

Koncepti – aktivnosti, fragmenti i namere

Doc. Dr Vladimir Milićević



UVOD

Osnovni koncepti Android aplikacije su: aktivnosti, fragmenti i namene.

Aktivnost je određena prozorom koji obuhvata korisnički interfejs mobilne aplikacije. Aplikacija može da ima jednu ili više aktivnosti čiji je osnovni zadatak obezbeđivanje komunikacije između korisnika i aplikacije.

Fragmenti predstavljaju male aktivnosti koje su grupisane na način da formiraju jednu veću, složenu, aktivnost. Poseban akcenat će biti stavljen na mogućnost istovremene upotrebe fragmenata i aktivnosti.

Konačno, u daljem izlaganju biće govora o veoma važnom mobilnom konceptu – namerama. Namere predstavljaju mehanizam pomoću kojeg aktivnosti iz različitih aplikacija funkcionišu zajedno sa ciljem izvršavanja zadataka neke konkretne aplikacije. Razumevanje i savladavanje ovog koncepta je jako bitno budući da će biti primenjivan prilikom izvršavanja brojnih ugrađenih aplikacija, poput: *Google Chrome, Telefon, Google Maps, Kontakti itd.*

UPOZNAVANJE KONCEPTA AKTIVNOST

Klasa koja implementira aktivnosti nasleđuje Activity klasu.

Da bi koncept **aktivnost** što bolje bio predstavljen, neophodno je prvo predstaviti JAVA klasu koja nasleđuje osnovnu klasu **Activity**. Neka to bude klasa MainActivity.java.

```
package com.metropolitan.prvaandroidapp;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

ANDROIDMANIFEST DATOTEKA

Svaka aktivnost aplikacije mora da bude deklarirana u AndroidManifest.xml datoteci.

Instrukcijom

`setContentView(R.layout.main)` navedena klasa vrši učitavanje korisničkog interfejsa pomoću xml datoteke `activity_main.xml` koja je smeštena u folderu `res/layout`. Prethodno je bilo posebno govora o ovoj datoteci i pokazano je kako se pomoću nje mogu dodavati komponente korisničkog interfejsa. Dalje, svaka aktivnost, koja je prisutna u aplikaciji, mora biti istaknuta u `AndroidManifest.xml` datoteci. Aktivnosti su grupisane u okviru xml tagova `<activity> </activity>`.

Sledi kod razmatrane `AndroidManifest.xml` datoteke.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.metropolitan.prvaandroidapp"
android:versionCode="1"
android:versionName="1.0" >
  <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="21" />
  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
      android:name=".MainActivity"
      android:label="@string/app_name" >
      <intent-filter>
        <action
          android:name="android.intent.action.MAIN" />
        <category
          android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
      </activity>
    </application>
</manifest>
```

ŽIVOTNI CIKLUS AKTIVNOSTI

Životni ciklus prati seriju aktivnosti od iniciranja do uklanjanja iz memorije.

Životni ciklus aktivnosti određen je serijom događaja definisanih klasom Activity koju nasleđuju klase aktivnosti Android aplikacija. Od velike važnosti je razumevanje sledećih događaja:

onCreate() – Poziva se prilikom prvog definisanja aktivnosti;

onStart()- Poziva se kada aktivnost postane vidljiva korisniku;

onResume() – Poziva se kada korisnik započne interakciju;

onStop() – Poziva se kada aktivnost više nije vidljiva korisniku;

onDestroy() – Poziva se pre uklanjanja aktivnosti;

onRestart() – Nakon zaustavljanja aktivnosti poziva se za njeno ponovno pokretanje.

Po osnovnim podešavanjima, kreirana aktivnost sadrži onCreate() događaj koji pomaže da se prikažu korisniku elementi korisničkog interfejsa.

ŽIVOTNI CIKLUS AKTIVNOSTI - NASTAVAK

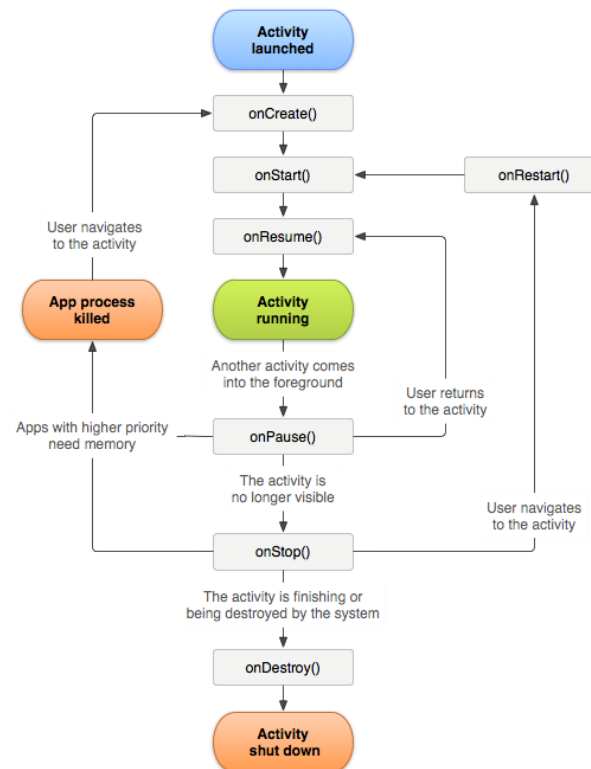
Aktivnosti se pakuju na stek aktivnosti.

Aktivnostima u sistemu se upravlja kao stekom aktivnosti. Svaka nova aktivnost se dodaje na vrh steka. Sve prethodne aktivnosti se nalaze ispod u steku i ostaju neaktivne sve dok tekuća aktivnost ne napusti stek.

Aktivnost se javlja u četiri osnovna oblika:

- Ako je aktivnost u prvom planu na ekranu, ima status *active* ili *running*;
- Ako je aktivnost izgubila fokus, a vidljiva je i dalje, pauzirana je;
- Ako je aktivnost u potpunosti u senci druge aktivnosti, zaustavljena je;
- Ako je aktivnost pauzirana ili zaustavljena, sistem može da je eliminiše iz memorije. Za naredno angažovanje aktivnosti neophodno je njeno restartovanje.

Sljedećom slikom prikazan je životni ciklus aktivnosti (izvor: <http://developer.android.com>).



Slika-1 Životni ciklus aktivnosti

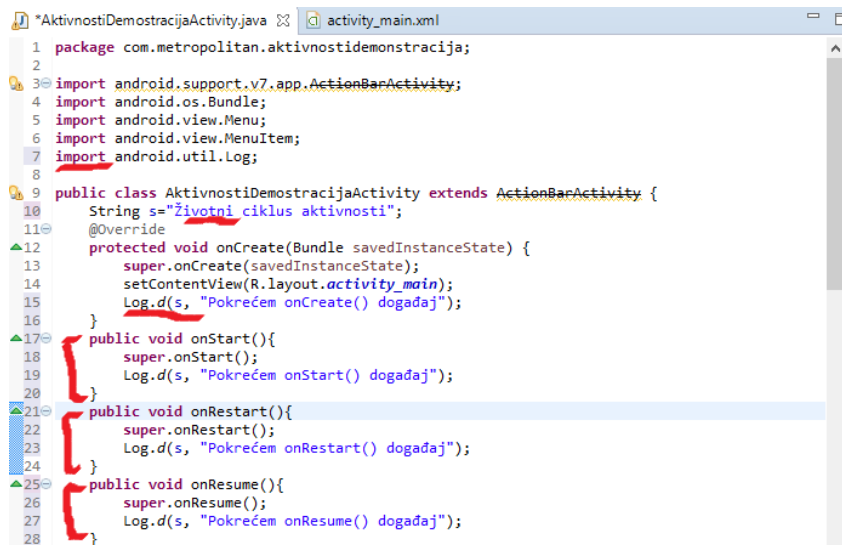
DODAVANJE AKTIVNOSTI U ACTIVITY DATOTEKU

Aktivnosti su realizovane specifičnim metodama implementiranim u tačno određenu JAVA datoteku.

Najbolji način za razumevanje funkcionisanja aktivnosti jeste uvođenje konkretnog primera:

Kreirati nov Android projekat sa nazivom AktivnostiDemonstracija;

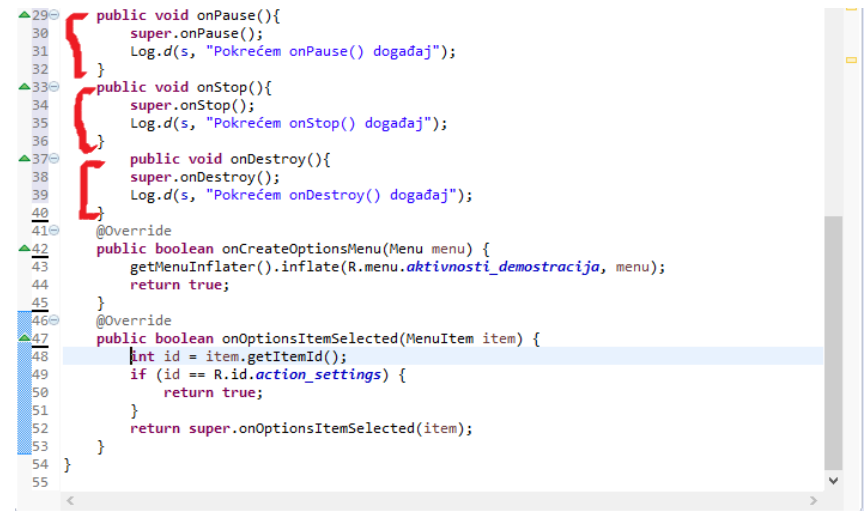
U JAVA datoteku aktivnosti ugraditi sledeći podvučeni kod:



```
1 package com.metropolitan.aktivnostidemonstracija;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.view.Menu;
6 import android.view.MenuItem;
7 import android.util.Log;
8
9 public class AktivnostiDemonstracijaActivity extends AppCompatActivity {
10     String s="Životni ciklus aktivnosti";
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_main);
15         Log.d(s, "Pokrećem onCreate() događaj");
16     }
17     public void onStart(){
18         super.onStart();
19         Log.d(s, "Pokrećem onStart() događaj");
20     }
21     public void onRestart(){
22         super.onRestart();
23         Log.d(s, "Pokrećem onRestart() događaj");
24     }
25     public void onResume(){
26         super.onResume();
27         Log.d(s, "Pokrećem onResume() događaj");
28     }
29 }
```

