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

Android korisnički interfejs

Lekcija 03

PRIRUČNIK ZA STUDENTE

KI205 - JAVA 8: PROGRAMIRANJE U JAVI NA ANDROID PLATFORMI

Lekcija 03

ANDROID KORISNIČKI INTERFEJS

UVOD

Za prikazivanje sadržaja ekrana biće korišćeni pogledi i grupe pogleda.

Do sada je naučeno da aktivnosti predstavljaju alat pomoću kojeg komuniciraju korisnici i aplikacije. Međutim, važno je napomenuti, aktivnost nema vlastitu reprezentaciju na ekranu. Otuda, neophodno je tragati za drugim sredstvima za prikazivanje sadržaja na ekranu. U tu svrhu koristiće se pogledi i grupe pogleda. Dakle, u ovoj lekciji će biti akcenat na načinu kreiranja korisničkog interfejsa i interakcijama korisnika sa aplikacijama. Takođe, biće govora o upravljanju promenama u orijentaciji ekrana na mobilnim uređajima.

Konkretno, u ovoj lekciji će se insistirati na:

- Korišćenju različitih ViewGroup objekata za razmeštanje elemenata korisničkog interfejsa na ekranu;
- Upravljanju promenama u orijentaciji ekrana mobilnog uređaja;
- Programskom kodu za kreiranje korisničkog interfejsa;
- Praćenju obaveštenja koja su u direktnoj vezi sa korisničkim interfejsom.

Nakon savladavanja ove lekcije, studenti će biti u stanju da kreiraju mobilne aplikacije koristeći sve osnovne poglede i grupe pogleda kao elemente korisničkog interfejsa. Takođe, studenti će moći da organizuju GUI komponente po nekom od poznatih GUI rasporeda.

Poglavlje 1

Razumevanje komponenata ekrana

DATOTEKA ACTIVITY_MAIN.XML

Datoteka main.xml nosi u sebi kod koji odgovara elementima korisničkog interfejsa

Osnovna jedinica Android aplikacije naziva se aktivnost. Pomoću nje omogućeno je prikazivanje korisničkog interfejsa aplikacije sa svim kontrolama koje taj interfejs nosi. Za Android aplikacije je karakteristično da se korisnički interfejs definiše na jednom mestu, u xml datoteci koja najčešće nosi naziv main.xml ili activity_main.xml. Ova datoteka je uvek locirana u folderu res/layout i ima sledeći opšti oblik:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.metropolitan.pogledidemo.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Primer jednostavnog GUI!" />
</RelativeLayout>
```

Interfejs definisan ovom datotekom izvršava se pomoću metode onCreate() integrisane u odgovarajuću klasu aktivnosti. Nakon toga koristi se metoda setContentView()klase aktivnosti. Tokom prevođenja Android aplikacije, svaki element main.xml datoteke prevodi se u odgovarajuću GUI klasu. Ovde je ideja, kao i kod većine novijih Java okvira, da se smanji količina Java koda.

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

POGLEDI

Pogled je element koji ima vlastiti prikaz na ekranu.

Aktivnosti Android aplikacija sadrže poglede ili grupe pogleda. Element aplikacije koji ima vlastitu reprezentaciju na ekranu naziva se pogled. Svaki pogled može da sadrži više komponenata, poput: dugmića, labela i tekst polja. Pogledi su definisani u baznoj Android klasi android.view.View.

Pogledi mogu biti grupisani u složene poglede ili grupe pogleda - ViewGroup. Grupa pogleda obezbeđuje način raspoređivanja komponenata korisničkog interfejsa na ekranu. Sve grupe pogleda izvedene su iz bazne Android klase android.view.ViewGroup.

U Android aplikacijama moguće je sresti sledeće grupe pogleda (izgleda):

- LinearLayout;
- AbsoluteLayout;
- TableLayout;
- RealtiveLayout;
- FrameLayout,
- List View, i
- Grid View

U narednim izlaganjima biće govora kako jednostavni pogledi i grupe pogleda mogu da kreiraju lepo dizajniran GUI. Posebno će biti diskutovano o grupama pogleda i njihovoj organizaciji.

ATRIBUTI GRUPE POGLEDA

Svaki pogled ili grupa pogleda imaju određeni skup zajedničkih atributa.

Svaki pogled ili grupa pogleda imaju određeni skup zajedničkih atributa. Zajednički atributi koji se najčešće koriste u pogledima ili grupama pogleda dati su sledećom tabelom.

Slika 1.1 Zajednički atributi različitih pogleda i grupa pogleda

PRIMER 1 - LINEARLAYOUT

LinearLayout je grupa pogleda koja organizuje poglede u jednu kolonu ili vrstu.

Primenom *LinearLayout* izgleda (ili grupe pogleda) poglede je moguće uređivati horizontalno ili vertikalno. Ova grupa pogleda to radi tako što su elementi korisničkog interfejsa organizovani u jednoj vrsti ili koloni na ekranu. Opšti oblik ove grupe pogleda dat je kodom prikazanim sledećim listingom. Na listingu se vidi da je u main.xml datoteci grupa pogleda

definisana xml tagom <LinearLayout> ... </LinearLayout >, a element korisničkog interfejsa za prikazivanje teksta tagom <TextView... />. Naknadno je moguće dodavati ostale poglede, fragmente i tako dalje.

