PT&SCHE

Introduction of part-time and short cycle studies in Serbia

Funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

UNIVERZITET METROPOLITAN BEOGRAD

KI205 - JAVA 8: PROGRAMIRANJE U JAVI NA ANDROID PLATFORMI

Koncepti Android aplikacije – aktivnosti, fragmenti i namere

Lekcija 02

PRIRUČNIK ZA STUDENTE

KI205 - JAVA 8: PROGRAMIRANJE U JAVI NA ANDROID PLATFORMI

Lekcija 02

KONCEPTI ANDROID APLIKACIJE – AKTIVNOSTI, FRAGMENTI I NAMERE

UVOD

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

U prethodnoj lekciji istaknuto je da je aktivnost 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. U ovoj lekciji će poseban akcenat 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.*

Poglavlje 1

Aktivnosti u Android aplikacijama

UPOZNAVANJE KONCEPTA AKTIVNOST

Klasa koja implementira aktivnosti nasleđuje Activity klasu.

Prvi zadatak jeste upoznavanje sa konceptom aktivnosti. Aktivnost predstavlja jedan ekran sa korisničkim interfejsom. Na primer aplikacija telefonkog imenika može imati jednu aktivnost koja prikazuje sve kontakte na sms kartici, druga aktivnost može vršiti pretragu imenika, treća aktivnost može omogućiti prikaz više informacija o izabranom kontaktu i tako dalje. Iako su aktivnosti u okviru jedne aplikacije komponovane i koordinirane tako da predstavljaju logičku celinu, svaka od njih je nezavisna. U skladu sa tim, različite aplikacije mogu da startuju aktivnosti iz drugih aplikacija. Na primer aplikacija za rad kamere, može startovati aktivnost iz Facebook aplikacije kako bi omogućila deljenje fotografija na društvenoj mreži. Svaka aktivnost se implementira kao podklasa klase Activity ili AppCompatActivity (Android 6.0).

PRIMER 1 - GLAVNA KLASA AKTIVNOSTI

Primer pokazuje kreiranje glavne Java klase u android aplikaciji.

Da bi koncept *aktivnost* što bolje bio predstavljen, neophodno je prvo predstaviti JAVA klasu koja nasleđuje osnovnu klasu Activity ili AppCompatActivity (Android 6.0). Neka to bude klasa MainActivity.java iz primera prikazanog u prethodnoj lekciji.

```
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 deklarisana 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. U prethodnoj lekciji 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. Izolovan kod ove metode dat je sledećim listingom.

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

Iz priloženog listinga moguće je primetiti da se pozivom metode ***onCreate()*** realizuje učitavanje aktivnosti glavnog korisničkog interfejsa. Ova aktivnost omogućava korisniku inicijalnu komunikaciju sa aplikacijom i pokretanje novih aktivnosti, ukoliko ih aplikacija poseduje.

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.

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

Slika 1.1.1 Životni ciklus aktivnosti

DODAVANJE AKTIVNOSTI U ACTIVITY DATOTEKU - DEMONSTRACIJA

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 AktivnostiDemo. U ovu svrhu biće otvoreno razvojno okruženje Android Studio IDE i kreiran projekat pod navedenim nazivom. Na lokaciji prikazanoj sledećom slikom, neophodno je otvoriti JAVA datoteku aktivnosti i u nju dodati kod sa slike broj 3.

Slika 1.1.2 Lokacija klase glavne aktivnosti

U JAVA datoteku aktivnosti ugraditi sledeći podvučeni kod:

Slika 1.1.3 Dodavanje koda u klasu aktivnosti

Kod klase aktivnosti u potpunosti je priložen sledećim listingom.

```
package com.metropolitan.aktivnostdemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.util.Log;

public class MainActivity extends AppCompatActivity {

    String s="Životni ciklus aktivnosti";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(s, "Pokrećem onCreate() događaj");
    }
    public void onStart(){
        super.onStart();
        Log.d(s, "Pokrećem onStart() događaj");
    }
    public void onRestart(){
        super.onRestart();
        Log.d(s, "Pokrećem onRestart() događaj");
    }
    public void onResume(){
        super.onResume();
        Log.d(s, "Pokrećem onResume() događaj");
```

```
    }
    public void onPause(){
        super.onPause();
        Log.d(s, "Pokrećem onPause() događaj");
    }
    public void onStop(){
        super.onStop();
        Log.d(s, "Pokrećem onStop() događaj");
    }
    public void onDestroy(){
        super.onDestroy();
        Log.d(s, "Pokrećem onDestroy() događaj");
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.aktivnosti_demostracija, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

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 pomoćuShift +F10 ili klikom na opciju Run app izvrši debagovanje na pokrenutom AVD emulatoru. Prvo što treba proveriti jeste LogCat prozor gde se može primetiti evidencija pokrenutih aktivnosti (sledeća slika).LogCat prozor je dostupan pod opcijom Android monitor razvojnog okruženja Android Studio IDE.

Slika 1.1.4 Evidentiranje pokrenutih aktivnosti u LogCat prozoru

IZVRŠAVANJE AKTIVNOSTI

Klikom na taster Back mobilnog uređaja ili emulatora realizuju se događaji onPause(), onStop() i onDestroy().

Da bi demonstracija izvršavanja aktivnosti bila u potpunosti realizovana i shvaćena na ovom jednostavnom primeru, neophodno je pokazati kako se aplikacija ponaša kada se koriste Android sistemski tasteri. Najbolji primer za to je upotreba i najčešće korišćenog tastera - tastera Back.

Neka demonstracija dalje teče na sledeći način – izvršiti klik na taster Back emulatora i pratiti promene u LogCat prozoru. Moguće je primetiti da je došlo do prokretanja događaja sledećim redom: onPause(), onStop() i onDestroy() što je prikazano sledećom slikom. Dakle, prvim klikom je pauzirana pokrenuta aktivnost aplikacije, nakon čega je ta aktivnost u potpunosti zaustavljena i, konačno, oslobođena iz memorije za neku narednu aktivnost.

Evidencija ovakvog sleda događaja aktivnosti prezentovana je sledećom slikom.

Slika 1.1.5 Aktivnosti nakon klika na Back u emulatoru

IZVRŠAVANJE AKTIVNOSTI - NASTAVAK

Klikom na standardni taster Home takođe se realizuju izvesni događaji.

Neka demonstracija dalje teče na sledeći način – izvršiti klik na ikonu aplikacije u emulatoru i pratiti promene u LogCat prozoru. Moguće je primetiti da je došlo do prokretanja događaja sledećim redom: ***onRestart(), onStart()*** i***onResume(***) što je prikazano sledećom slikom. Dakle, prvim klikom je restartovana aktivnost, pokrenuta ponovo i natavila sa izvršavanjem. To je pokazano sledećom slikom.

Slika 1.1.6 Ponovno pokretanje aplikacije iz emulatora

Na kraju, klikom na sledeće sistemsko dugme Home na emulatoru moguće je primetiti sledeće događaje aktivnosti: ***onPause()*** i ***onStop(),*** a to znači da se istog momenta pauzira i zaustavlja pokrenuta aplikacija.