Slika-2a Dodavanje aktivnosti u Activity datoteku

Sledi nastavak koda:



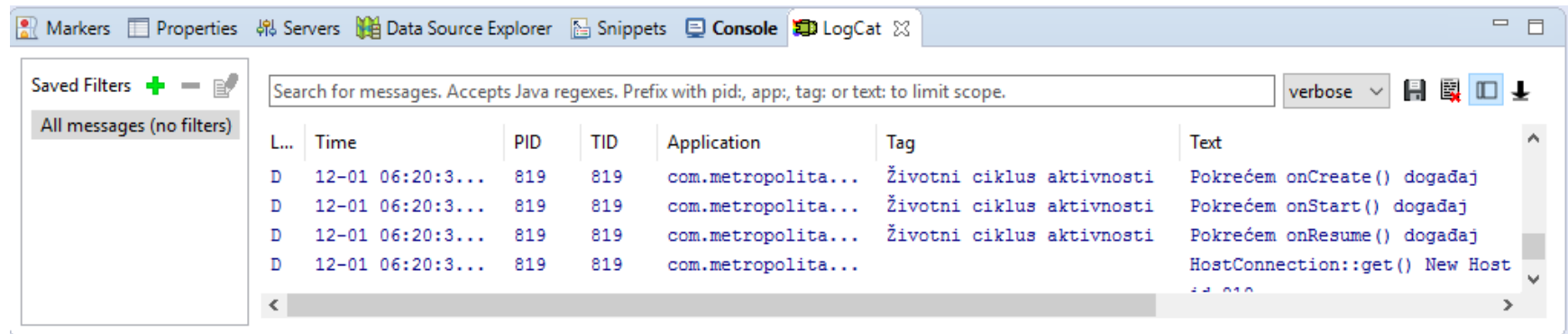
```
29     public void onPause(){
30         super.onPause();
31         Log.d(s, "Pokrećem onPause() događaj");
32     }
33     public void onStop(){
34         super.onStop();
35         Log.d(s, "Pokrećem onStop() događaj");
36     }
37     public void onDestroy(){
38         super.onDestroy();
39         Log.d(s, "Pokrećem onDestroy() događaj");
40     }
41     @Override
42     public boolean onCreateOptionsMenu(Menu menu) {
43         getMenuInflater().inflate(R.menu.aktivnosti_demonstracija, menu);
44         return true;
45     }
46     @Override
47     public boolean onOptionsItemSelected(MenuItem item) {
48         int id = item.getItemId();
49         if (id == R.id.action_settings) {
50             return true;
51         }
52         return super.onOptionsItemSelected(item);
53     }
54 }
55 }
```

Slika-2b Dodavanje aktivnosti u Activity datoteku-nastavak

PRAĆENJE ŽIVOTNOG CIKLUSA AKTIVNOSTI

Sve izvršene aktivnosti su evidentirane u LogCat prozoru.

U nastavku, neophodno je da se pritiskom na ctrl+s snimi izmenjena datoteka, a zatim pomoću F11 izvrši debugovanje na pokrenutom AVD emulatoru. Prvo što treba proveriti jeste LogCat prozor gde se može primetiti evidencija pokrenutih aktivnosti (sledeća slika).



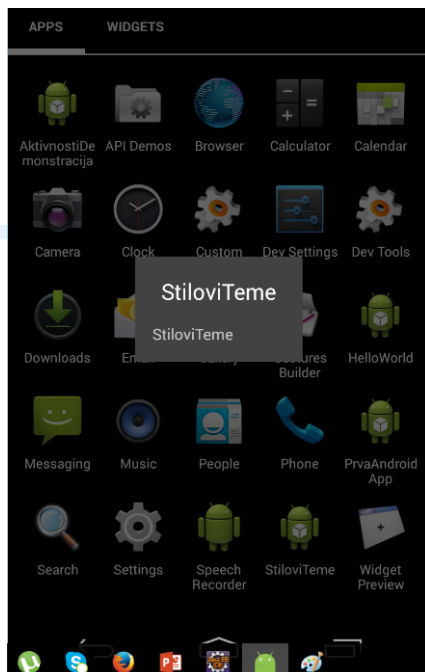
Slika-3 Evidentiranje aktivnosti u LogCat prozoru

UPOTREBA STILOVA I TEMA

Inicijalna podešavanja aktivnosti mogu da se prilagode i promene primenom stilova i tema.

Početnim podešavanjima, aktivnosti se predaje celokupan ekran. Primenom tema moguće je promeniti početna podešavanja. Tako, ako se želi da se aktivnost prikaže kao npr. *iskačući prozor* kojim se obavlja dijalog između sistema i korisnika, primenom teme za dijalog u tagu <Activity ...> datoteke AndroidManifest.xml početna podešavanja se zamenjuju željenim. Sledećim kodom je prikazan deo AndroidManifest.xml datoteke u kojem su izvršene naznačene korekcije.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.metropolitan.stiloviteme"
4     android:versionCode="1"
5     android:versionName="1.0" >
6
7     <uses-sdk
8         android:minSdkVersion="11"
9         android:targetSdkVersion="23" />
10
11 <application
12     android:allowBackup="true"
13     android:icon="@drawable/ic_launcher"
14     android:theme="@style/Theme.AppCompat.Dialog" ←
15     android:label="@string/app_name"
16
17 >
18     <activity
19         android:label="@string/app_name"
20         android:name=".Glavna_aktivnost"
21
22     >
23         <intent-filter>
24             <action android:name="android.intent.action.MAIN" />
25
26             <category android:name="android.intent.category.LAUNCHER" />
27         </intent-filter>
28     </activity>
29 </application>
30
31 </manifest>
```



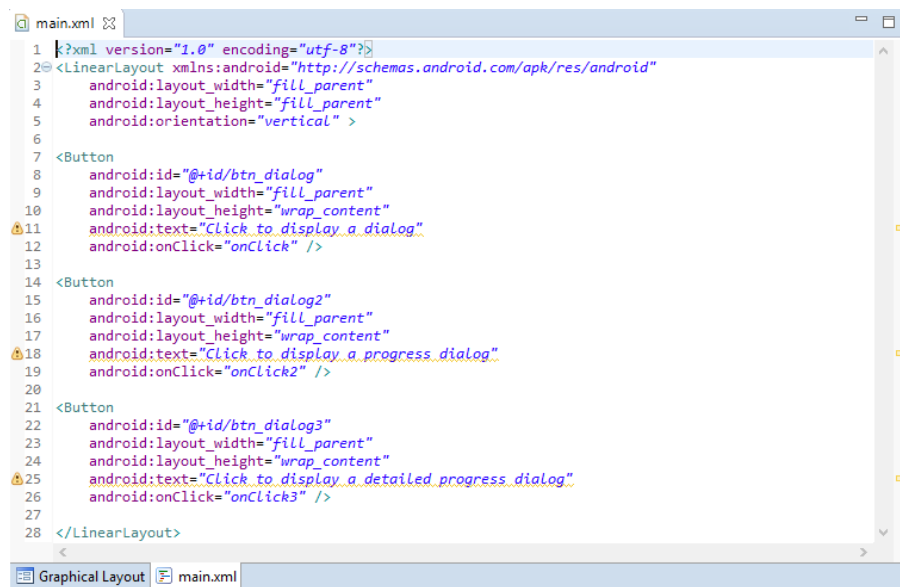
Slika-7 Dodavanje teme u AndroidManifest.xml

PRIKAZIVANJE OKVIRA DIJALOGA KORIŠĆENJEM AKTIVNOSTI

Potvrda interakcije sa korisnikom prikazuje se okvirom za dijalog.

U realnim situacijama veoma često je neophodno prikazati okvir za dijalog koji u sebi nosi potvrdu interakcije sa korisnikom. U tom slučaju je neophodno izvršiti predefinisanje zaštićene metode `onCreateDialog()` definisane u osnovnoj klasi aktivnosti. Za ilustraciju biće iskorišćen sledeći primer.

Otvoriti Eclipse IDE, kreirati u `com.metropolitan` paketu novi projekat sa nazivom `Dialog` i proširiti `main.xml` datoteku sledećim kodom:

The image shows a screenshot of the Eclipse IDE's code editor. The file being edited is 'main.xml'. The XML code defines a vertical LinearLayout containing three buttons. The first button has the text 'Click to display a dialog' and an onClick listener 'onClick'. The second button has the text 'Click to display a progress dialog' and an onClick listener 'onClick2'. The third button has the text 'Click to display a detailed progress dialog' and an onClick listener 'onClick3'. The XML code is as follows:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:orientation="vertical" >
6
7     <Button
8         android:id="@+id/btn_dialog"
9         android:layout_width="fill_parent"
10        android:layout_height="wrap_content"
11        android:text="Click to display a dialog"
12        android:onClick="onClick" />
13
14    <Button
15        android:id="@+id/btn_dialog2"
16        android:layout_width="fill_parent"
17        android:layout_height="wrap_content"
18        android:text="Click to display a progress dialog"
19        android:onClick="onClick2" />
20
21    <Button
22        android:id="@+id/btn_dialog3"
23        android:layout_width="fill_parent"
24        android:layout_height="wrap_content"
25        android:text="Click to display a detailed progress dialog"
26        android:onClick="onClick3" />
27
28 </LinearLayout>
```

Slika-8 Podešavanje main.xml datoteke

PRIKAZIVANJE OKVIRA DIJALOGA KORIŠĆENJEM AKTIVNOSTI - NASTAVAK

Da bi bio prokazan okvir za dijalog neophodno je implementirati onCreateDialog() metodu u klasi Activity.