```
<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="Dugme 1"
    android:onClick="onClick1" />

<Button
    android:id="@+id/btn_dialog2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Dugme2"
    android:onClick="onClick2" />

<Button
    android:id="@+id/btn_dialog3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Dugme 3"
    android:onClick="onClick3" />

</LinearLayout>
```

LINEARLAYOUT - HORIZONTALNI I VERTIKALNI RASPORED

Komandama android.orientation = „vertical“ ili android.orientation = „horizontal“ bira se vertikalni ili horizontalni raspored kontrola u LinearLayout grupi pogleda.

Definisanjem datoteke main.xml na takav način da je na njenom početku izabrana grupa pogleda LinearLayout pri čemu je orijentacija grupe pogleda definisana komandom android.orientation = „vertical“, dobija se *podloga* na kojoj se kontrole korisničkog interfejsa pakuju redom i vertikalno.

Slika 1.2 Vertikalni raspored u LinearLayout

Zamenom navedene komande komandom android.orientation = „horizontal“ dobija se horizontalna organizacija komponenata korisničkog interfejsa.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="horizontal" >

<Button
    android:id="@+id/btn_dialog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Dugme 1"
    android:onClick="onClick1" />

<Button
    android:id="@+id/btn_dialog2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Dugme2"
    android:onClick="onClick2" />

<Button
    android:id="@+id/btn_dialog3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Dugme 3"
    android:onClick="onClick3" />

</LinearLayout>
```

Takođe, da bi tri dugmeta mogla horizontalno da budu raspoređena i vidljiva, atributu android:layout_width je dodeljena vrednost "wrap_content" koja ukazuje da kontrole treba da se uklope u raspoloživi prostor na ekranu. U prethodnom slučaju, vrednost "fill_parent" kaže kontroli da zauzme celu širinu ekrana i ostale kontrole ne mogu da se uklope. Nakon ove modifikacije, program se ponovo prevodi i pokreće. Na ekranu mobilnog uređaja sada je sledeća slika.

Slika 1.3 Horizontalni raspored u LinearLayout

PRIMER 2 - ABSOLUTELAYOUT

AbsoluteLayout omogućava određivanje tačne lokacije na kojoj će kontrola korisničkog interfejsa biti pozicionirana na ekranu.

AbsoluteLayout omogućava određivanje tačne lokacije, pomoću x i y koordinata, na kojoj će kontrola korisničkog interfejsa biti pozicionirana na ekranu. Ova grupa pogleda je manje fleksibilna i teža za održavanje nego što je slučaj sa drugim grupama pogleda koje ne koriste apsolutno pozicioniranje.

Slika 1.4 AbsoluteLayout organizacija - grafički prikaz

Definisanjem main.xlm datoteke na sledeći način dobija se korisnički interfejs sa AbsoluteLayout organizacijom komponenata.

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
 >

<Button
    android:id="@+id/btn_dialog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="50px"
    android:layout_y="50px"
    android:text="Dugme 1"
    android:onClick="onClick1" />

<Button
    android:id="@+id/btn_dialog2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="350px"
    android:layout_y="50px"
    android:text="Dugme2"
    android:onClick="onClick2" />

<Button
    android:id="@+id/btn_dialog3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="50px"
    android:layout_y="150px"
    android:text="Dugme 3"
    android:onClick="onClick3" />

</AbsoluteLayout>
```

Pošto su učinjene modifikacije u main.xml datoteci, program se ponovo prevodi i pokreće. Sada su kontrole organizovane AbsoluteLayout grupom pogleda.

Slika 1.5 AbsoluteLayout grupa pogleda

PRIMER 3 - TABLELAYOUT

Komponente korisničkog interfejsa mogu biti organizovane i tabelarno po vrstama i kolonama.

Android grupa pogleda TableLayout organizuje komponente korisničkog intrerfejsa po kolonama i vrstama u formi tabele. Umain.xml datoteci upotrebom taga <TableRow> kreira se nova vrsta u tabelarnom poretku. Svaka vrsta može da sadrži više ćelija, od kojih svaka

može da uključi vlastiti pogled. Sledećim listingom je pokazano kako se main.xml datotekom definiše TableLayoutgrupa pogleda.

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
 >
<TableRow
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
<Button
    android:id="@+id/btn_dialog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="1"
    android:text="Dugme 1"
    android:onClick="onClick1" />

<Button
    android:id="@+id/btn_dialog2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="2"
    android:text="Dugme2"
    android:onClick="onClick2" />
    </TableRow>
    <TableRow>
<Button
    android:id="@+id/btn_dialog3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="1"
    android:text="Dugme 3"
    android:onClick="onClick3" />
    </TableRow>
</TableLayout>
```

TABLELAYOUT - NASTAVAK

TableLayout kontejner ne prikazuje linije između vrsta i kolona koje čine grupu pogleda.

Sledećom slikom grafički je prikazan jedan od mogućih načina organizacije kontrol korisničkog interfejsa primenom TableLayout grupe pogleda.