Slika 1.1.7 Događaji aktivnosti nakon klika na dugme Home

FUNKCIONISANJE AKTIVNOSTI

Aktivnost se eliminiše klikom na taster Back.

Uvek kada se pokrene aktivnost izvršavaju se metode *onStart()* i *onResume()* bez obzira da li je aktivnost nova ili je mirovala u pozadini. Za svako definisanje nove aktivnosti pokreće se metoda *onCreate()*.

Prethodni primeri su pokazali sledeće:

- Metoda ***onCreate()*** se koristi sa ciljem kreiranja objekata koji će se koristiti u aplikaciji;
- Metodom ***onResume()*** startuju se svi servisi ili kod koji se izvršava kada je aktivnost u prvom planu;
- Metodom ***onPause()***zaustavljaju se navedeni servisi ili kod kada aktivnost nije u prvom planu;
- Metodom ***onDestroy()***oslobađaju se resursi pre uklanjanja aktivnosti.

Sledeći video materijal govori o životnom ciklusu Android aktivnosti.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZADATAK 1

Samostalno uključivanje aktivnosti u Android projekat

- Kreirajte nov Android projekat;
- Kreirajte jednostavnu aktivnost određenu željenim GUI;
- U glavnu klasu aktivnosti učitajte XML datoteku sa kreiranim GUI.

1.1 Primer 2 - Primena stilova, tema i dijaloga

UPOTREBA STILOVA I TEMA

Inicijalna podešavanja ativnosti 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 <application ...> datoteke AndroidManifest.xml početna podešavanja se zamenjuju željenim. Sledećim kodom je prikazan deo AndroidManifest.xml datoteke u kojem su izvršene naznačene korekcije: *android:theme="@style/Theme.AppCompat.Dialog"*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.metropolitan.stiloviitemedemo">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
```

```
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Dialog">
        <activity android:name=".Glavna_aktivnost">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Izgled ekrana na kojem je aktivnost oblikovana primenom željene teme, prikazan je sledećom slikom.

Slika 1.2.1 Aktivnost koja ne zauzima ceo ekran

PRIKAZIVANJE OKVIRA - AKCIJE ZA POZIVANJE DIJALOGA

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 *onCreateDilaog()* definisane u osnovnoj klasi aktivnosti. Za ilustraciju biće iskorišćen sledeći primer.

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

```
<?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_dialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Klikni za prikaz dijaloga"
    android:onClick="onClick" />

<Button
    android:id="@+id/btn_dialog2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Klikni za prikaz dijaloga progresa"
    android:onClick="onClick2" />
```

```
<Button
    android:id="@+id/btn_dialog3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Klikni za prikaz dijaloga progresa preuzimanja"
    android:onClick="onClick3" />

</LinearLayout>
```

Iz ove datoteke, koja čuva elemente korisničkog interfejsa, moguće je primetiti da će u linearnom i vertikalnom rasporedu biti postavljena tri dugmeta čijim klikom će biti inicirane akcije za prikaz tri veoma često korišćena dijaloga u Android aplikacijama. Klikom na prvo dugme biće prikazan dijalog sa mogućnošću izbora između nekoliko ponuđenih opcija. Klikom na drugo dugme biće simuliran dijalog progresa. Na kraju, klikom na treće dugme simulira se preuzimanje nekog sadržaja odgovarajućim dijalogom.

PRIKAZIVANJE OKVIRA DIJALOGA U AKTIVNOSTI - PRIPREMA KLASE AKTIVNOSTI

JAVA datoteku, koja je po inicijalnim podešavanjima nazvana DialogActivity.java, redefinisaćemo novim kodom.

Nakon što je kreiran korisnički interfejs, sa kontrolama koje će nakon interakcije sa korisnikom inicirati pokretanje određenog dijaloga i podizanje komunikacije korisnik - mobilna aplikacija na novi nivo, moguće je pristupiti definisanju samih dijaloga koji će se javiti tokom izvršavanja glavne aktivnosti.

JAVA datoteku, koja je po inicijalnim podešavanjima nazvana DialogActivity.java, redefinisaćemo novim kodom prikazanim u sledećem izlaganju. Prvo je neophodno uvesti sve neophodne sistemske klase za rad na ovoj aplikaciji.

```
import android.support.v7.app.AppCompatActivity
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
```

Kao prethodnica ovim ***import*** instrucijama, u kodu se nalazi i naziv paketa kojem aplikacija pripada. Ovo je identično kao i u svim ostalim Java aplikacijama.

```
package com.metropolitan.dialog;
```

U nastavku nalazi se ***onCreate()*** metoda čiji će zadatak biti učitavanje korisničkog interfejsa definisanog main.xml datotekom. Pre navođenja ove metode, nalazi se još malo koda. Ovim kodom će biti ponuđene opcije koje će moći da budu birane u dijalogu, ukoliko se klikne na prvo dugme korisničkog interfejsa aplikacije.

Navedeni deo koda klase aktivnosti prikazan je sledećim listingom.

```
public class DialogActivity extends AppCompatActivity {

CharSequence[] items = { "FIT", "Fakultet za menadžment", "Fakultet digitalnih
umetnosti" };
    boolean[] itemsChecked = new boolean [items.length];

    ProgressDialog progressDialog;
    /** Poziva se prvim definisanjem aktivnosti. */

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

DEFINISANJE METODA ZA AKCIJE KORISNIČKOG INTERFEJSA

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

U main.xml datoteci ovog projekta definisana su tri dugmeta. Klikom na prvo dugme izvršava se metoda onClick() čiji je zadatak učitavanje prvog dijaloga po redu. Izlolovan kod ove metode, prikazan je u sledećem listingu.

```
public void onClick(View v) {
        showDialog(0);
    }
```

Klikom na drugo dugme, po vertikalnom redosledu, izvršava se metoda pod nazivom onClick2(). Ova metoda učitava okvir u kojem se simulira proces koji traje određeni vremeski period, u konkretnom slučaju 5 sekundi. Ovakav dijalog se naziva dijalogom progresa i njegovo izvršavanje obično je praćeno i prisustvom pogodnih komentara koje aplikacija servira korisnicima. Kod ove metode je izolovan u sledećem listingu.

```
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();
    }
```

Na samom kraju, izlaganja u vezi sa metodama akcija dugmadi korisničkog interfejsa, navodi se metoda onClick3(). Ova metoda se izvršava ukoliko korisnik aplikacije klikne na treće dugme u vertikalnom rasporedu. U tom slučaju, aplikacija će u glavnu aktivnost ponovo učitati dijalog progresa koji će, u ovom slučaju, simulirati preuzimanje nekog sadržaja sa mreže. Kod ove metode je prikazan sledećim listingom.