U JAVA datoteku, koja je po inicijalnim podešavanjima nazvana DialogActivity.java, predefinisaćemo novim kodom prikazanim u sledećem izlaganju:

```
1 package com.metropolitan.Dialog;
2
3 import android.app.Activity;
4 import android.app.AlertDialog;
5 import android.app.AlertDialog.Builder;
6 import android.app.Dialog;
7 import android.app.ProgressDialog;
8 import android.content.DialogInterface;
9 import android.os.Bundle;
10 import android.view.View;
11 import android.widget.Toast;
```

Slika-9a Uključivanje neophodnih paketa u primer

U nastavku biće programirana metoda onCreateDialog() klase DialogActivity:

```
13 public class DialogActivity extends Activity {
14     CharSequence[] items = { "FIT", "Fakultet za Menadžment", "Fakultet digitalnih umetnosti" };
15     boolean[] itemsChecked = new boolean [items.length];
16
17     ProgressDialog progressDialog;
18     /** Poziva se prvim definisanjem aktivnosti. */
19     @Override
20     public void onCreate(Bundle savedInstanceState) {
21         super.onCreate(savedInstanceState);
22         setContentView(R.layout.main);
23     }
24
25     public void onClick(View v) {
26         showDialog(0);
27     }
28
29     public void onClick2(View v) {
30         //---prikazuje dijalog---
31         final ProgressDialog dialog = ProgressDialog.show(
32             this, "Nešto se dešava.", "Sačekajte...", true);
33         new Thread(new Runnable(){
34             public void run(){
35                 try {
36                     //---simulacija da nešto radi---
37                     Thread.sleep(5000);
38                     //---odjavljuje dijalog---
39                     dialog.dismiss();
40                 } catch (InterruptedException e) {
41                     e.printStackTrace();
42                 }
43             }
44         }).start();
45     }
46 }
```

Slika-9b Programiranje metode onCreateDialog()

PRIKAZIVANJE OKVIRA DIJALOGA KORIŠĆENJEM AKTIVNOSTI – JOŠ KODA

Neophodno je definisati akcije koje se realizuju klikovima na definisanu dugmad.

Programski kod nastavak:

```
47 public void onClick3(View v) {
48     showDialog(1);
49     progressDialog.setProgress(0);
50
51     new Thread(new Runnable(){
52         public void run(){
53             for (int i=1; i<=15; i++) {
54                 try {
55                     //---simulacija da nešto radi---
56                     Thread.sleep(1000);
57                     //---osvežavanje dijaloga---
58                     progressDialog.incrementProgressBy((int)(100/15));
59                 } catch (InterruptedException e) {
60                     e.printStackTrace();
61                 }
62             }
63             progressDialog.dismiss();
64         }
65     }).start();
66 }
```

Slika-9c Programiranje onCreate() metode

Konačno onCreate() metoda:

```
69     protected Dialog onCreateDialog(int id) {
70         switch (id) {
71             case 0:
72                 return new AlertDialog.Builder(this)
73                     .setIcon(R.drawable.ic_launcher)
74                     .setTitle("Dijalog sa malo teksta...")
75                     .setPositiveButton("OK",
76                         new DialogInterface.OnClickListener() {
77                             public void onClick(DialogInterface dialog, int whichButton)
78                             {
79                                 Toast.makeText(getBaseContext(),
80                                     "OK je kliknut!", Toast.LENGTH_SHORT).show();
81                             }
82                         }
83                     )
84                     .setNegativeButton("Cancel",
85                         new DialogInterface.OnClickListener() {
86                             public void onClick(DialogInterface dialog, int whichButton)
87                             {
88                                 Toast.makeText(getBaseContext(),
89                                     "Cancel je kliknut!", Toast.LENGTH_SHORT).show();
90                             }
91                         }
92                     )
93                     .setMultiChoiceItems(items, itemsChecked,
94                         new DialogInterface.OnMultiChoiceClickListener() {
95                             public void onClick(DialogInterface dialog,
96                                 int which, boolean isChecked) {
97                                 Toast.makeText(getBaseContext(),
98                                     items[which] + (isChecked ? " checked!:" " unchecked!"),
99                                     Toast.LENGTH_SHORT).show();
100                             }
101                         }
102                     ).create();
```

Slika-9d Programiranje metode onCreate()

PRIKAZIVANJE OKVIRA DIJALOGA KORIŠĆENJEM AKTIVNOSTI – KRAJ I DEMONSTRACIJA

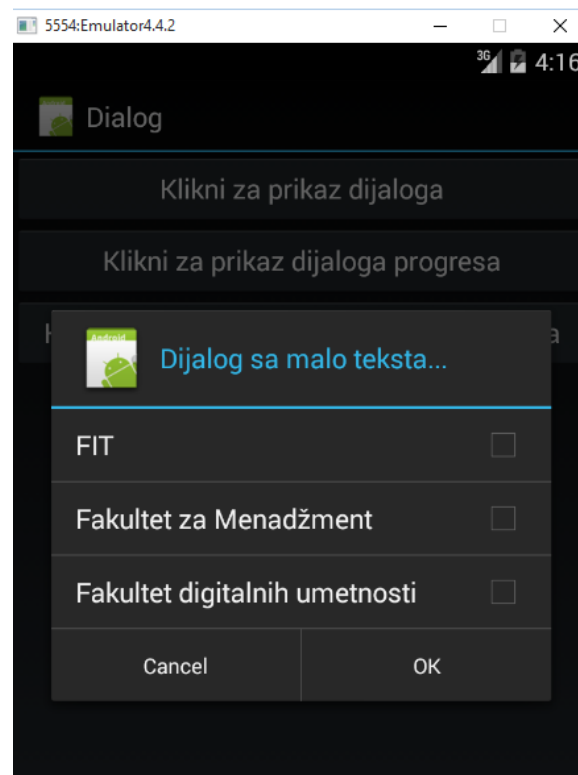
Klikom na dugmad biće prikazane sve akcije definisane metodom.

Kraj programskog koda:

```
103
104
105     case 1:
106         progressDialog = new ProgressDialog(this);
107         progressDialog.setIcon(R.drawable.ic_launcher);
108         progressDialog.setTitle("Preuzimanje datoteka...");
109         progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
110         progressDialog.setButton(DialogInterface.BUTTON_POSITIVE, "OK",
111             new DialogInterface.OnClickListener() {
112                 public void onClick(DialogInterface dialog,
113                     int whichButton)
114                 {
115                     Toast.makeText(getBaseContext(),
116                         "OK je kliknut!", Toast.LENGTH_SHORT).show();
117                 }
118             });
119         progressDialog.setButton(DialogInterface.BUTTON_NEGATIVE, "Cancel",
120             new DialogInterface.OnClickListener() {
121                 public void onClick(DialogInterface dialog,
122                     int whichButton)
123                 {
124                     Toast.makeText(getBaseContext(),
125                         "Cancel je kliknut!", Toast.LENGTH_SHORT).show();
126                 }
127             });
128         return progressDialog;
129     }
130 }
131 }
```

Slika-9e Programiranje metode onCreate()

Klikom na F11 vrši se debugovanje programa i pokretanje u emulatoru.



Slika-10 Demonstracija urađenog primera

FUNKCIONISANJE DIJALOG OKVIRA

Metoda `onCreateDialog()` je reakcija na kreiranje okvira za dijalog kojim upravlja odgovarajuća aktivnost.

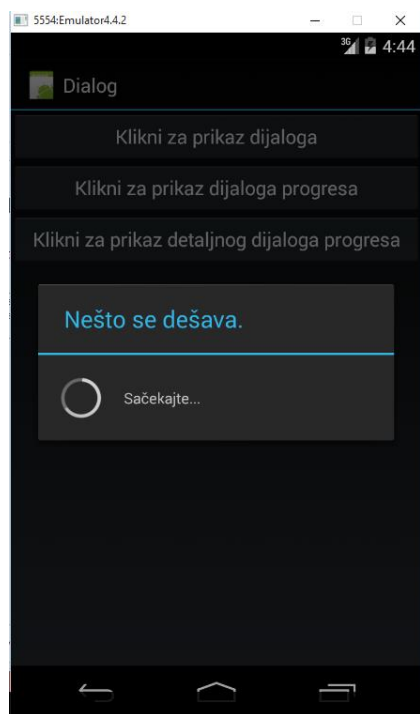
U prethodni primer ugrađena su tri veoma česta načina korišćenja okvira za dijalog. Da bi, uopšte, okvir za dijalog bio prikazan, neophodno je ispuniti sledeće uslove:

- Implementirana metoda `onCreateDialog()` u klasi aktivnosti;
- Pozivanje `onCreateDialog()` metode se dešava nakon izvršavanja metode `showDialog()` koja prihvata celobrojni argument koji odgovara konkretnom okviru za dijalog;
- Da bi okvir za dijalog bio kreiran, neophodno je koristiti Builder konstruktor `AlertDialog` klase;

PRIKAZIVANJE OKVIRA ZA DIJALOG PROGRESA

Dijalog progresna je najčešće korišćena komponenta korisničkog interfejsa u Android aplikacijama.