Slika 1.6 TableLayout - grafički prikaz

Program koji implementira navedenu main.xml datoteku sa TableLayout organizacijom kontrola, na mobilnom uređaju imao bi sledeći reprezentaciju.

Slika 1.7 TableLayout - kontrole na ekranu

PRIMER 4 - REALTIVELAYOUT

RealtiveLayout grupa pogleda pakuje komponente korisničkog interfejsa po relativnom rasporedu.

RealtiveLayout grupa pogleda pakuje komponente korisničkog interfejsa po relativnom rasporedu – gore, dole, levo, desno, primenom sledećih parametara:top, bottom, left, right, center, center_vertical, center_horizontal, alignParentLeft, alignParentRigiht i tako dalje.

Slika 1.8 RealtiveLayout - grafički prikaz

Kao i svaka grupa pogleda, RelativeLayout se definiše main.xml datotekom kao što je to urađeno na sledeći način.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >
<Button
    android:id="@+id/btn_dialog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:text="Dugme 1"
    android:onClick="onClick1" />

<Button
    android:id="@+id/btn_dialog2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/btn_dialog"
    android:text="Dugme2"
    android:onClick="onClick2" />

<Button
    android:id="@+id/btn_dialog3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:text="Dugme 3"
    android:onClick="onClick3" />

</RelativeLayout>
```

Ovako definisanoj main.xml datoteci odgovara sledeći izgled ekrana.

Slika 1.9 RelativeLayout grupa pogleda

PRIMER 5 - FRAMELAYOUT

FrameLayout se koristi sa ciljem izolovanja dela ekrana sa ciljem prikazivanja pojedinačne stavke korisničkog interfejsa.

U osnovi, ova organizacija trebalo bi, kao okvir, da se koristi u svrhu prikazivanja pojedinačnih kontrola iz razloga komplikovanog upravljanja raspoređivanem komponenata iz vizure različitih dimenzija ekrana. Primenom android:layout_gravity atributa moguće, je ipak, dodati više stavki na ekran primenom ove grupe pogleda.

Takođe, i FrameLayout organizacija korisničkog interfeksa definiše se main.xml datotekom kao u sledećem primeru.

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="@color/colorPrimaryDark">
    <ImageView
        android:src="@mipmap/ic_launcher"
        android:scaleType="center"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"
        />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_gravity="center"
        >
<Button
    android:id="@+id/btn_dialog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Dugme 1"
    android:onClick="onClick1" />

<Button
    android:id="@+id/btn_dialog2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Dugme2"
    android:onClick="onClick2" />

<Button
    android:id="@+id/btn_dialog3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Dugme 3"
```

```
    android:onClick="onClick3" />
    </LinearLayout>
</FrameLayout>
```

Pošto su izvršene modifikacije nad originalnom main.xml datotekom, neophodno je ponovo prevesti i pokrenuti kreiranu Android aplikaciju da bi aktivnost bila organizovana FrameLayout grupom pogleda.

Slika 1.10 0 FrameLayout grupa pogleda

PRIMER 6 - LISTVIEW

ListView grupa pogleda organizuje poglede u formi liste.

ListView grupa pogleda organizuje poglede u formi liste. Članovi liste se vertikalno automatski popunjavaju primenom Adapter- a koji predstavlja most između podataka i komponenata interfesja u koje se smeštaju navedeni podaci. ListView se kao kontrola može dodati u neku poznatu grupu pogleda i neka to u konkretnom slučaju bude LinearLayout..

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:id="@+id/lista"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>

</LinearLayout>
```

U nastavku, učitavanje podataka u listu, učitavanje korisničkog interfejsa i prikazivanje glavne aktivnosti, prepušteno je glavnoj klasi aktivnosti.

```
import android.widget.ListView;


public class MainActivity extends AppCompatActivity {
    String [] listaOS =
{"Android","IPhone","WindowsMobile","BlackBarry","WebOS","Ubuntu",
            "Windows7", "Mac OS X"};

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

     //Učitavanje niza predmeta u listu
        ArrayAdapter adapter = new ArrayAdapter <String> (this,
```

```
R.layout.activity_main, listaOS);
        ListView listView = (ListView) findViewById(R.id.lista);
        listView.setAdapter(adapter);
   }
}
```

U ovom slučaju, kreiran je niz stringova čiji su članovi prosleđeni u listu predstavljenu ListView pogledom kreiranim u activity_main.xml datoteci.

Pošto je modifikovana originalna aplikacija, neophodno je ponovo izvršiti njeno prevođenje i testiranje.

Slika 1.11 ListView pogled

GRIDVIEW

GridView prikazuje stavke korisničkog interfejsa u formi matrice.

GridView prikazuje stavke korisničkog interfejsa u formi matrice pri čemu su stavke korisničkog interfejsa automatski raspoređene u grupu pogleda primeom adaptera pod nazivom ListAdapter. GridView grupa pogleda daje organizaciju korisničkog interfejsa kao što je prikazano sledećom slikom.

Slika 1.12 GridView - grafički prikaz

Za razumevanje i primenu, ove grupe pogleda, biće uveden primer čiji će zadatak biti prikazivanje korisničkog interfejsa u formi prikazanoj sledećom slikom.

Slika 1.13 GridView u Android aplikaciji

PRIMER 7 - GRIDVIEW – KLASE I XML PODEŠAVANJA

GridView i ListView su podklase klase AdapterView.

GridView i ListView su podklase klase AdapterView i moguće ih je popuniti vezivanjem za Adapter koji preuzima podatke iz eksternog izvora i kreira pogled kojim je reprezentovan svaki pojedinačni unos. Neka je klasa aktivnosti modifikovana na sledeći način.