```
public void onClick3(View v) {
        showDialog(1);
        progressDialog.setProgress(0);

        new Thread(new Runnable(){
            public void run(){
                for (int i=1; i<=15; i++) {
                    try {
                        //---simulacija da nešto radi---
                        Thread.sleep(1000);
                        //---osvežavanje dijaloga---
                        progressDialog.incrementProgressBy((int)(100/15));
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                progressDialog.dismiss();
            }
        }).start();
    }
```

METODA ZA KREIRANJE DIJALOGA ONCREATEDIALOG()

Za kreiranje konkretnog dijaloga zadužena je metoda onCreateDialog()

Za kompletiranje klase aktivnosti neophodno je dodati metodu koja će kreirati odgovarajući tip dijaloga i predati ga metodi showDialog() za prikazivanje na osnovi odgovarajuće vrednosti za id. Metoda preuzima prosti it podatak (id), a kao rezultat izvršavanja i obrade vraća objekat tipa Dialog. Metoda je preuzeta iz klase Dialog, njen originalni naziv je onCreateDialog() i u ovom primeru je redefinisana na način prikazan sledećim listingom.

```
@Override
    protected Dialog onCreateDialog(int id) {
        switch (id) {
            case 0:
                return new AlertDialog.Builder(this)
                        .setIcon(R.mipmap.ic_launcher)
                        .setTitle("Dijalog sa malo teksta...")
                        .setPositiveButton("OK",
                                new DialogInterface.OnClickListener() {
```

```
                                    public void onClick(DialogInterface dialog, int
whichButton)
                                    {
                                        Toast.makeText(getBaseContext(),
                                                "OK je kliknut!",
Toast.LENGTH_SHORT).show();
                                    }
                                }
                        )
                        .setNegativeButton("Cancel",
                                new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface dialog, int
whichButton)
                                    {
                                        Toast.makeText(getBaseContext(),
                                                "Cancel je kliknut!",
Toast.LENGTH_SHORT).show();
                                    }
                                }
                        )
                        .setMultiChoiceItems(items, itemsChecked,
                                new DialogInterface.OnMultiChoiceClickListener() {
                                    public void onClick(DialogInterface dialog,
                                                        int which, boolean
isChecked) {
                                        Toast.makeText(getBaseContext(),
                                                items[which] + (isChecked ? "
čekiran!":" nečekiran!"),
                                                Toast.LENGTH_SHORT).show();
                                    }
                                }
                        ).create();

            case 1:
                progressDialog = new ProgressDialog(this);
                progressDialog.setIcon(R.mipmap.ic_launcher);
                progressDialog.setTitle("Preuzimanje datoteka...");
                progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
                progressDialog.setButton(DialogInterface.BUTTON_POSITIVE, "OK",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                                int whichButton)
                            {
                                Toast.makeText(getBaseContext(),
                                        "OK je kliknut!",
Toast.LENGTH_SHORT).show();
                            }
                        });
                progressDialog.setButton(DialogInterface.BUTTON_NEGATIVE, "Cancel",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                                int whichButton)
                            {
```

```
                                Toast.makeText(getBaseContext(),
                                        "Cancel je kliknut!",
Toast.LENGTH_SHORT).show();
                            }
                        });
                return progressDialog;
        }
        return null;
    }
```

FUNKCIONISANJE DIJALOG OKVIRA - DEMONSTRACIJA

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;

U prvom koraku biće pokazan dijalog koji nudi izbor neke opcije korisniku. Pre toga neophodno je izvršiti prevođenje kreiranje aplikacije i omogućiti njeno testiranje na emulatoru - klikom na Runn app u Android Studio IDE okruženju.

Glavna aktivnost je učitana (sledeća slika).

Slika 1.2.2 Učitana glavna aktivnost

Klikom na prvo dugme učitava se okvir koji nudi izbor između ponuđenih opcija korisniku aplikacije. To je pokazano narednom slikom.

Slika 1.2.3 Dijalog ovir sa opcijama

PRIKAZIVANJE OKVIRA ZA DIJALOG PROGRESA

Dijalog progresa je često korišćena komponenta korisničkog interfejsa u Android aplikacijama.

U aplikaciji su demonstrirana dva dijaloga progresa. Prvi simulira zadatak koji bi mogao da se opiše kao preuzimanje sadržaja sa Interneta. Iniciran je klikom na drugo dugme i prate ga komentari "Nešto se dešava.", "Sačekajte..." kao dopunska informacija korisniku. Navedeno je prikazano sledećom slikom.

Slika 1.2.4 Prvi dijalog progresa

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

Drugi dijalog progresa osmišljen je tako da simulira preuzimanje sadržaja iskazano u procentima preuzetog sadržaja sa Interneta. Navedeno je prikazano sledećom slikom.

Slika 1.2.5 Drugi dijalog progresa

ZADATAK 2 - SAMOSTALNO IMPLEMENTIRANJE STILOVE, TEMA I DIJALOGA

Nastavite gde ste stali u zadatku 1

- Otvorite projekat koji ste kreirali u Zadatku 1;
- Prateći izlaganje iz ovog dela lekcije omogućite korišćenje stilova, tema i različitih dijaloga u vašoj aplikaciji.

1.2 Primer 3 - Aplikacije sa više aktivnosti

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. Neophodno je istaći da svaka aktivnost koja se koristi u Android aplikaciji mora da nađe svoje mesto u AndroidManifest.xml datoteci.

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 iTrecaAktivnost.
3. U AndroidManifest.xml datoteci uraditi sledeće dopune, a koje će uključiti kreirane klase aktivnosti:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.metropolitan.namere">
```

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        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:label="Druga aktivnost"
            android:name=".DrugaAktivnost" >
            <intent-filter >
                <action android:name="com.metropolitan.namere.DrugaAktivnost" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>

        <activity
            android:label="Treca aktivnost"
            android:name=".TrecaAktivnost" >
            <intent-filter >
                <action android:name="com.metropolitan.namere.TrecaAktivnost" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

KREIRANJE VLASTITIH XML DATOTEKA ZA AKTIVNOSTI

Svaka aktivnost mora da ima svoju xml datoteku sa odgovarajućim GUI kontrolama.

U res/layout folderu neophodno je kreirati nove dve xml datoteke pod nazivima: DrugaAktivnost.xml i TrecaAktivnost.xml. Ove datoteke će odrediti GUI koji odgovara stranicama koje učitavaju drugu i treću aktivnost koje će biti kreirane. Na samom početku neophodno je definisati XML datoteku za glavnu aktivnost. Njen kog priložen je sledećim listingom.

```
<?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="Prikazi drugu aktivnost"
        android:onClick="onClick"/>

</LinearLayout>
```

Klikom na dugme Prikazi drugu aktivnost otvara se nova aktivnost na ekranu. Metoda koja inicira ovu akciju naziva se onClick().

Učitavanje druge aktivnosti zahteva i učitavanje novog GUI - a koji je određen XML datotekom drugaaktivnost.xml. Datoteka je kreirana u istom folderu u kojem se nalazi datoteka activity_main.xml (slika) desnim klikom, te izborom opcije New layout recource file. Njen kod sledi posle slike.