Kada aplikacija ukazuje na zadatke koji se izvršavaju relativno dugo, najčešće dolazi do prikazivanja okvira za dijalog progresna koji korisniku nosi odgovarajuću poruku



Slika-11 Dijalog okvira progresna

Da bi okvir za dijalog progresna bio kreiran neophodno je kreiranje objekta klase *ProgressDialog* i izvršavanje njene metode *show()*. Simulacija zadatka koji se duže izvršava implementirana je metodom *run()*.

```
public void onClick2(View v) {
    //---prikazuje dijalog---
    final ProgressDialog dialog = ProgressDialog.show(
        this, "Nešto se dešava.", "Sačekajte...", true);
    new Thread(new Runnable(){
        public void run(){
            try {
                //---simulacija da nešto radi---
                Thread.sleep(5000);
                //---odjavljuje dijalog---
                dialog.dismiss();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }).start();
}
```

Slika-12 Konstruktor objekta klase *ProgressDialog* i metoda *run()*

KORIŠĆENJE NAMERA ZA POVEZIVANJE AKTIVNOSTI

U Android operativnom sistemu namere imaju ulogu navigatora između pojedinačnih aktivnosti.

Svaka Android aplikacija može biti izgrađena od jedne ili više aktivnosti. Ukoliko aplikacija sadrži više aktivnosti neophodno je obezbediti mehanizme kojima se vrši navigacija sa jedne aktivnosti na drugu. Taj zadatak u Android aplikacijama realizovan je pomoću namera. Za predstavljanje i razumevanje ovog koncepta biće uveden novi primer:

1. Kreirati nov projekat pod nazivom KoriscenjeNamera
2. Desnim klikom na folder src i izborom opcija new – class kreirati nove dve klase pod nazivima: DrugaAktivnost i TrecaAktivnost.
3. U AndroidManifest.xml datoteci uraditi sledeće dopune:

```
<activity
    android:label="DrugaAktivnost"
    android:name=".DrugaAktivnost" >
    <intent-filter >
        <action android:name="com.metropolitan.DrugaAktivnost" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

<activity
    android:label="TrecaAktivnost"
    android:name=".TrecaAktivnost" >
    <intent-filter >
        <action android:name="com.metropolitan.DrugaAktivnost " />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Slika-13 Dodavanje novih aktivnosti u AndroidManifest.xml

KREIRANJE VLASTITIH XML DATOTEKA ZA AKTIVNOSTI

Svaka aktivnost mora da ima svoju xml datoteku.

U res/layout folderu neophodno je kreirati nove dve xml datoteke pod nazivima: DrugaAktivnost.xml i TrecaAktivnost.xml. Njihov kod odgovara modifikaciji main.xml datoteke na sledeći način:

DrugaAktivnost.xml	TrecaAktivnost.xml
<pre><?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="Ovo je druga aktivnost!" /> <TextView android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Unesite vase ime" /> <EditText android:id="@+id/txt_username" android:layout_width="fill_parent" android:layout_height="wrap_content" /> <Button android:id="@+id/btn_OK" android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="OK" android:onClick="onClick"/> </LinearLayout></pre>	<pre><?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="Ovo je treca aktivnost!" /> </LinearLayout></pre>

Slika-14 Kreiranje vlastitih xml datoteka za aktivnosti

MODIFIKOVANJE JAVA DATOTEKA AKTIVNOSTI

Za svaku aktivnost neophodno je definisati vlastitu onCreate() metodu.

U datoteke DrugaAktivnost.java i TrecaAktivnost.java dodati linije programskog koda na način prikazan sledećom slikom:

DrugaAktivnost.java	TrecaAktivnost.java
<pre>package com.metropolitan.KoriscenjeNamera; import android.app.Activity; import android.content.Intent; import android.net.Uri; import android.os.Bundle; import android.view.View; import android.widget.EditText; public class SecondActivity extends Activity{ @Override public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.drugaaktivnost); } public void onClick(View view) { Intent data = new Intent(); //---definisiranje EditText pogleda--- EditText txt_username = (EditText) findViewById(R.id.txt_username); //---definisiranje podataka koji se vraćaju data.setData(Uri.parse(txt_username.getText().toString())); setResult(RESULT_OK, data); //---closes the activity--- finish(); } }</pre>	<pre>package com.metropolitan.KoriscenjeNamera; import android.app.Activity; import android.os.Bundle; public class ThirdActivity extends Activity{ @Override public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.trecaaktivnost); } }</pre>

Slika-15 Programiranje JAVA datoteka aktivnosti

MODIFIKACIJA MAIN.XML I KORISCENJENAMERA.JAVA DATOTEKA

Na kraju je neophodno modifikovati i glavne datoteke

Da bi bilo moguće potpuno sagledavanje angažovanja više aktivnosti u jednoj Android aplikaciji neophodno je još modifikovati datoteke main.xml i KoriscenjeNamera.java na sledeći način:

KoriscenjeNamera.java	main.xml
<pre>package com.metropolitan.KoriscenjeNamera; import android.app.Activity; import android.content.Intent; import android.net.Uri; import android.os.Bundle; import android.view.View; import android.widget.Toast; public class KoriscenjeNameraActivity extends Activity{ int request_Code = 1; /** Poziva se kada se aktivnost kreira. */ @Override public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.main); } public void onClick(View view) { startActivityForResult(new Intent("net.learn2develop.DrugaAktivnost"), request_Code); } public void onActivityResult(int requestCode, int resultCode, Intent data) { if (requestCode == request_Code) { if (resultCode == RESULT_OK) { Toast.makeText(this,data.getData().toString(), Toast.LENGTH_SHORT).show(); } } } }</pre>	<pre><?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" > <Button android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Prikazuje drugu aktivnost" android:onClick="onClick"/> </LinearLayout></pre>

Slika-16 Korekcije glavnih JAVA i xml datoteka

DEMONSTRACIJA ANGAŽOVANJA VIŠE AKTIVNOSTI U APLIKACIJI

Aktivnost je definisana odgovarajućom komponentom korisničkog interfejsa i klasom.

Kao što je naglašeno aktivnost je definisana odgovarajućom komponentom korisničkog interfejsa i klasom pa ukoliko je neophodno dodati novu aktivnost u projekat, ove dve komponente moraju biti uključene. Kao što je već pomenuto, aktivnost se dodaje u aplikaciju navođenjem odgovarajućeg koda u `AndroidManifest.xml`. U konkretnom primeru taj kod nosi sledeće informacije:

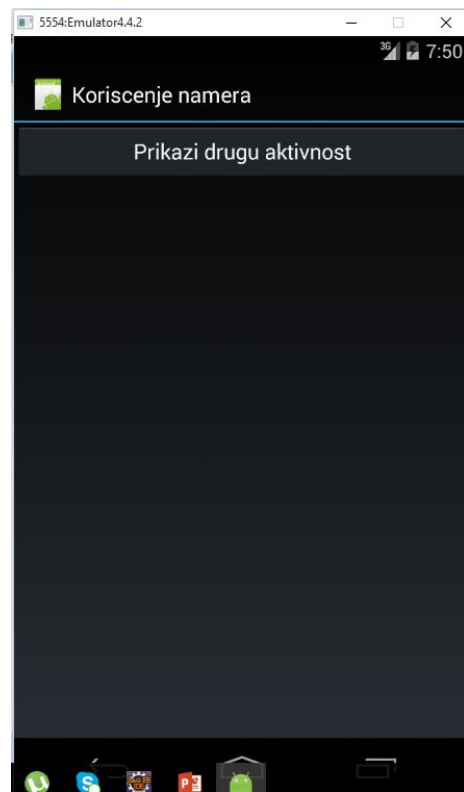
1. Oznake novih aktivnosti su *DrugaAktivnost* i *TrecaAktivnost*;
2. Naziv filtera za nove aktivnosti je *com.metropolitan.DrugaAktivnost* i *com.metropolitan.TrecaAktivnost*;
3. Kategorija za filter namera je *android.intent.category.DEFAULT*. Ova kategorija je neophodna da bi neka aktivnost mogla da bude pokrenuta drugom aktivnošću pomoću metode *startActivity()*;
4. Klikom na taster, metodom *startActivity()* pokreće se nova aktivnost npr *DrugaAktivnost* kreiranjem objekta klase *Intent* i prosleđivanjem filtera namera na sledeći način:

```
public void onClick(View view) {  
    startActivity (new Intent(" com.metropolitan.DrugaAktivnost "));  
}
```

DEMONSTRACIJA ANGAŽOVANJA VIŠE AKTIVNOSTI U APLIKACIJI - NASTAVAK

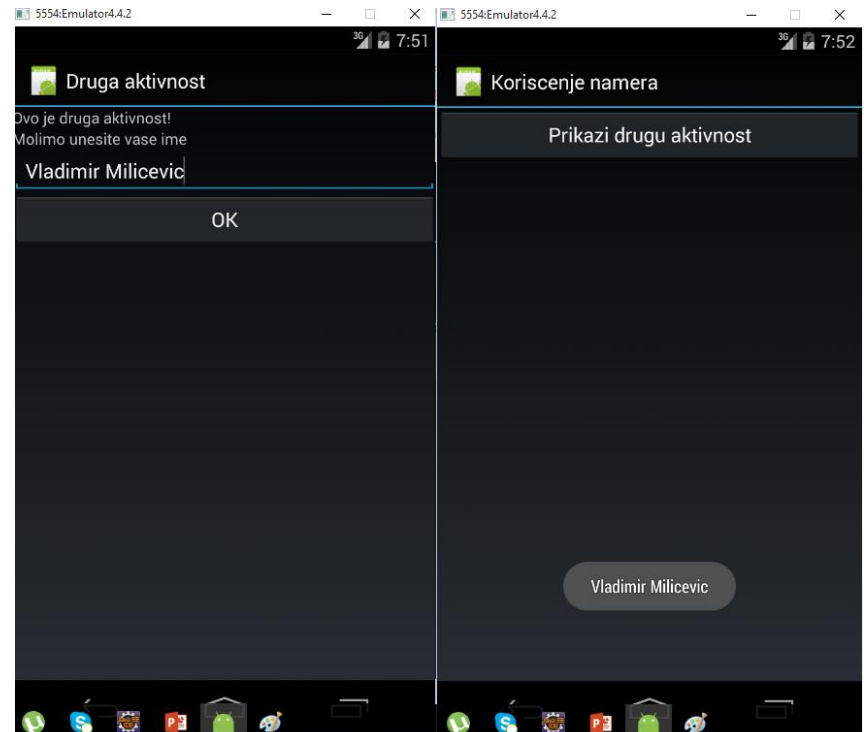
Pokretanjem emulatora testira se program za upravljanje aktivnostima.

Pokretanjem programa otvara se početni ekran sa porukom koja ukazuje na način pokretanja druge aktivnosti.



Slika-17 Početni ekran aplikacije

Klikom na dugme pokreće se druga aktivnost koja zahteva akcije kao na sledećoj slici.



Slika-18 Aktivnost DrugaAktivnost

IZBEGAVANJE SUDARANJA FILTERA NAMERE

Ukoliko dve aktivnosti imaju isti naziv filtera Android prikazuje selekciju aktivnosti.

Ukoliko se pogleda u kod AndroidManifest.xml datoteke koji ukazuje na drugu i trecu aktivnost moguće je uočiti da obe aktivnosti koriste isti filter:

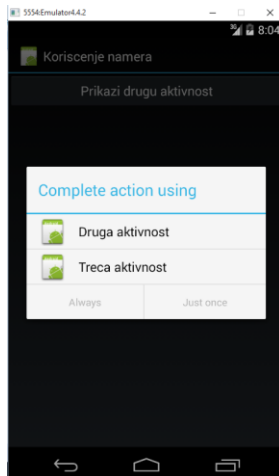
```
<intent-filter >
```

```
    <action android:name=„com.metropolitian.DrugaAktivnost" />
```

```
    <category android:name="android.intent.category.DEFAULT" />
```

```
</intent-filter>
```

Izvršavanjem metode startActivity() Android operativni sistem će pokazati selekciju aktivnosti kao na sledećoj slici.

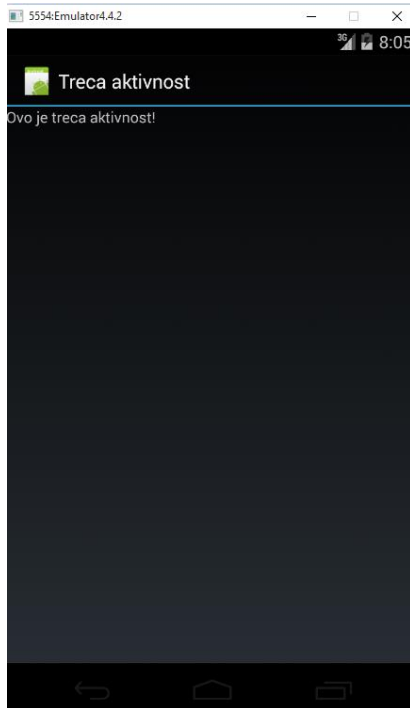


Slika-19 Selekcija aktivnosti

PRELAZAK SA JEDNE AKTIVNOSTI NA DRUGU

Metoda `startActivity()` inicira drugu aktivnost.

Izvršavanjem metode `startActivity()` i izborom treće aktivnosti u Android selekciji aktivnosti dolazi do izvršavanja navedene aktivnosti. Korigovanjem koda u `AndroidManifest.xml` datoteci, gde će kao filter namera biti uveden `com.metropolitan.TrecaAktivnost`, izbegava se navedena selekcija i dolazi do direktnog izvršavanja treće aktivnosti.



Slika-20 Aktivnost
TrecaAktivnost

VRAĆANJE REZULTATA IZ INICIRANE AKTIVNOSTI

Metoda `startActivity()` inicira drugu aktivnost ali ne vraća rezultat u trenutnu aktivnost.

Metoda `startActivity()` inicira drugu aktivnost ali ne vraća rezultat u trenutnu aktivnost. Ako se obrati pažnja na primer, moguće je uočiti da se u drugoj aktivnosti zahteva unošenje korisničkih podataka. Klikom na odgovarajuće dugme, ovi podaci trebalo bi da se proslede u početnu aktivnost. Za navedenu akciju koristi se metoda `startActivityForResult()`, a to je prikazano slikom18. Neophodno je navesti i sledeće:

1. Da bi aktivnost bila inicirana i rezultati izvršavanja bili prikazani, neophodno je upotrebiti navedenu metodu na sledeći način:

```
startActivityForResult (new Intent(„com.metropolitan.DrugaAktivnost“), requestCode);
```
2. Da bi aktivnost vratila vrednost u polaznu aktivnost, neophodno je koristiti Intent objekat koji koristi metodu `setData()` za vraćanje podataka nazad.
3. Metoda `setResult()` definiše kod (`RESULT_OK` ili `RESULT_CANCELLED`) i Intent objekat koji se vraćaju u polaznu aktivnost.
4. Metoda `finish()` prekida tekuću aktivnost i vraća kontrolu na polaznu aktivnost.
5. Polazna aktivnost mora da poseduje metodu `onActivityResult()` koja se izvršava kada se vraća vrednost iz inicirane aktivnosti.

UPOZNAVANJE KONCEPTA FRAGMENT

Fragmenti su još jedna forma aktivnosti.

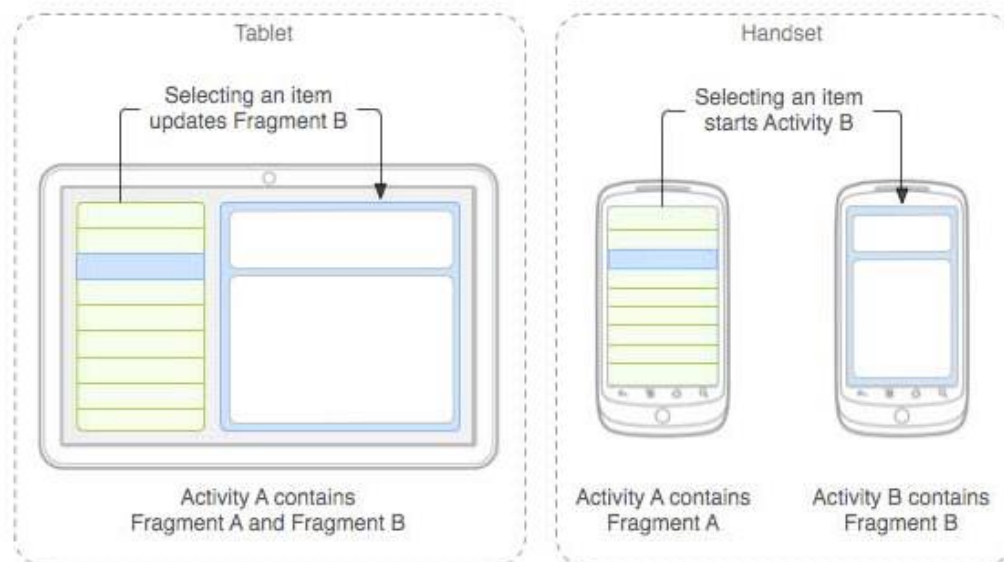
Fragmenti su detalji aktivnosti koji omogućavaju višemodularni dizajn aktivnosti. Fragment može da se tumači kao podaktivnost.

Za fragmente važi sledeće:

- Svaki fragment ima vlastiti izgled, ponašanje i životni ciklus;
- Fragmente je moguće dodavati i brisati iz aktivnosti dok je ona aktivna;
- Moguće je kombinovati više fragmenata u jednu aktivnost sa ciljem postizanja složenog korisničkog interfejsa;
- Životni ciklus fragmenta je u direktnoj vezi sa životnim ciklusom aktivnosti u koju je ugrađen fragment;
- Fragment može da implementira ponašanje koje ne uključuje komponentu korisničkog interfejsa;

- Fragmenti su uključeni u Android API od verzije Honeycomb (API 11).

Fragmenti se kreiraju kroz nasleđivanje klase *Fragment*, a dodaju se u aktivnost uvođenjem taga `<Fragment />` u xml datoteku aktivnosti. Sledećom slikom prikazano je kako dva UI modula mogu biti kombinovana kao jedna aktivnost na tablet dizajnu ili kao odvojene na dizajnu mobilnog telefona. Aktivnost je u direktnoj vezi sa veličinom ekrana.

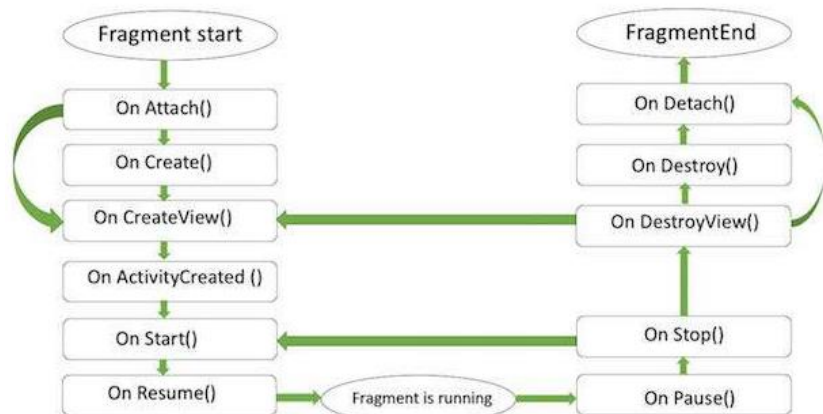


Slika-1 Fragmenti i veličina ekrana (izvor tutorialspoint.com)

ŽIVOTNI CIKLUS FRAGMENTATA

Fragmenti, kao i aktivnosti, imaju sopstveni životni ciklus.

Android fragmenti imaju vlastite životne cikluse veoma slično kao i aktivnosti. Sledećom slikom prikazane su različite faze životnog ciklusa fragmenta.



Slika-2 Životni ciklus fragmenta (izvor tutorialspoint.com)

Najveći broj stanja fragmenata identičan je stanjima aktivnosti. Ipak, postoji nekoliko stanja karakterističnih isključivo za fragmente:

- *onAttached()* – Izvršava se kada se fragment povezuje sa aktivnošću;
- *onCreateView()* – Izvršava se kada se definiše pogled na fragment;
- *onActivityCreated()* – Izvršava se kada se vrati rezultat *onCreate()* metode aktivnosti.
- *onDestroyView()* – Izvršava se kada se ukloni pogled na fragment;
- *onDetach()* – Izvršava se kada se fragment ukloni iz određene aktivnosti.

UPOTREBA FRAGMENTATA

Kreiranje fragmenata prati nekoliko obaveznih koraka tokom kreiranja aplikacije.

Kreiranje fragmenata teče kroz nekoliko jednostavnih koraka:

1. Prvo je neophodno odlučiti koliko fragmenata će biti uključeno u aktivnost;
2. Zatim, u zavisnosti od broja fragmenata, kreiraju se klase koje nasleđuju klasu *Fragment*;
3. Posebno, za svaki fragment, neophodno je u xml fajlu definisati izgled koji će fragmenti imati.
4. Konačno, fajl aktivnosti će morati da bude modifikovan tako da definiše aktuelnu logiku angažovanja fragmenata.

TIPOVI FRAGMENTATA

Fragmenti su podeljeni u tri nivoa.

U osnovi fragmenti su podeljeni u tri nivoa i to:

1. Fragmenti jednog okvira (single frame) – Ovakvi fragmenti imaju najčešću upotrebu kod uređaja sa malim ekranima, poput mobilnih telefona.
2. Lista fragmenata – Fragmenti su poređani u posebnu listu pogleda;
3. Transakcije fragmenata – Primenom transakcija fragmenata moguće je pomerati jedan fragment ka drugom.



Slika-3 Tipovi fragmenata

PRIMER KORIŠĆENJA FRAGMENTATA

Na novom Android projektu biće demonstrirana upotreba fragmenata.

Sledećom vežbom biće demonstrirana upotreba fragmenata:

1. Koristeći Eclipse IDE kreirati novi projekat i nazvati ga *Fragmenti*.
2. Modifikovati datoteku main.xml i kreirati dve nove fragment1.xml i fragment2.xml;
3. Kreirati dve klase Fragment1.java i Fragment2.java;
4. Prevesti program i pokrenuti emulator za testiranje.
5. Testirati funkcionalnost programa za prezentaciju fragmenata

PRIMER KORIŠĆENJA FRAGMENTATA – XML DATOTEKE

Program sadrži glavnu xml datoteku i xml datoteke fragmenata.

U main.xml datoteci dodati programski kod kao što je prikazano prvom kolonom sledeće slike, a zatim kreirati datoteke fragment1.xml i fragment2.xml na način prikazan drugom i trećom kolonom slike. Sve datotene moraju da se nalaze u res/layout folderu.

main.xml	fragment1.xml	fragment2.xml
<pre><?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="horizontal" > <fragment android:name="com.metropolitan.Fragments.Fragment1" android:id="@+id/fragment1" android:layout_weight="1" android:layout_width="0px" android:layout_height="match_parent" /> <fragment android:name="com.metropolitan.Fragments.Fragment2" android:id="@+id/fragment2" android:layout_weight="1" android:layout_width="0px" android:layout_height="match_parent" /> </LinearLayout></pre>	<pre><?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:orientation="vertical" android:layout_width="fill_parent" android:layout_height="fill_parent" android:background="#00FF00" > <TextView android:id="@+id/lblFragment1" android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Ovo je prvi fragment" android:textColor="#000000" android:textSize="25sp" /> </LinearLayout></pre>	<pre><?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:orientation="vertical" android:layout_width="fill_parent" android:layout_height="fill_parent" android:background="#FFFE00" > <TextView android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Ovo je drugi fragment" android:textColor="#000000" android:textSize="25sp" /> <Button android:id="@+id/btnGetText" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Prosledite tekst u prvi fragment" android:textColor="#000000" android:onClick="onClick" /> </LinearLayout></pre>

Slika-4 xml datoteke projekta

PRIMER KORIŠĆENJA FRAGMENTATA – JAVA DATOTEKE

Neophodno je dodati i dve java datoteke za fragmente.

U paketu, koji je nazvan com.metropolitan.Fragments, a u folderu src kreirati dve datoteke koje sadrže JAVA klase fragmenata. Datoteke će biti nazvane Fragment1.java i Fragment2.java. Kod klasa prikazan je sledećom slikom.

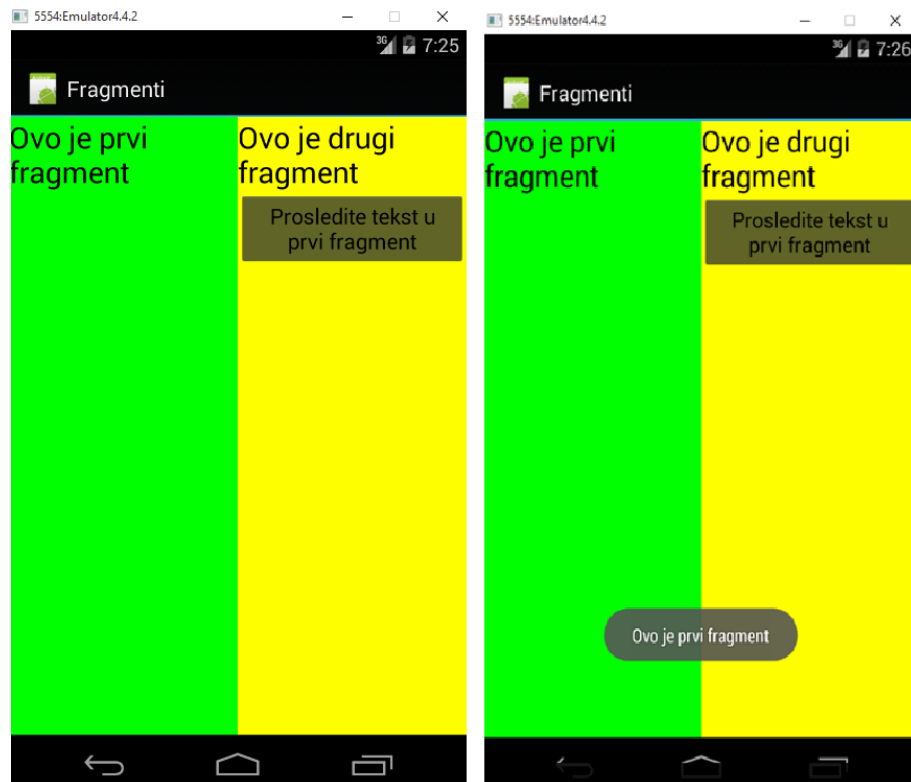
Fragment1.java	Fragment2.java
<pre>package com.metropolitan.Fragments; import android.app.Activity; import android.app.Fragment; import android.os.Bundle; import android.util.Log; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; public class Fragment1 extends Fragment { @Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) { Log.d("Fragment 1", "onCreateView"); //---kreiranje izgleda fragmenta--- return inflater.inflate(R.layout.fragment1, container, false); } @Override public void onAttach(Activity activity) { super.onAttach(activity); Log.d("Fragment 1", "onAttach"); } @Override public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); Log.d("Fragment 1", "onCreate"); } @Override public void onActivityCreated(Bundle savedInstanceState) { super.onActivityCreated(savedInstanceState); Log.d("Fragment 1", "onActivityCreated"); } @Override public void onStart() { super.onStart(); Log.d("Fragment 1", "onStart"); } @Override public void onResume() { super.onResume(); Log.d("Fragment 1", "onResume"); } @Override public void onPause() { super.onPause(); Log.d("Fragment 1", "onPause"); } @Override public void onStop() { super.onStop(); Log.d("Fragment 1", "onStop"); } @Override public void onDestroyView() { super.onDestroyView(); Log.d("Fragment 1", "onDestroyView"); } @Override public void onDestroy() { super.onDestroy(); Log.d("Fragment 1", "onDestroy"); } @Override public void onDetach() { super.onDetach(); Log.d("Fragment 1", "onDetach"); } }</pre>	<pre>package com.metropolitan.Fragments; import android.app.Fragment; import android.os.Bundle; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.Button; import android.widget.TextView; import android.widget.Toast; public class Fragment2 extends Fragment { @Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) { //---kreira izgled fragmenta--- return inflater.inflate(R.layout.fragment2, container, false); } @Override public void onStart() { super.onStart(); //---Pogled dugmeta--- Button btnSetText = (Button) getActivity().findViewById(R.id.btnSetText); btnSetText.setOnClickListener(new View.OnClickListener() { public void onClick(View v) { TextView lbl = (TextView) getActivity().findViewById(R.id.lblFragment1); Toast.makeText(getActivity(), lbl.getText(), Toast.LENGTH_SHORT).show(); } }); } }</pre>

Slika-5 JAVA klase fragmenata

PRIMER KORIŠĆENJA FRAGMENTATA – TESTIRANJE KODA

Prevođenjem program i pokretanjem emulatora testira se kreirana aplikacija.

Klikom na taster F11 aplikacija se debuguje u Android emulatoru nakon čega prikazuje na istom ekranu dve aktivnosti. Interakcijom korisnika sa interfejsom aplikacija demonstrira vlastitu funkcionalnost (sledeća slika).



Slika-6 Demonstracija fragmenata

NAČIN FUNKCIONISANJA FRAGMENTA

Fragment funkcioniše slično kao i aktivnost, definisan je odgovarajućom JAVA klasom, a učitava interfejs iz xml datoteke.

Fragment funkcioniše slično kao i aktivnost, definisan je odgovarajućom JAVA klasom, a učitava interfejs iz xml datoteke. JAVA klasa fragmenta nasleđuje baznu klasu *Fragment*:

```
public class Fragment1 extends Fragment {..}
```

Pored bazne klase, fragment može da nasleđuje i klase koje su izvedene iz klase *Fragment* poput: *DialogFragment*, *ListFragment*, *PreferenceFragment* i sl.

Metodom *onCreateView()* vraća se View objekat, a to je definisano sledećim kodom:

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
    return inflater.inflate (R.layout.fragment2, container, false);  
}
```

Korišćenjem objekta *LayoutInflater* definisan je korisnički interfejs prikazan pomoću xml datoteke. Argument *container* ukazuje na osnovni *ViewGroup* element, a to je aktivnost koja se ugrađuje u fragment. Argument *savedInstanceState* omogućava vraćanje prethodno zapamćenog stanja fragmenta.

IZVRŠAVANJE UGRAĐENIH APLIKACIJA PRIMENOM NAMERA

Korišćenjem Intent objekata moguće je iskoristiti ugrađene aplikacije umesto da kreiramo vlastite za postizanje određene svrhe programa.

Sve što je do sada prikazano oslanjalo se na kreiranje i izvršavanje aktivnosti vlastitih aplikacija. Jedan od najvažnijih segmenata Android programiranja jeste mogućnost upotrebe aktivnosti koje su definisane u drugim aplikacijama. Svaka aplikacija, koja se kreira, ima mogućnost poziva brojnih aplikacija koje su dostupne na konkretnom mobilnom uređaju. U konkretnom slučaju, ukoliko aplikacija zahteva učitavanje određene web stranice, dovoljno je iskoristiti Intent objekat sa ciljem upotrebe postojećeg web čitača, umesto kreiranja novog.

U daljem izlaganju, zadatak će biti na uvođenju konkretnog primera za demonstraciju izvršavanja izvesnih ugrađenih aplikacija koje se u velikoj meri nalaze instalirane na svakom mobilnom uređaju.

PRIMER UPOTREBE INTENT OBJEKATA

Konstruktorom Intent objekta definiše se akcija koju taj objekat treba da obavi.

Zadatak:

1. Koristeći Eclipse IDE kreirati novi Android projekat sa nazivom Namere;
2. Na određeni način izvršiti modifikacije u main.xml datoteci – dodati dugmiće koji će inicirati konkretne akcije nad ugrađenim aplikacijama;
3. Modifikovati JAVA klasu aktivnosti tako što će svaka akcija biti pokrivena odgovarajućim Intent objektom;
4. Prevesti program i pokrenuti da na Android emulatoru.

KOREKCIJE MAIN.XML DATOTEKE

U main.xml datoteci neophodno je dodati kontrole korisničkog interfejsa koje će inicirati pokretanje ugrađenih aplikacija.

U main.xml datoteku biće integrisana tri dugmeta. Prvim će biti pokrenut web čitač, drugo će inicirati telefonski poziv, a treće će zahtevati pozivanje Google mapa. Kod koji to omogućava prikazan je sledećom slikom.

```
<?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" >

    <Button
        android:id="@+id/btn_webbrowser"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Web čitač"
        android:onClick="onClickWebBrowser" />

    <Button
        android:id="@+id/btn_makecalls"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Pozovi"
        android:onClick="onClickMakeCalls" />

    <Button
        android:id="@+id/btn_showMap"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Prikaži mapu"
        android:onClick="onClickShowMap" />

    <Button
        android:id="@+id/btn_launchMyBrowser"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Pokreni čitač"
        android:onClick="onClickLaunchMyBrowser" />
```

Slika-1 main.xml datoteka za demonstraciju poziva ugrađenih aplikacija

JAVA DATOTEKE SA INTENT OBJEKTIMA

U JAVA datotekama definisani su Intent objekti za omogućavanje upotrebe ugrađenih aplikacija.

Prva akcija podrazumeva kreiranje glavne klase aktivnosti.

```
package com.metropolitan.Namere;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;

public class NamereAktivnost extends Activity {
    int request_Code = 1;

    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onClickWebBrowser(View view) {
        Intent i = new Intent("android.intent.action.VIEW");
        i.setData(Uri.parse("http://www.metropolitan.edu.rs/"));
        startActivity(i);
    }

    public void onClickMakeCalls(View view) {
        Intent i = new Intent(android.content.Intent.ACTION_DIAL,
            Uri.parse("tel:+381692030885"));
        startActivity(i);
    }

    public void onClickShowMap(View view) {
        Intent i = new Intent(android.content.Intent.ACTION_VIEW,
            Uri.parse("geo:44.8305392,20.4528576"));
        startActivity(i);
    }

    public void onClickLaunchMyBrowser(View view) {
        Intent i = new Intent(android.content.Intent.ACTION_VIEW,
            Uri.parse("http://www.metropolitan.edu.rs"));
        //i.addCategory("com.metropolitan.DrugaAplikacije");
        //---ova kategorija se ne poklapa ni sa jednim Intent filterom---
        i.addCategory("com.metropolitan.DrugaAplikacije");
        i.addCategory("com.metropolitan.NekeDrugaAplikacije");
        startActivity(Intent.createChooser(i, "Open URL using..."));
    }
}
```

Slika-2 Glavna klasa ativnosti

Potom je neophodno definisati i klasu aktivnosti za web čitač.

```
package com.metropolitan.Aktivnost;

import android.app.Activity;
import android.net.Uri;
import android.os.Bundle;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class CitacAktivnost extends Activity {
    /** Poziva se kreiranjem aktivnosti. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.browser);

        Uri url = getIntent().getData();
        WebView webView = (WebView) findViewById(R.id.WebView01);
        webView.setWebViewClient(new Callback());
        webView.loadUrl(url.toString());
    }

    private class Callback extends WebViewClient {
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
            return(false);
        }
    }
}
```

Slika-3 Klasa aktivnosti web čitača

ANDROIDMANIFEST.XML DATOTEKA

Za svaku Android aplikaciju mora se voditi računa o AndroidManifest.xml datoteci

Sledeći zadatak, vezan za tekući primer, jeste isticanje AndroidManifest.xml datoteke. Aktuelna datoteka prikazana je kodom sa sledeće slike.