```
package com.metropolitan.griddemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
```

```
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    //---Slike za prikazivanje---
    Integer[] imageIDs = {
            R.mipmap.pic1,
            R.mipmap.pic2,
            R.mipmap.pic3,
            R.mipmap.pic4,
            R.mipmap.pic5,
            R.mipmap.pic6,
            R.mipmap.pic7
    };
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        GridView gridView = (GridView) findViewById(R.id.gridview);
        gridView.setAdapter(new ImageAdapter(this));

        gridView.setOnItemClickListener(new OnItemClickListener(){
            public void onItemClick(AdapterView<?> parent,
                                    View v, int position, long id){
                Toast.makeText(getBaseContext(),
                        "Slika " + (position + 1) + " je kliknuta",
                        Toast.LENGTH_SHORT).show();
            }
        });

    }

    public class ImageAdapter extends BaseAdapter{
        private Context context;

        public ImageAdapter(Context c){
            context = c;
        }

        //---vraća broj slika---
        public int getCount() {
            return imageIDs.length;
        }

        //---vraća stavku---
        public Object getItem(int position) {
            return position;
        }
```

```
        //---vraća ID stavke---
        public long getItemId(int position) {
            return position;
        }

        //---vraća ImageView pogled---
        public View getView(int position, View convertView,
                            ViewGroup parent){
            ImageView imageView;
            if (convertView == null) {
                imageView = new ImageView(context);
                imageView.setLayoutParams(new
                        GridView.LayoutParams(85, 85));
                imageView.setScaleType(
                        ImageView.ScaleType.CENTER_CROP);
                imageView.setPadding(5, 5, 5, 5);
            } else {
                imageView = (ImageView) convertView;
            }
            imageView.setImageResource(imageIDs[position]);
            return imageView;
        }
    }

}
```

Biće kreirana nova xml datoteka strings.xml i modifikovanamain.xml za postavljanje GridView pogleda u korisnički interfejs.

```
<?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" >

<GridView
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:columnWidth="90dp"
    android:stretchMode="columnWidth"
    android:gravity="center" />

</LinearLayout>
```

Slike sa detaljima našeg Univerziteta učitavaju se kao niz i predaju se GridView pogledu za prikazivanje. Klikom na svaku od, njih putem Toast klase, realizuje se informacija o broju slike na koju je neko kliknuo.

GRIDVIEW – IMAGEADAPTER

Za dodavanje slika u pogled neophodno je angažovati ImageAdapter.

Na kraju, neophodno je kreirati JAVA klasu kojom se uvodi ImageAdapter u program. Ova klasa je realizovana kao unutrašnja klasa glavne klase aktivnosti, nasleđuje osnovnu klasu BaseAdapter i obavlja neke ključne zadatke u ovom programu:

- vraća broj slika;
- vraća stavku;
- vraća ID stavku;
- vraća ImageView pogled.

Sledećim kodom je izolovana klasa ImageAdapter.

```
public class ImageAdapter extends BaseAdapter{
        private Context context;

        public ImageAdapter(Context c){
            context = c;
        }

        //---vraća broj slika---
        public int getCount() {
            return imageIDs.length;
        }

        //---vraća stavku---
        public Object getItem(int position) {
            return position;
        }

        //---vraća ID stavke---
        public long getItemId(int position) {
            return position;
        }

        //---vraća ImageView pogled---
        public View getView(int position, View convertView,
                            ViewGroup parent){
            ImageView imageView;
            if (convertView == null) {
                imageView = new ImageView(context);
                imageView.setLayoutParams(new
                        GridView.LayoutParams(85, 85));
                imageView.setScaleType(
                        ImageView.ScaleType.CENTER_CROP);
                imageView.setPadding(5, 5, 5, 5);
            } else {
                imageView = (ImageView) convertView;
            }
```