Slika 1.3.1 Lokacija res/layout

```
<?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="Molimo 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:onClick="onClick"
        android:text="OK" />

</LinearLayout>
```

Na kraju, predviđena je još jedna aktivnost za aplikaciju. Na istoj lokaciji biće kreiranja njoj odgovarajuća xml datoteka čiji je kod dat sledećim listingom.

```
<?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>
```

MODIFIKOVANJE JAVA DATOTEKA AKTIVNOSTI

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

U datoteke DrugaAktivnost.java i TrecaAktivnost.java dodati linije programskog koda koje će učitati GUI definisan prethodno kreiranim xml datotekama i implementirati odgovarajuću programsku logiku.

Svaka od klasa aktivnosti, u navedenom svetlu, mora da poseduje vlastitu onCreate() metodu kojom će učitati odgovarajući GUI iz pripadajuće XML datoteke. Sledećim kodom snabdevena je Java datoteka DrugaAktivnost.java koja odgovara drugoj aktivnosti aplikacije.

```
package com.metropolitan.namere;

/**
 * Created by Vladimir Milicevic on 18.10.2016..
 */

import android.support.v7.app.AppCompatActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class DrugaAktivnost extends AppCompatActivity{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.drugaaktivnost);
    }

    public void onClick(View view) {
        Intent data = new Intent();

        //---pogled EditText  ---
        EditText txt_username =
                (EditText) findViewById(R.id.txt_username);
```

```
        //---podešavanje podataka za prosleđivanje---
        data.setData(Uri.parse(
                txt_username.getText().toString()));
        setResult(RESULT_OK, data);

        //---zatvaranje aktivnosti---
        finish();
    }
}
```

Aplikacija poseduje i treću aktivnost, koja je određena Java klasom TrecaAktivnost.java, koja poseduje vlastitu onCreate() metodu i pripadajuću XML datoteku trecaaktivnost.xml. Sledi kod navedene klase.

```
package com.metropolitan.namere;

/**
 * Created by Vladimir Milicevic on 18.10.2016..
 */
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class TrecaAktivnost extends AppCompatActivity{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.trecaaktivnost);
    }
}
```

MODIFIKACIJA MAINACTIVITY.JAVA DATOTEKE I UGRAĐIVANJE NAMERA

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 glavnu datoteku aktivnosti MainActivity.java. Njen zadatak je, prvenstveno, učitavanje korisničkog interfejsa iz već pokazane datoteke main_activity.xml, a zatim i iniciranje poziva druge aktivnosti nakon čega će toj aktivnosti i ustupiti mesto na ekranu mobilnog uređaja ili emulatora.

Prelazak sa prve aktivnosti na drugu omogućeno je klikom na dugme "*Prikazi drugu aktivnost*" nakon čega se poziva metoda onClick(). Tokom izvršavanja ove metode kreira seIntent objekat (namera) koji ukazuje na aktivnost na koji se prelazi iz glavne aktivnosti (sledeća slika). To je u ovom slučaju druga aktivnost.

Slika 1.3.2 Kreiranje Intent objekta za prelazak na određenu aktivnost

Glavna klasa aktivnosti poseduje još jednu metodu onActivityResult() čiji je zadatak da određene podatke prikaže na ekranu primenom Toast klase.

Kompletan kod glavne klase aktivnosti priložen je sledećim listingom.

```
package com.metropolitan.namere;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Toast;


public class MainActivity extends AppCompatActivity {

    int request_Code = 1;

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

    public void onClick(View view) {


        startActivityForResult(new Intent(
                        "com.metropolitan.namere.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();
            }
        }
    }

}
```

AKTIVNOSTI I FILTERI NAMERA

Aktivnost i filteri moraju da budu deklarisani u AndroidManifest.xml datoteci.

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.namere.DrugaAktivnost* i *com.metropolitan.namere.TrecaAktivnos*t;
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.

Sledi kod datoteke AndroidManifest.xml sa dodatim aktivnostima i filterima namera.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.metropolitan.namere">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        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:label="Druga aktivnost"
            android:name=".DrugaAktivnost" >
            <intent-filter >
                <action android:name="com.metropolitan.namere.DrugaAktivnost" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>

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

```
</manifest>
```

DEMONSTRACIJA ANGAŽOVANJA VIŠE AKTIVNOSTI U APLIKACIJI

Pokretanjem emulatora testira se program za upravljanje aktivnostima.

Pokretanjem programa otvara se početni ekran sa porukom koja ukazuje na način pokretanja druge aktivnosti, a to je klik na odgovarajuće dugme aktivnosti.

Slika 1.3.3 Učitana glavna aktivnost

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

Slika 1.3.4 Učitana druga aktivnost

Popunjavanjem TextField kontrole i klikom na dugme OK realizuje se programska logika definisana za drugu aktivnost. Rezultat izvršavanja prikazan je Toast klasom u donjem delu ekrana mobilnog uređaja (uokvireno crvenom bojom).

Slika 1.3.5 Izvršena logika druge aktivnosti

IZBEGAVANJE SUDARANJA FILTERA NAMERE

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

Kada bi dve aktivnosti, iste aplikacije, imale identičan filter namera, aplikacija bi reagovala pozivom selektora aktivnosti. Ovim selektorom moguće je izabrati aktivnost kojom želi da se nastavi izvršavanje aplikacije. I ne samo to. Moguće je izabrati da li će ta aktivnost biti korišćena samo taj put ili će postati podrazumevana, tj. izvršavaće uvek, u tom slučaju,se sve dok se ne obrišu preferencije aplikacije.

Za demonstraciju ovog problema u kod AndroidManifest.xml datoteke biće dodata modifikacija kojom isti filter namera ukazuje na drugu i trecu aktivnost.

Slika 1.3.6 Podudaranje filtera namera

Pokretanjem ovakve aplikacije, kao što je i naglašeno, selektoru aktivnosti biće prepušten izbor aktivnosti koja će naslediti glavnu aktivnost na ekranu uređaja ili emulatora. Takođe, ponuđena su dva scenarija JUST ONCE i ALWAYS kojim je određeno trajanje ovakvog izbora. Selektor aktivnosti je prikazan sledećom slikom.

Slika 1.3.7 Selektor 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.namere.TrecaAktivnost, izbegava se navedena selekcija i dolazi do direktnog izvršavanja treće aktivnosti. Treća aktivnost će biti izvršena i ako se u selektoru aktivnosti ona izabere.

Slika 1.3.8 Učitana treća aktivnost kao izbor iz selektora ili na osnovu vlastitog filtera namera

VRAĆANJE REZULTATA IZ INICIRANE AKTIVNOSTI

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

Metodastartactivity() inicira drugu aktivnost ali ne vraća rezultat u trenutnu aktivnost. Ako se obrati pažnja na primer, moguće je uočiti da seu 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 glavne aktivnosti startActivityForResult() i metoda onClick() druge aktivnosti , a to je prikazano kodom koji sledi na kraju ovog izlaganja.