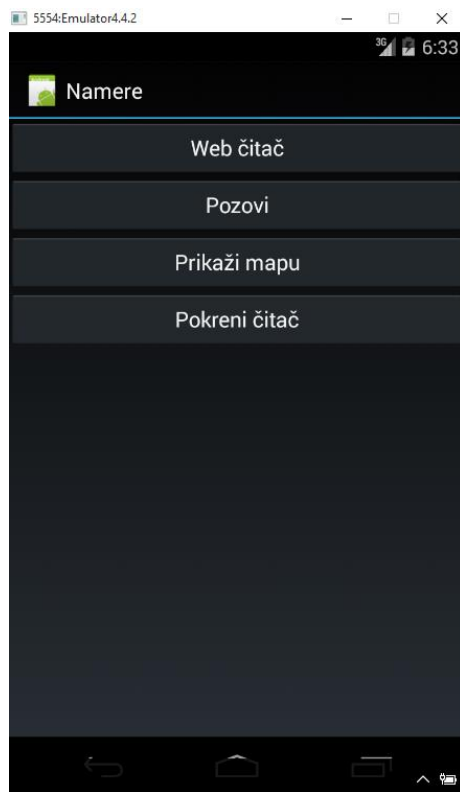
AndroidManifest.xml	
<pre><?xml version="1.0" encoding="utf-8"?> <manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.metroplotan.Namere" android:versionCode="1" android:versionName="1.0" > <uses-sdk android:minSdkVersion="14" /> <uses-permission android:name="android.permission.CALL_PHONE"/> <uses-permission android:name="android.permission.INTERNET"/> <application android:icon="@drawable/ic_launcher" android:label="Namere" > <activity android:label="Namere" android:name=".NamereAktivnost" > <intent-filter > <action android:name="android.intent.action.MAIN" /> <category android:name="android.intent.category.LAUNCHER" /> </intent-filter> </activity> <activity android:name=".CitacAktivnost" android:label="Namere"> <intent-filter> <action android:name="android.intent.action.VIEW" /> <action android:name="com.metroplotan.Citac" /> <category android:name="android.intent.category.DEFAULT" /> <category android:name="com.metroplotan.Aplikacije" /> </intent-filter> </activity> </application> </manifest></pre>	<pre> <category android:name="com.metroplotan.DrugeAplikacije" /> <data android:scheme="http" /> </intent-filter> </activity> <!-- <activity android:name=".CitacAktivnost" android:label="Namere"> <intent-filter> <action android:name="android.intent.action.VIEW" /> <category android:name="android.intent.category.DEFAULT" /> <data android:scheme="http" /> </intent-filter> <intent-filter> <action android:name="com.metroplotan.Citac" /> <category android:name="android.intent.category.DEFAULT" /> <data android:scheme="http" /> </intent-filter> </activity> --> </application> </manifest></pre>

Slika-4 AndroidManifest.xml datoteka primera

DEMONSTRACIJA FUNKCIONALNOSTI PRIMERA

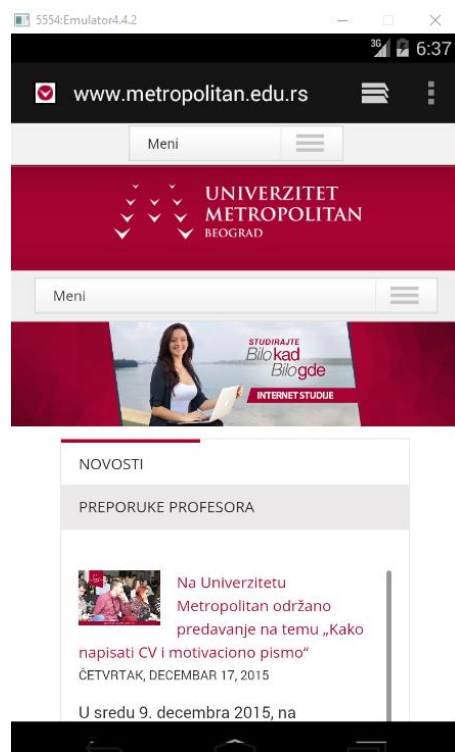
Klikom na F11 vrši se prevođenje i pokretanje primera u emulatoru.

Pokretanjem programa pojavljuje se početna stranica kao na sledećoj slici.



Slika-5 Početni interfejs aplikacije

Izborom prve opcije otvara se ugrađena aplikacija web čitača sa unapred definisanom stranicom.

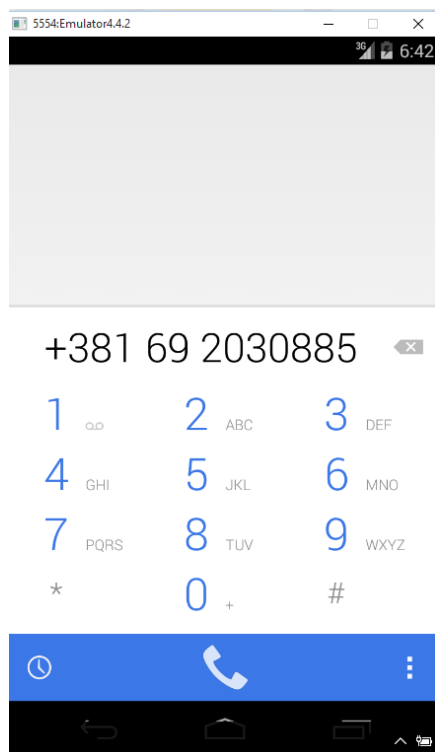


Slika-6 Angažovanje ugrađene aplikacije web čitač

DEMONSTRACIJA FUNKCIONALNOSTI PRIMERA - NASTAVAK

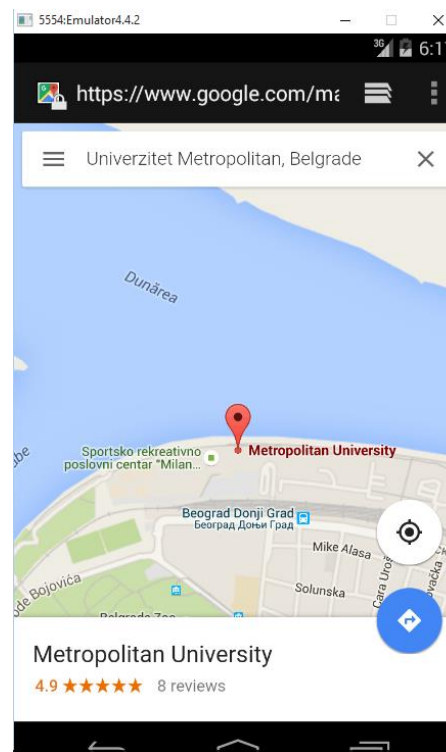
Aplikacije poput Google mapa i telefona takođe mogu biti pozivane.

Izborom opcije *Pozovi* otvara se aplikacija za obavljanje telefonskih poziva.



Slika-7 Pozivanje definisanog broja telefona

Izborom opcije *Prikaži mapu* otvara se web čitač koji pokreće Google mape sa prezentacijom definisane lokacije.



Slika-8 Pozivanje Google mapa

NAČIN FUNKCIONISANJA PROGRAMA ZA POZIVANJE UGRAĐENIH APLIKACIJA

Namere za pozivanje aplikacija čine par: akcija i podaci.

U primeru je demonstrirano korišćenje Intent klase za pozivanje aplikacija koje su sastavni deo Android operativnog sistema skoro svakog mobilnog uređaja.

U Android operativnom sistemu namere kojima se pozivaju ugrađene aplikacije imaju formu para: *akcija i podaci*. Akcijom je definisano kako se nešto izvršava npr. prikazivanje određenog sadržaja, a podaci ukazuju detalje na koje se utiče npr. broj telefona u bazi kontakata i specificirani su kao određeni *Uri* objekti.

Ovo su primeri najčešće korišćenih akcija:

- *ACTION_VIEW*,
- *ACTION_DIAL*;
- *ACTION_PICK*, itd.

U primeru su definisani sledeći podaci (Uri objekti):

- www.metropolitam.edu.rs;
- tel: +381692030885;
- geo:44.8305354,20.4550463, itd.

Akcija i podaci su vrednosti kojima je definisana operacija koja treba da se izvrši. Kao što je pokazano u primeru, da bi neki telefonski broj bio pozvan neophodno je koristiti par *ACTION_DIAL*/tel: +381692030885. Da bi neki kontakt bio izabran iz liste koristi se par *ACTION_PICK/content://contacts*.

Konkretno, u primeru je za prvu komponentu korisničkog interfejsa kreiran Intent objekat pri čemu su u njegov konstruktor prosleđena dva podatka koja odgovaraju akciji i podacima:

Intent i = **new**

```
Intent("android.intent.action.VIEW");
```

```
i.setData(Uri.parse("http://www.metropolitan.edu.rs/"));
```

```
startActivity(i);
```

Navedena aktivnost je ponovljena i za ostale komponente korisničkog interfejsa kojima se pozivaju ugrađene aplikacije.

INTENT OBJEKAT

Intent objekat se kreira prilikom pozivanja aktivnosti ugrađenih aplikacija.

Do sada je naučeno da se pozivanje neke druge aktivnosti obavlja tako što se njene akcije prosleđuju u konstruktor Intent objekta:

```
Intent i = new Intent(android.content.Intent.ACTION_DIAL,  
Uri.parse("tel:+381692030885"));  
startActivity(i);
```

Međutim, podatke je moguće proslediti Intent objektu i upotrebom metode `setData()` na sledeći način:

```
Intent i = new Intent("android.intent.action.VIEW");  
i.setData(Uri.parse("http://www.metropolitan.edu.rs/"));  
startActivity(i);
```

Na ovaj način Android operativnom sistemu je sugerisano da se nastoji pristupiti web stranici koja se nalazi na konkretnoj URL adresi.

Pored akcije i tipa podataka, Intent objekat, a to je pokazano u primeru, može da specificira i *kategoriju* kao skup aktivnosti grupisanih u logičke jedinice koje Android koristi za filtriranje.

Intent objekat može da sadrži sledeće informacije:

- *Action*;
- *Data*;
- *Type*;
- *Category*.

KORIŠĆENJE INTENT FILTERA NA UGRAĐENE APLIKACIJE

Intent filteri, kojima jedna aplikacija inicira drugu, definisani su AndroidManifest.xml datoteci.

U primeru je detaljno pokazano kako jedna aktivnost inicira drugu primenom Intent objekta. Da bi bilo moguće aktiviranje jedne aktivnosti drugom, neophodno je specificirati akciju i kategoriju u okviru xml taga <intent-filter>... </intent-filter> datoteke AndroidManifest.xml. Navedeno je moguće prikazati sledećim programskim kodom:

```
<intent-filter>  
    <action android:name="com.metropolitan.Citac" />  
    <category android:name="android.intent.category.DEFAULT" />  
    <data android:scheme="http" />  
</intent-filter>
```