```
            imageView.setImageResource(imageIDs[position]);
            return imageView;
        }
    }
```

Pokretanjem aplikacije u Genymotion emulatoru, dobija se sledeći ekran sa prikazanim funkcionalnostima.

Slika 1.14 Testiranje GridDemo programa

VIDEO MATERIJALI

Rezime pogleda i grupa pogleda

Recap on Views and ViewGroups - Developing Android Apps (youtube.com)

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

Views and View Groups - Android Studio (youtube.com)

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

ZADATAK 1 - KREIRAJTE GUI SA PREDEFINISANOM GRUPOM POGLEDA

Pokušajte sami

- Kreirajte nov Android projekat;
- Dodajte kontrole kao da pravite stranicu za prijavljivanje: dve labele - user name i password i dva polja za unos teksta;
- Organizujte ove kontrole koristeći GridView;
- Pokušajte da ih organizujete na neki drugi način.

Poglavlje 2

Prikazi i orijentacija ekrana

ORIJENTACIJA EKRANA MOBILNOG UREĐAJA

Android operativni sistem podržava dve orijentacije ekrana: portrait i landscape.

Jedna od osnovnih karakteristika mobilnih uređaja, koji rade pod Android operativnim sistemom, jeste mogućnost menjanja orijentacije ekrana. Android operativni sistem podržava dve orijentacije ekrana: uspravno (portrait) i horizontalno (landscape). Osnovnim podešavanjima, prilikom promene orijentacije ekrana Android uređaja, ponovo se učitava trenutna aktivnost koja se povezuje sa sadržajem koji je prikazan u novoj orijentaciji. Dakle, svaki put kada se orijentacija ekrana promeni, ponovo se poziva metoda onCreate() za iniciranje aktivnosti. To zapravo znači, da će, prilikom promene, orijentacije ekrana, tekuća aktivnost biti uklonjena, a na njenom mestu biće kreirana nova aktivnost.

Prilikom novog prikazivanja pogleda, elementi korisničkog interfejsa se ponovo iscrtavaju na svojim originalnim pozicijama. To praktično znači da prelaskom sa uspravne na horizontalnu orijentaciju ekrana, na novom, širem ekranu, može ostati izvesna količina slobodnog i neiskorišćenog prostora.

Da bi navedeni nedostaci bili otklonjeni, moguće je koristiti sledeće tehnike upravljanja promenama orijentacije ekrana:

- *Anchoring* (usidrenje) - Ovom tehnikom prikaz se fiksira za četiri ivice ekrana;
- *Resizing* (promena veličine) i *repositioning* (promena položaja) – Napredna tehnika promene veličine svakog pojedinačnog prikaza elemenata u zavisnosti od trenutne orijentacije ekrana mobilnog uređaja.

PRIMER 8 - USIDRENJE

Primenom RelativeLayout prikaza moguće je jednostavno usidriti elemente ekrana.

Primenom RelativeLayout prikaza moguće je jednostavno usidriti elemente ekrana. Elementi korisničkog interfejsa na veoma jednostavan i elegantan način, primenom sledećih atributa, vezuju se za četiri ivice ekrana:

- layout_alignParentLeft – Uređuje pogled u odnosu na levu stranu osnovnog pogleda;
- layout_alignParentRight – Uređuje pogled u odnosu na desnu stranu osnovnog pogleda;
- layout_alignParentTop - Uređuje pogled u odnosu na gornju ivicu osnovnog pogleda;

- layout_alignParentBottom - Uređuje pogled u odnosu na donju ivicu osnovnog pogleda;
- layout_centerVertical - Centrira pogled po vertikali u odnosu osnovni pogled;
- layout_centerHorizontal - Centrira pogled horizontalno u odnosu osnovni pogled;

Primena ove tehnike biće demonstrirana na sledećem primeru. Neka je data sledeća main.xml datoteka sa RelativeLayout rasporedom za pet pogleda koji odgovaraju kontrolama *dugmići.*

U osnovnu main.xml datoteku ubačen je sledeći kod.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.metropolitan.pogledi.MainActivity" >

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Left"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Right"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Left"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"
        />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Right"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        />
    <Button
```

```
        android:id="@+id/button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Centrirano"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        />

</RelativeLayout>
```

USIDRENJE – PROMENA ORIJENTACIJE AVD

Usidrenjem kreirane kontrole se pričvršćuju za ivice i središte ekrana.

Upotrebom navedenih atributa, a prilikom promene orijentacije ekrana, odgovarajuće kontrole biće raspoređene po ivicama ekrana ili centrirane. Sledećom slikom je prikazana originalna aktivnost koja je učitana sa prvi pokretanjem aplikacije.

Slika 2.1 Vertikalna (portrait) orijentacija ekrana

Budući da je primenjena tehnika usidrenje, promenom orijentacije ekrana u horizontalnu (landscape), dugmići će ostati na svojim mestima jer su fiksirani za ivice ekrana.

Slika 2.2 Horizontalna (landscape) orijentacija ekrana

PRIMER 9 - PROMENA VELIČINE I POLOŽAJA

Upravljanje promenama ekrana moguće je kreiranjem posebnih xml datoteka za svaku pojedinačnu orijentaciju.

Promena veličine i položaja predstavlja napredniju tehniku upravljanja promenama u orijentaciji ekrana. Tehnika podrazumeva kreiranje posebnog res/layout foldera koji sadrži xml datoteke pojedinačnih orijentacija ekrana.

Neka podrazumevani res/layout folder sadrži activity_main.xml datoteku koja odgovara vertikalnoj orijentciji ekrana. Ukoliko je cilj obezbeđivanje mogućnosti rada uređaja u horizontalnom režimu, neophodno je kreirati novi folder koji može da dobije naziv res/layout-land. Takođe, ovaj folder će biti snabdeven odgovarajućom activity_main.xml datotekom.