Neophodno je navesti i sledeće:

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

startActivityForResult (new Intent(„com.metropolitan.DrugaAktivnost"), request_Code);

- Da bi aktivnost vratila vrednost u polaznu aktivnost, neophodno je koristiti Intent objekat koji koristi metodu setData()za vraćanje podataka nazad.
- Metoda setResult() definiše kod (RESULT_OK ili RESULT_CANCELLED) i Intent objekat koji se vraćaju u polaznu aktivnost.
- Metoda finish() prekida tekuću aktivnost i vraća kontrolu na polaznu aktivnost.
- Polazna aktivnost mora da poseduje metodu onActivityResult() koja se izvršava kada se vraća vrednost iz inicirane aktivnosti.

```
//metoda glavne aktivnosti
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();
            }
        }
```

```
    }
--------------------------------------------------------------------------------
-----------------------
//metoda druge aktivnosti
public void onClick(View view) {
        Intent data = new Intent();

        //---pogled EditText  ---
        EditText txt_username =
                (EditText) findViewById(R.id.txt_username);

        //---podešavanje podataka za prosleđivanje---
        data.setData(Uri.parse(
                txt_username.getText().toString()));
        setResult(RESULT_OK, data);

        //---zatvaranje aktivnosti---
        finish();
    }
```

ZADATAK 3 - DODAJTE NOVU AKTIVNOST U VAŠU APLIKACIJU

Nastavite gde ste stali u zadatku 2

- Pokušajte samostalno da dodate dve nove aktivnosti u vaš projekat;
- Koja aktivnost će biti prikazana, na ekranu mobilnog uređaja, zavisi od izbora koji korisnik vrši na glavnoj aktivnosti;
- Pokušajte da dodate još jednu aktivnost i implementirajte filter aktivnosti za njega i drugu aktivnost.

Poglavlje 2

Fragmenti

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 2.1 Fragmenti i veličina ekrana (izvor tutorialspoint.com)

ŽIVOTNI CIKLUS FRAGMENATA

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 2.2 Životni ciklus fragmenta (izvor tutorialspoint.com)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

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 I TIPOVI FRAGMENATA

Fragmenti su podeljeni u tri nivoa.

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 izglede koje će fragmenti imati.
4. Konačno, fajl aktivnosti će morati da bude modifikovan tako da definiše aktuelnu logiku angažovanja fragmenata.

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 2.3 Tipovi fragmenata

PRIMER 4 - KORIŠĆENJA FRAGMENATA

Na novom Android projektu biće demonstrirana upotreba fragmenata.

Sledećom vežbom biće demonstrirana upotreba fragmenata:

1. Koristeći razvojno okruženje Android Studio IDE kreirati novi projekat i nazvati ga FragmentiDemo.
2. Modifikovati datoteku main.xml i kreirati dve nove fragment1.xml i fragment2.xml;
3. Kreirati dve klaseFragment1.java i Fragment2.java pored glavne klase aktivnosti MainActivity;
4. Prevesti program i pokrenuti emulator za testiranje.
5. Testirati funkcionalnost programa za prezentaciju fragmenata.

U narednom koraku biće definisane datoteke sa kontrolama koje su implementirane na fragmentima. Prvo će biti prezentovan kod datoteke fragment1.xml.

```
<?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>
```

Dalje, dat je kod datoteke fragment2.xml sa odgovarajućim GUI elementima.

```
<?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>
```

PRIMER KORIŠĆENJA FRAGMENATA – XML DATOTEKA GLAVNE AKTIVNOSTI

Program sadrži glavnu xml datoteku i xml datoteke fragmenata.

U activity_main.xml datoteci dodati programski kod koji se prikazan sledećim listingom. Nakon izvođenja ove akcije kompletiran je GUI aplikacije. Sve navedene datoteke moraju da se nalaze u res/layout folderu.

```
<?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.fragmentdemo.Fragment1"
        android:id="@+id/fragment1"
        android:layout_weight="1"
        android:layout_width="0px"
        android:layout_height="match_parent" />
    <fragment
        android:name="com.metropolitan.fragmentdemo.Fragment2"
        android:id="@+id/fragment2"
        android:layout_weight="1"
        android:layout_width="0px"
        android:layout_height="match_parent" />

</LinearLayout>
```

Kao što je moguće primetiti, glavna aktivnost učitava korisnički interfejs koji je podeljen na dva fragmenta tj, na dve podaktivnosti. Svako fragment ima svoj GUI definisan odgovarajućom XML datotekom. U nastavku, neophodno je definisati programsku logiku aplikacije kroz glavnu klasu aktivnosti i klase fragmenata.

PRIMER KORIŠĆENJA FRAGMENATA – JAVA DATOTEKE

Neophodno je dodati i dve java datoteke za fragmente.

U paketu, koji je nazvan com.metropolitan.fragmentdemo, a u folderu src kreirati dve datoteke koje sadrže JAVA klase fragmenata. Datoteke će biti nazvane Fragment1.java i Fragment2.java. Pre ovih klasa definiše se glavna klasa aktivnosti, a njen kod je prezentovan sledećim listingom.

```
package com.metropolitan.fragmentdemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View v) {
        TextView lbl = (TextView)
                findViewById(R.id.lblFragment1);
        Toast.makeText(this, lbl.getText(),
                Toast.LENGTH_SHORT).show();
    }
}
```

Za implementaciju programske logike prvog fragmenta, ugrađenog u glavnu aktivnost, zadužena je klasa Fragment1.java i njen kod je dat u formi sledećeg listinga.

```
package com.metropolitan.fragmentdemo;

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;

/**
 * Created by Vladimir Milicevic on 18.10.2016..
 */

public class Fragment1 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {

        Log.d("Fragment 1", "onCreateView");

        //---Učitava izgled za ovaj fragment---
        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");
    }
}
```

Na kraju, glavna aktivnost poseduje dve podaktivnosti. Kod koji odgovara klasi Fragment2.java sledi kao sledeći listing.

```
package com.metropolitan.fragmentdemo;
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;
/**
 * Created by Vladimir Milicevic on 18.10.2016..
 */
public class Fragment2 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        //---Inflate the layout for this fragment---
        return inflater.inflate(
                R.layout.fragment2, container, false);
    }

    @Override
    public void onStart() {
        super.onStart();
        //---Button view---
        Button btnGetText = (Button)
                getActivity().findViewById(R.id.btnGetText);
        btnGetText.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();
            }
        });
    }

}
```

TESTIRANJE KODA I NAČIN FUNKCIONISANJA FRAGMENATA

Prevođenjem program i pokretanjem emulatora testira se kreirana aplikacija.

Klikom na taster Run app ili Shift + F10 aplikacija se debaguje u Android emulatoru nakon čega prikazuje na istom ekranu dve aktivnosti. Interakcijom korisnika sa interfejsom aplikacija demonstrira vlastitu funkcionalnost (sledeća slika).

Slika 2.4 Fragmenti u glavnoj aktivnosti

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. Argumentcontainer ukazuje na osnovni ViewGroup element, a to je aktivnost koja se ugrađuje u fragment. Argument savedInstanceState omogućava vraćanje prethodno zapamćenog stanja fragmenta.

ZADATAK 4 - DODAJTE FRAGMENTE U VAŠU AKTIVNOST

Nastavite gde ste stali u Zadatku 3.

- Izaberite jednu od kreiranih aktivnosti, iz prethodnog primera, i podelite je na 2 fragmenta;
- Svaki od kreiranih fragmenata prikazuje sadržaj po vašoj želji.

