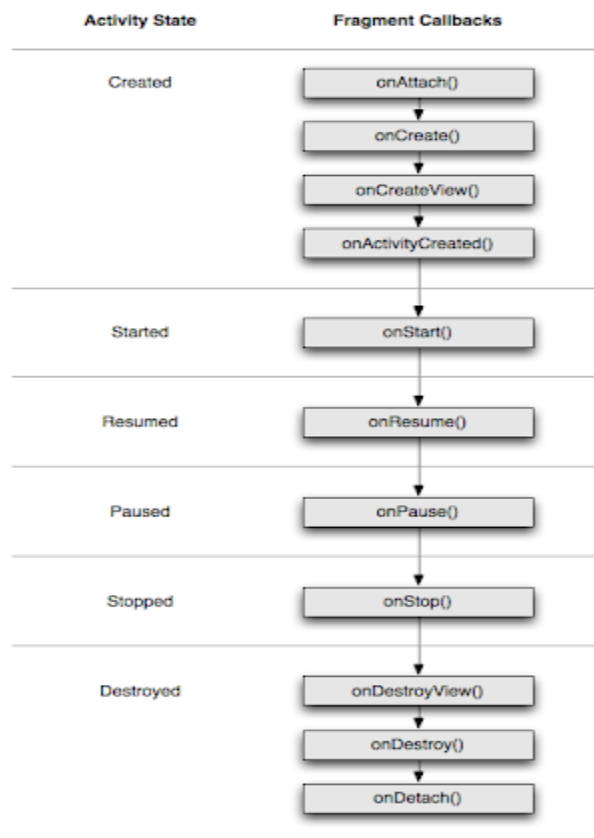
    return view;
}

```

Slika 19. Povezivanje fragmenta i njegovog layout-a

## 2.2 Životni ciklus fragmenta

Fragmenti takođe imaju životni ciklus i on zavisi od životnog ciklusa aktivnosti u kojoj se nalaze. Životni ciklus fragmenta je sličan kao i životni ciklus aktivnosti (slika 20), samo što je proširen sa dodatnim metodama koje omogućavaju interakciju sa aktivnošću u kojoj se nalaze.



Slika 20. Životni ciklus fragmenta

## onAttach

Metoda *onAttach* se poziva kada se fragment povezuje sa aktivnošću (slika 21).

```
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    Toast.makeText(getActivity(), text: "onAttach()", Toast.LENGTH_SHORT).show();
}
```

Slika 21. Primer *onAttach* metode

## onCreateView

Metoda *onCreateView* se poziva da bi se iscrtao korisnički interfejs fragmenta (slika 22).

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup vg, Bundle data) {
    return inflater.inflate(R.layout.myfragment_layout, vg, attachToRoot: false);
}
```

Slika 22. Primer *onCreateView* metode

## onActivityCreated

Ova metoda se poziva kada se *onCreate* metoda aktivnosti izvrši (slika 23).

```
@Override
public void onActivityCreated(Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    Toast.makeText(getActivity(), text: "onActivityCreated()", Toast.LENGTH_SHORT).show();
}
```

Slika 23. Primer *onActivityCreated* metode

## onDestroyView

Da bi se uništio korisnički interfejs fragmenta poziva se metoda *onDestroyView* (slika 24).

```
@Override
public void onDestroyView() {
    super.onDestroyView();
    Toast.makeText(getActivity(), text: "onDestroyView()", Toast.LENGTH_SHORT).show();
}
```

Slika 24. Primer *onDestroyView* metode

## onDetach

Na samom kraju se poziva *onDetach* kada se fragment odvezuje od aktivnosti (slika 25).

```
@Override
public void onDetach() {
    super.onDetach();
    Toast.makeText(getActivity(), text: "onDeattach()", Toast.LENGTH_SHORT).show();
}
```

Slika 25. Primer *onDetach* metode

### 3. Domaći

Domaći se nalazi na *Canvas-u* ([canvas.ftn.uns.ac.rs](https://canvas.ftn.uns.ac.rs)) na putanji *Вежбе/02 Вежбе/02 Задамак.pdf*.

Primer možete preuzeti na sledećem linku: <https://gitlab.com/antesevicceca/mobilne-aplikacije>

Za dodatna pitanja možete se obratiti asistentima:

- Svetlana Antešević ([svetlanaantesevic@uns.ac.rs](mailto:svetlanaantesevic@uns.ac.rs))
- Jelena Matković ([matkovic.jelena@uns.ac.rs](mailto:matkovic.jelena@uns.ac.rs))