Slika 2.3 Layout folderi za orijentacije

Da bi problem što bolje bio identifikovan i prikazan, za horizontalni raspored biće proširena odgovarajuća activity_main.xml datoteka. U ovu datoteku će biti dodata još dva dugmeta, a to je prikazano sledećim kodom.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.metropolitan.usidrenjedemo.MainActivity" >

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Left"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Top Right"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Left"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"
        />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Right"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        />
    <Button
        android:id="@+id/button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Centar"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        />
    <Button
        android:id="@+id/button6"
        android:layout_width="180px"
```

```
        android:layout_height="wrap_content"
        android:text="Centar gore"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:layout_alignParentTop="true"
        />
    <Button
        android:id="@+id/button7"
        android:layout_width="180px"
        android:layout_height="wrap_content"
        android:text="Centar dole"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        />

</RelativeLayout>
```

PROMENA VELIČINE I POLOŽAJA – PROMENA ORIJENTACIJE EKRANA

Za svaku orijentaciju Android automatski aktivira odgovarajući xml dokument.

Nakon što je novom res/layout-land folderu kreirana main.xml datoteku koja se od main.xml datoteke iz res/layout foldera razlikuje po dodatim linijama xml koda , moguće je pristupiti testiranju ovako modifikovanog programa.

Učitavanje aktivnosti u vertikalnom režimu već je prikazano i uočeno je da je došlo do prikazivanja pet dugmića. Okretanjem telefona, ili izborom opcije rotate za emulator, dolazi do promene u orijentaciji ekrana u horizontalni raspored. Upravo tada, u konkretnom primeru, interfejs će biti učitan iz novog xml fajla i biće prikazano sedam dugmića. Navedeno je prikazano sledećom slikom.

Slika 2.4 Učitavanje GUI iz res/layout-land foldera

VIDEO MATERIJALI

Rezime orijentacije ekrana

Android Activity Screen Orientation Part 1 | coursetro.com (youtube.com)

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

Android Activity Screen Orientation Part 2 | coursetro.com (youtube.com)

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

ZADATAK 2 - PRIMENITE PROMENU ORIJENTACIJE EKRANA

Pokušajte sami

- Posmatrajte prethodni zadatak za promenu orijentacije ekrana;
- Da li bi trebalo da uradite neku modifikaciju nad Zadatkom 1?

Poglavlje 3

Primer 10 - Upravljanje orijentacijom ekrana

AKTIVNOSTI I PROMENA ORIJENTACIJE EKRANA

Promena orijentacije ekrana izaziva prekid trenutne aktivnosti.

Budući da su pokazane dve tehnike za upravljanje prikazom sadržaja na ekranu mobilnog uređaja, prilikom promene njegove orijentacije, neophodno je još izvršiti analizu stanja aktivnosti. Stanje aktivnosti će najbolje biti ispraćeno na monitoru, odnosno LogCat prozoru. Zato će biti demonstriran primer čija klasa aktivnosti prosleđuje odgovarajuće monitoring informacije uLogCat prozor.

Neka je GUI ove aplikacije određen sledećim xml kodom datoteke 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="vertical" >

    <EditText
        android:id="@+id/txtField1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <ToggleButton android:id="@+id/toggle1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Glavna klasa aktivnosti data je sledećim programskim kodom.

```
package com.metropolitan.orijentacijedemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Display;
```

```
import android.view.WindowManager;

public class MainActivity extends AppCompatActivity {

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

        WindowManager wm = getWindowManager();
        Display d = wm.getDefaultDisplay();

        if (d.getWidth() > d.getHeight()) {
            //---prelazak u landscape mode---
            Log.d("Orientation", "Landscape mode");
        }
        else {
            //--- prelazak u portrait mode---
            Log.d("Orientation", "Portrait mode");
        }
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        //---snimanje svega što želite da sačuvate---
        outState.putString("ID", "1234567890");
        super.onSaveInstanceState(outState);
    }

    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        //---vraćanje ranije sačuvane informacije---
        String ID = savedInstanceState.getString("ID");
    }

    @Override
    public Object onRetainNonConfigurationInstance(){
        return ("Čuvanje nekog teksta!!!");
    }
    @Override
    public void onStart() {
        Log.d("StateInfo", "onStart");
        super.onStart();
    }

    @Override
    public void onResume() {
        Log.d("StateInfo", "onResume");
        super.onResume();
    }
```

```
    @Override
    public void onPause() {
        Log.d("StateInfo", "onPause");
        super.onPause();
    }

    @Override
    public void onStop() {
        Log.d("StateInfo", "onStop");
        super.onStop();
    }

    @Override
    public void onDestroy() {
        Log.d("StateInfo", "onDestroy");
        super.onDestroy();
    }

    @Override
    public void onRestart() {
        Log.d("StateInfo", "onRestart");
        super.onRestart();
    }