Poglavlje 3

Intent objekti i ugrađene aplikacije

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.

U tom svetlu neka je definisan sledeći zadatak:

1. Koristeći Android Studio IDE kreirati novi Android projekat sa nazivom Namere2;
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. Aktivnosti koje će se obaviti klikom na odgovarajuću dugmad su sledeće:

1. *Prvim će biti pokrenut web čitač,*
2. *Drugo će inicirati telefonski poziv, a*
3. *Treće će zahtevati pozivanje Google mapa.*

Kod koji to omogućava prikazan je sledećim listingom.

```
<?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" />

</LinearLayout>
```

JAVA DATOTEKE SA INTENT OBJEKTIMA

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

Prva akcija podrazumeva kreiranje glavne klase aktivnosti. Glavna klasa aktivnosti će za svako dugme registrovati odgovarajući Intent objekat. Svaki od tih objekata imaće zadatak da pozove neku od ugrađenih aplikacija i da im dodeli odgovarajući zadatak: otvaranje web stranice ili lokacije na Google mapi, pozivanje nekog broja telefona i slično. Glavna klasa aktivnosti nosi naziv MainActivity.java i njen kod je priložen sledećim listingom.

```
package com.metropolitan.namere2;

import android.content.Intent;
```

```
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

public class MainActivity extends AppCompatActivity {
    int request_Code = 1;

    /** Poziva se kada se kreira glavna aktivnost */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_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("https://www.google.rs/
maps/@44.0255012,20.8522824,15z?hl=sr"));


        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("net.learn2develop.Apps");
        //---kategorija se ne podudara ni sa jednim filterom---

        i.addCategory("com.metropolitan.namere2.DrugeAplikacije");
        i.addCategory("com.metropolitan.namere2.NekeDrugeAplikacije");
```

```
        startActivity(Intent.createChooser(i, "Otvaram URL pomoću..."));
    }

}
```

Potom je neophodno definisati i klasu aktivnosti za web čitač.

```
/**
 * Created by Vladimir Milicevic on 18.10.2016..
 */
package com.metropolitan.namere2;

import android.support.v7.app.AppCompatActivity;
import android.net.Uri;
import android.os.Bundle;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class MyBrowserActivity extends AppCompatActivity {
    /** 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);
        }
    }
}
```

DEMONSTRACIJA FUNKCIONALNOSTI PRIMERA

Klikom na Run app ili Shift+F10 vrši se prevođenje i pokretanje primera u emulatoru.

Pokretanjem programa pojavljuje se početna stranica kao na sledećoj slici. Na stranici je organizovan GUI u formi četiri vertikalno poređana dugmeta od kojih je svaki zadužen za pozivanje određene Android ugrađene aplikacije.

Slika 3.1 Početni ekran

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

Slika 3.2 Pozivanje ugrađenog web čitača

DEMONSTRACIJA FUNKCIONALNOSTI PRIMERA - NASTAVAK

Aplikacije poput Google mapa i telefona takođe mogu biti pozivane.

Izborom opcije Pozovi otvara se aplikacija za obavljanje telefonskih poziva. Na ovaj način može se podesiti da se iz aplikacije direktno poziva, na primer, broj centrale našeg Univerziteta.

Slika 3.3 Pozivanje aplikacije za telefoniranje

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

Slika 3.4 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 prosledeiti 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.MyBrowserActivity" />

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

<data android:scheme="http" />

</intent-filter>

Sledeći video materijal govori o namerama u Android aplikacijama.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 5 - NAMERE I UGRAĐENE APLIKACIJE

Vežbanje pozivanja ugrađenig aplikacija pomoću Intent objekata.

Zaokružimo prethodno izlaganje zadatkom:

Kreirati Android aplikaciju koja:

1. otvara sajt našeg Univerziteta klikom na dugme;
2. poziva broj našeg Univeriteta klikom na dugme;
3. pokazuje željenu lokaciju na Google mapi, klikom na dugme.

GUI datoteka je data sledećim kodom:

```
<?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" />

</LinearLayout>
```

Datoteka klase aktivnosti data je sledećim kodom.

```
package com.metropolitan.namere2;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

public class MainActivity extends AppCompatActivity {
    int request_Code = 1;

    /** Poziva se kada se kreira glavna aktivnost */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_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("https://www.google.rs/
maps/@44.0255012,20.8522824,15z?hl=sr"));


        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("net.learn2develop.Apps");
        //---kategorija se ne podudara ni sa jednim filterom---

        i.addCategory("com.metropolitan.namere2.MyBrowserActivity");
        i.addCategory("com.metropolitan.namere2.NekeDrugeAplikacije");

        startActivity(Intent.createChooser(i, "Otvaram URL pomoću..."));
    }

}
```

PRIMER 6 - VEŽBANJE DIALOG OKVIRA - OBNAVLJANJE

Savladavanje primene Dialog okvira u Android aplikacijama.

Zadatak:

1. Kreirati Android aplikaciju koja učitava aktivnost čiji GUI sadrži tri dugmeta raspoređena Linear Layout (Vertical);

2. Klikom na prvo dugme otvara se dijalog okvir koji nudi izbor između tri opcije - opcije su fakulteti našeg Univerziteta; nakon izabrane ili otkazane opcije, na ekranu, Toast klasom prikazati odgovarajuću poruku;
3. Klikom na drugo dugme povezuje se dijalog okvir koji simulira proces koji se izvršava određeni vremenski period (na primer 5000 ms = 5 s) - koristiti klasu Thread.
4. Klikom na treće dugme simulira se preuzimanje nekog sadržaja, na primer sa Interneta, pri čemu je proces preuzimanja iskazan u procentima.
5. Kreirati odgovarajuću datoteku korisničkog interfejsa koja uključuje navedene kontrole.
6. Kreirati glavnu klasu aktivnosti koja će implementirati programsku logiku navedenu u zahtevima.
7. Za AndroidManifest.xml datoteku ostaviti ponuđena podešavanja.
8. Izvršiti prevođenje programa i testiranje na emulatoru.

Nakon kreiranja projekta u paketu: com.metropolitan.dialog, zadaje se naziv datotekama korisničkog interfejsa i glavne java klase: main.xml i DialogActivity.java respektivno.

Kod datoteke main.xml priložen je u sledećem listingu.

```
<?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_dialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Klikni za prikaz dijaloga"
    android:onClick="onClick" />

<Button
    android:id="@+id/btn_dialog2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Klikni za prikaz dijaloga progresa"
    android:onClick="onClick2" />

<Button
    android:id="@+id/btn_dialog3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Klikni za prikaz dijaloga progresa preuzimanja"
    android:onClick="onClick3" />

</LinearLayout>
```

PRIMER 6 - VEŽBANJE DIALOG OKVIRA - GLAVNA KLASA

Primer je zaokružen glavnom klasom.