}
```

PREVOĐENJE PRIMERA I PRAĆENJE AKTIVNOSTI

Uvidom u LogCat prozor moguće je uočiti da se aktivnost uklanja kada uređaj menja orijentaciju.

Prevođenjem programa i njegovim pokretanjem u emulatoru dobija se vertikalni prikaz korisničkog interfejs. Izborom opcije *Rotate* Screen, iz menija u donjem desnom uglu emulatora, ili klikom na Ctrl + F11 (Slika 1), vrži se rotiranje ekrana i tada je u LogCat prozoru moguće pratiti promene u aktivnosti (Slika 2).

Slika 3.1 Izbor opcije za rotiranje ekrana u Genimotion emulatoru

Moguće je primetiti da je promenom orijentacije došlo do potpunog gašenja inicijalne aktivnosti i pokretanja nove. Ovo je veoma važno da se zna u slučaju da postoji potreba za čuvanjem određenih podataka koji bi promenom orijentacije bili izgubljeni. U tom slučaju je neophodno sačuvati stanje aktivnosti pomoću metode *onPause()* koja se uvek izvršava, za određenu aktivnost, kada dolazi do promene orijentacije ekrana.

Još je neophodno napomenuti da se samo pogledi u kojima su nazivi definisani pomoću android:id atributa, u određenoj aktivnosti, ne mogu eliminisati kreiranjem nove aktivnosti prilikom promene orijentacije ekrana. Upravo su, da bi navedeno bilo demonstrirano, kreirana dva različita tekst polja u main.xml datoteci primera.

Slika 3.2 Praćenje promena aktivnosti nakon rotiranja ekrana

POKRETANJE PRIMERA I PRAĆENJE PERZISTENTNOSTI PODATAKA

Pogled koji odgovara kontroli čiji je naziv definisan android:id atributom biće učitan sa promenom orijentacije uređaja.

Sledećom slikom dat je emulator sa vertikalnom orijentacijom ekrana i sa pokrenutim primerom. Kao što je moguće primetiti Popunjena su oba polja za unos teksta. Prikazana je i nova kontrola ToggleButton koja može da ima dva stanja on/off.

Slika 3.3 Vertikalni raspored u emulatoru

Promenom orijentacije ekrana uočava se gubljene podataka iz drugog tekst polja prilikom učitavanja nove aktivnosti. U drugom tekst polju ne postoji podatak vezan za *android.id* atribut. Sve kontrole koje nemaju id atribut, nakon rotiranja ekrana i učitavanja nove aktivnosti, ne mogu da sačuvaju podatke koji su u njih uneti.

Slika 3.4 Horizontalni raspored u emulatoru

ČUVANJE INFORMACIJA PRILIKOM PROMENE ORIJENTACIJE EKRANA

Da bi određene aktivnosti bile sačuvane moguće je uvek implementirati metodu onPause().

Prilikom gašenja aktivnosti dolazi do iniciranja jedne ili obe sledeće metode:

- onPause() – Metoda se uvek inicira kada se aktivnost ukloni ili prosledi u pozadinu;
- onSaveInstanceState()– Radi isto što i prethodna metoda sa razlikom da se ne inicira kada se aktivnost ukloni sa steka (pritiskom korisnika na taster *Back*) iz razloga što ne postoji potreba za ponovnim korišćenjem stanja aktvivnosti.

Da bi određene aktivnosti bile sačuvane moguće je uvek implementirati metodu onPause(), a zatim koristiti neke do načina za čuvanje podataka npr. baze podataka, pomoćne datoteke i sl.

Mnogo jednostavniji način za čuvanje podataka neke aktivnosti je primena metode onSaveInstanceState().Ova metoda koristi *Bundle* objekat koji može da se iskoristi za snimane stanja tekuće aktivnosti.

Nakon što se ponovo kreira aktivnost, onCreate() metodom, prethodno snimljeno stanje učitava se metodom onRestoreInstanceState(). *Metode su date sledećim kodom.*

U primeru postoji i metoda onReatainNonConfigurationInstance()koja se inicira kada se aktivnost uklanja usled konfiguracionih promena, a to su: orijentacija ekrana, promena tastature i sl. Sledećim kodom biće upravo izolovane metode onSaveInstanceState() i onRestoreInstanceState().

```
@Override
    public void onSaveInstanceState(Bundle outState) {
        //---snimanje svega što želite da sačuvate---
        outState.putString("ID", "1234567890");
        super.onSaveInstanceState(outState);
    }

    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        //---vraćanje ranije sačuvane informacije---
        String ID = savedInstanceState.getString("ID");
    }
```

PROMENA ORIJENTACIJE EKRANA

Utvrđivanje trenutne orijentacije ekrana moguće je obaviti pomoću WindowsManager klase.

Nekada je, tokom izvršavanja programa, neophodno imati informaciju o aktuelnoj orijentaciji ekrana. Ovu informaciju je moguće dobiti pomoću WindowsManager klase. U tekućem primeru je uključen programski kod kojim je realizovana programska detekcija trenutne orijentacije ekrana koja se odnosi na trenutnu aktivnost. Posebnu pažnju trebalo bi obratiti na metodu getDefaultDisplay()koja vraća objekat tipa Display koji predstavlja ekran posmatranog mobilnog uređaja. Odavde je moguće dobiti informacije o ekranu, poput širine i visine, kao i njegove trenutne orijentacije.

Navedeno je, u programu, realizovano sledećim delom glavne klase aktivnosti.