Realizacija programske logike prepuštena je glavnoj klasi čiji listing sledi. Konačno, pokrenućemo Genymotion emulator, a potom u Android Studiju izborom Run app ili Shift + F10, testirati kreiranu aplikaciju.

```
package com.metropolitan.dialog;

import android.app.AlertDialog;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Toast;

public class DialogActivity extends AppCompatActivity {
    CharSequence[] items = { "FIT", "Fakultet za menadžment", "Fakultet digitalnih
umetnosti" };
    boolean[] itemsChecked = new boolean [items.length];

    ProgressDialog progressDialog;
    /** Poziva se prvim definisanjem aktivnosti. */

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public void onClick(View v) {
        showDialog(0);
    }

    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();
    }

    public void onClick3(View v) {
        showDialog(1);
        progressDialog.setProgress(0);

        new Thread(new Runnable(){
            public void run(){
                for (int i=1; i<=15; i++) {
                    try {
                        //---simulacija da nešto radi---
                        Thread.sleep(1000);
                        //---osvežavanje dijaloga---
                        progressDialog.incrementProgressBy((int)(100/15));
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                progressDialog.dismiss();
            }
        }).start();
    }

    @Override
    protected Dialog onCreateDialog(int id) {
        switch (id) {
            case 0:
                return new AlertDialog.Builder(this)
                        .setIcon(R.mipmap.ic_launcher)
                        .setTitle("Dijalog sa malo teksta...")
                        .setPositiveButton("OK",
                                new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface dialog, int
whichButton)
                                    {
                                        Toast.makeText(getBaseContext(),
                                                "OK je kliknut!",
Toast.LENGTH_SHORT).show();
                                    }
                                }
                        )
                        .setNegativeButton("Cancel",
                                new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface dialog, int
whichButton)
                                    {
                                        Toast.makeText(getBaseContext(),
                                                "Cancel je kliknut!",
Toast.LENGTH_SHORT).show();
                                    }
                                }
```

```
                        )
                        .setMultiChoiceItems(items, itemsChecked,
                                new DialogInterface.OnMultiChoiceClickListener() {
                                    public void onClick(DialogInterface dialog,
                                                        int which, boolean
isChecked) {
                                        Toast.makeText(getBaseContext(),
                                                items[which] + (isChecked ? "
čekiran!":" nečekiran!"),
                                                Toast.LENGTH_SHORT).show();
                                    }
                                }
                        ).create();

            case 1:
                progressDialog = new ProgressDialog(this);
                progressDialog.setIcon(R.mipmap.ic_launcher);
                progressDialog.setTitle("Preuzimanje datoteka...");
                progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
                progressDialog.setButton(DialogInterface.BUTTON_POSITIVE, "OK",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                                int whichButton)
                            {
                                Toast.makeText(getBaseContext(),
                                        "OK je kliknut!",
Toast.LENGTH_SHORT).show();
                            }
                        });
                progressDialog.setButton(DialogInterface.BUTTON_NEGATIVE, "Cancel",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                                int whichButton)
                            {
                                Toast.makeText(getBaseContext(),
                                        "Cancel je kliknut!",
Toast.LENGTH_SHORT).show();
                            }
                        });
                return progressDialog;
        }
        return null;
    }

}
```

PRIMER 7 - KREIRANJE VIŠE AKTIVNOSTI U ISTOJ APLIKACIJI - OBNAVLJANJE

Vežbanje koncepta namera kroz navigaciju kroz aktivnosti u aplikaciji.

Zadatak:

1. Kreirati Android aplikaciju koja uključuje tri aktivnosti;
2. Za svaku od njih kreirati odgovarajući GUI i klase aktivnosti;
3. Intent objektima obezbediti prelak sa jedne aktivnosti na drugu;
4. Testirati sudaranje filtera namera - demonstrirati primenu selektora aktivnosti;
5. Demonstrirati vraćanje preuzete informacije iz druge u prvu aktivnost.

U prvom koraku biće pokazana AndroidManifest.xml datoteka sa <intent-filter > elementima.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.metropolitan.namere">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        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:label="Druga aktivnost"
            android:name=".DrugaAktivnost" >
            <intent-filter >
                <action android:name="com.metropolitan.namere.DrugaAktivnost" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>

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

</manifest>
```

Dalje, u res/layout folderu kreirajte GUI datoteke. Prva od njih je activity_main.xml koja sadrži jedno dugme za prelazak na sledeću aktivnost.

```
<?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="Prikazi drugu aktivnost"
        android:onClick="onClick"/>

</LinearLayout>
```

Dalje, data je xml datoteka drugaaktivnost.xml.

```
<?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="Molimo 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:onClick="onClick"
        android:text="OK" />

</LinearLayout>
```

PIMER 7 - KREIRANJE VIŠE AKTIVNOSTI U ISTOJ APLIKACIJI - NASTAVAK

Treća GUI datoteka i klase aktivnosti.

Sledećim kodom data je datoteka tracaaktivnost.xml.

```
<?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>
```

Na red dolaze klase aktivnosti. Glavna klasa aktivnosti učitava glavnu aktivnosti i prikazuje informaciju Toast klasom preuzetu iz druge aktivnosti.

```
package com.metropolitan.namere;

/**
 * Created by Vladimir Milicevic on 18.10.2016..
 */

import android.support.v7.app.AppCompatActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class DrugaAktivnost extends AppCompatActivity{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.drugaaktivnost);
    }

    public void onClick(View view) {
        Intent data = new Intent();

        //---pogled EditText  ---
        EditText txt_username =
                (EditText) findViewById(R.id.txt_username);

        //---podešavanje podataka za prosleđivanje---
        data.setData(Uri.parse(
                txt_username.getText().toString()));
        setResult(RESULT_OK, data);

        //---zatvaranje aktivnosti---
        finish();
```

```
    }
}
```

```
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Toast;


public class MainActivity extends AppCompatActivity {

    int request_Code = 1;

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

    public void onClick(View view) {


        startActivityForResult(new Intent(
                        "com.metropolitan.namere.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();
            }
        }
    }

}
```

Druga klasa aktivnosti preuzima informaciju iz svoje kontrole tipa EditText i prosleđuje je glavnoj aktivnosti.

```
package com.metropolitan.namere;
import android.support.v7.app.AppCompatActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
```

```
public class DrugaAktivnost extends AppCompatActivity{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.drugaaktivnost);
    }
    public void onClick(View view) {
        Intent data = new Intent();
        //---pogled EditText  ---
        EditText txt_username =
                (EditText) findViewById(R.id.txt_username);
        //---podešavanje podataka za prosleđivanje---
        data.setData(Uri.parse(
                txt_username.getText().toString()));
        setResult(RESULT_OK, data);
        //---zatvaranje aktivnosti---
        finish();
    }
}
```

PRIMER 7 - KREIRANJE VIŠE AKTIVNOSTI U ISTOJ APLIKACIJI - KRAJ

Treća aktivnost i AndroidManifest.xml.

Treća klasa aktivnosti, javlja se kao alternativa drugoj. Ukoliko se podesi intent - filter za učitavanje treće aktivnosti, ona će biti učitana. Takođe, ako se dozvoli isti intent - filter za drugu i treću aktivnost, na scenu će stupiti selektor aktivnosti.

```
package com.metropolitan.namere;

/**
 * Created by Vladimir Milicevic on 18.10.2016..
 */
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class TrecaAktivnost extends AppCompatActivity{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.trecaaktivnost);
    }
}
```

Podešavanje intent - filtera ostavljeno je za kraj. To je u domenu datoteke AndroidManifest.xml. Ovde se razvoj završava i počinje prevođenje i testiranje.

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

```
    package="com.metropolitan.namere">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        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:label="Druga aktivnost"
            android:name=".DrugaAktivnost" >
            <intent-filter >
                <action android:name="com.metropolitan.namere.DrugaAktivnost" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>

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

</manifest>
```

PRIMER 8 - FRAGMENTI U ANDROID APLIKACIJI - OBNAVLJANJE

Vežbanje primene fragmenata kao podaktivnosti.

Zadatak:

1. Kreirati Android aplikaciju koja u okviru glavne aktivnosti uključuje dva fragmenta;
2. Za svaki od fragmenata realizovati GUI datoteku i klasu aktivnosti koja nasleđuje osnovnu klasu fragment;
3. Drugi fragment vraća informaciju u prvi fragment koja se prikazuje Toast klasom;
4. Kreirati GUI datoteku glavne aktivnosti sa fragment komponentama;
5. Kreirati glavnu klasu aktivnosti za realizovanje kompletne logike po zahtevima aplikacije.

Glavan i dve fragment GUI datoteke se kreiraju u res/layout folderu. Sledi fragment1.xml:

```
<?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>
```

Sledećim kodom data je GUI datoteka fragment2.xml.

```
<?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>
```

Konačno, datoteka koja kao GUI elemente uključuje ove fragmente je activity_main.xml.

```
<?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.fragmentdemo.Fragment1"
        android:id="@+id/fragment1"
        android:layout_weight="1"
        android:layout_width="0px"
        android:layout_height="match_parent" />
    <fragment
        android:name="com.metropolitan.fragmentdemo.Fragment2"
        android:id="@+id/fragment2"
        android:layout_weight="1"
        android:layout_width="0px"
        android:layout_height="match_parent" />

</LinearLayout>
```

PRIMER 8 - FRAGMENTI U ANDROID APLIKACIJI - JAVA DATOTEKE

Na red dolaze Fragment klase i glavna klasa aktivnosti.

Fragment1.java preuzima informaciju iz Fragment2.java i ima funkcionalnosti koje dozvoljavaju da se prati njegov životni vek u LogCat prozoru.

```
package com.metropolitan.fragmentdemo;

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;

/**
 * Created by Vladimir Milicevic on 18.10.2016..
 */

public class Fragment1 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {

        Log.d("Fragment 1", "onCreateView");

        //---Učitava izgled za ovaj fragment---
        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");
    }
}
```

Sledi kod klase Fragment2.java.

```
package com.metropolitan.fragmentdemo;
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;
/**
 * Created by Vladimir Milicevic on 18.10.2016..
 */
public class Fragment2 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        //---Inflate the layout for this fragment---
        return inflater.inflate(
                R.layout.fragment2, container, false);
    }

    @Override
    public void onStart() {
        super.onStart();
        //---Button view---
        Button btnGetText = (Button)
                getActivity().findViewById(R.id.btnGetText);
        btnGetText.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();
            }
        });
    }

}
```

Na kraju, glavna klasa aktivnosti je prezentovana sledećim kodom.

```
package com.metropolitan.fragmentdemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View v) {
        TextView lbl = (TextView)
                findViewById(R.id.lblFragment1);
        Toast.makeText(this, lbl.getText(),
                Toast.LENGTH_SHORT).show();
    }
}
```

ZADATAK 5 - DODAJTE NAMERE U VAŠ PROJEKAT

Pokušajte sami!!!

- U vaš projekat, dodajte mogućnost pozivanja neke ugrađene aktivnosti, na primer poziv nekog broja telefona iz sistemske aplikacije.
- Pokrenite i testirajte urađeni Android projekat.

Poglavlje 4

Domaći zadatak 2

DOMAĆI ZADATAK 1 – VEŽBANJE PRIMENE NAMERA U ANDROID APLIKACIJAMA

Kreiranje programa koji koristi aktivnosti

1. Kao osnovu projekta iskoristiti projekat Dialog iz ove lekcije;
2. Koristiti Intent objekte za pozivanje ugrađenih aplikacija;
3. Ugrađena aplikacija koja će biti korišćena je podrazumevani web čitač;
4. Klikom na dugme *Klikni za prikaz dijaloga*, otvara se dijalog okvir sa sledećim opcijama: *FIT; Fakultet za mendžment, Fakultet digitalnih umetnosti*.
5. Izborom opcije *FIT* i klikom na dugme *OK* pokreće se web čitač i otvara sajt *Fakulteta informacionih tehnologija*;
6. Izborom opcije *Fakultet za menadžment* i klikom na dugme *OK* pokreće se web čitač i otvara sajt *ovog fakulteta;*
7. Izborom opcije *Fakultet digitalnih umetnosti* i klikom na dugme *OK* pokreće se web čitač i otvara sajt *ovog fakulteta;*
8. Nastaviti vežbanje primene namera na ugrađene aplikacije koristeći i preostale komponente korisničkog interfejsa iz primera Dialog.

Zaključak

PREGLED LEKCIJE L02

Aktivnosti, fragmenti i namere su ključni koncepti razvoja Android aplikacija.

U ovoj lekciji je detaljno prikazan način funkcionisanja aktivnosti i fragmenata i različitih formi u kojima se oni mogu prikazivati. Za aktivnosti akcenat je bio na: definiciji, životnom ciklusu i njegovom praćenju, ugrađivanju novih aktivnosti, izvršavanju i funkcionisanju aktivnosti. Posebna pažnja je posvećena upotrebi stilova i tema kao načina prilagođavanja inicijalnih podešavanja aktivnosti. Nakon toga prvi put se pominju namere koje imaju ulogu navigatora između pojedinačnih aktivnosti.

U daljoj diskusiji bilo je govora o fragmentima kao specifičnim podaktivnostima. Poseban akcenat je stavljen na: upoznavanje koncepta fragment, životni ciklus fragmenata, upotrebu i tipove fragmenata. Diskusija o fragmentima zaokružena je konkretnim primerom iskazanim u formi Android aplikacije.

Kao treća celina lekcije javlja se diskusija o angažovanju namera prilikom izvršavanja aplikacija koje su već ugrađene u Android uređaj. Kroz konkretan primer pokazano je: kako se kreiraju Intent objekti i kako funckionišu, kako funkcionišu aplikacije koje pozivaju druge Android aplikacije. Posebno, istaknuta je povezanost Intent objekata i filtera u Android aplikacijama

LITERATURA

Za pripremanje lekcije korišćena je sledeća literatura

1. Lee W. M. 2012. *Android 4 – razvoj aplikacija*, Wiley Publishing, INC
2. http://developer.android.com/training/index.html
3. http://www.tutorialspoint.com/android/
4. http://www.vogella.com/tutorials/android.html