```
public class MainActivity extends AppCompatActivity {

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

        WindowManager wm = getWindowManager();
        Display d = wm.getDefaultDisplay();

        if (d.getWidth() > d.getHeight()) {
            //---prelazak u landscape mode---
            Log.d("Orientation", "Landscape mode");
        }
```

```
        else {
            //--- prelazak u portrait mode---
            Log.d("Orientation", "Portrait mode");
        }
    }
. . .
```

KONTROLA ORIJENTACIJE

Programsko sprečavanje promene orijentacije ekrana obavlja se metodom setRequestOrientation() klase aktivnosti.

Za određene programe neophodno je fiksirati orijentaciju ekrana. Primer može biti igrica koja se izvršava isključivo u horizontalnom režimu. U ovom slučaju, programsko sprečavanje promene orijentacije ekrana obavlja se metodom setRequestOrientation() klase aktivnosti. Navedeno je prikazano sledećim kodom.

```
public class MainActivity extends AppCompatActivity {

    /** poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setRequestOrientation  (ActivityInfo.SREEN_ORIENTATION_LANDSCAPE);
    }
```

Fiksiranje u vertikalni raspored izvodi se instrukcijom setRequestOrientation() za konstantuActivityInfo.SCREEN_ORIENTATION_PORTRAIT.

Ovu akciju je moguće izvesti u AndroidManifest datoteci u okviru elementa <activity> na sledeći način: android:screenOrientation="landscape". Atribut za definisanje orijentacije ekrana može imati sledeće vrednosti: landscape, portrait i sensor(default vrednost zasnovana na senzoru).

Sledećim listingom je prikazano definisanje orijentacije ekrana u okviru datoteke AndroidManifest.xml. U liniji koda br 13 izvršeno je fiksiranje orijentacije ekrana na horizontalnu.

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

    <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"
            android:screenOrientation="landscape">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

ZADATAK 3 - OBJASNITE SLEDEĆE INTRUKCIJE U VEZI ZA PROMENOM ORIJENTACIJE EKRANA

Pokušajte sami

1. Objasnite ulogu sledeće grupe instrukcija?

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

        WindowManager wm = getWindowManager();
        Display d = wm.getDefaultDisplay();

        if (d.getWidth() > d.getHeight()) {

            Log.d("Orientation", "Landscape mode");
        }
        else {

            Log.d("Orientation", "Portrait mode");
        }
    }
```

2. Objasnite ulogu sledeće grupe instrukcija?

```
@Override
    public void onSaveInstanceState(Bundle outState) {

        outState.putString("ID", "1234567890");
        super.onSaveInstanceState(outState);
    }

    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
```

```
        String ID = savedInstanceState.getString("ID");
    }
```

Poglavlje 4

Upotreba funkcije ActionBar

PREDSTAVLJANJE FUNKCIJE ACTIONBAR

ActionBar prikazuje ikonu aplikacije i naziv aktivntosti.

Jedna od funkcija koja je uvedena u novije verzije Android operativnog sistema jeste *ActionBar*. Ovom funkcijom zamenjena je tradicionalna naslovna linija koja se nalazila u gornjem delu ekrana i ona prikazuje ikonu aplikacije i naziv aktivnosti. Desno od ove funkcije, opciono, moguće je naći i određene ActionBar stavke. U Android aplikacijama ActionBar može biti prikazan ili sakriven i upravo će biti akcenat na načinima prikazivanja i skrivanja ove funkcije.

Slika 4.1 ActionBar sa dodatnim stavkama

Upravljanje ActionBar funkcijom biće prikazano na sledećem primeru:

- Kreirati nov Android projekat MojActionBarDemo;
- Dodavanje i skrivanje funkcije definisati datotekom AndroidManifest.xml;

- Modifikovati klasu *MojActionBar.java.*
- Izvršiti prevođenje i pokretanje programa za demonstraciju emulatorom.
- Datoteka main.xml će nositi podatak o nazivu našeg Univeriteta koji će biti prikazan na ekranu.

Sledećim kodom prikazana je AndroidManifest.xml datoteka sa ActionBar podrškom (linija koda 10). Opciono, umetanjem sledeće instukcije u blok <activity> ActionBar može biti skiven:

android:theme="@style/Theme.AppCompat.Light.NoActionBar"

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.DarkActionBar">

        <activity android:name=".MainActivity">
```

```
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

GLAVNA JAVA DATOTEKA AKTIVNOSTI I GUI DATOTEKA

Dodavanje stavki akcija dešava se u klasi aktivnosti.

Na osnovu prethodno definisanih zahteva razvija se GUI aplikacije. U tu svrhu je neophodno kreirati odgovarajuću activity_main.xml datoteku i snabdeti je sledećim XML 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" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Univerzitet Metropolitan Beograd" />

</LinearLayout>
```

Glavna klasa aktivnosti nosi podatke o načinu funkcionisanja glavne aktivnosti ali i o stavkama koje se dodaju u ActionBar komponentu. Takođe, u glavnoj klasi aktivnosti je moguće na dinamički način sakriti ActionBar dodavanjem instrukcije *actionBar.hide().* Alternativno, isključeni ActionBar je moguće ponovo prikazati instrukcijom *actionBar.show().*

Sledećim listingom prikazana je glavna klasa aktivnosti sa funkcionalnostima koje su istaknute u prethodno navedenim zahtevima.

```
package com.metropolitan.actionbardemo;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
```

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

    ActionBar actionBar = getSupportActionBar();
    actionBar.setDisplayHomeAsUpEnabled(true);

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    CreateMenu(menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    return MenuChoice(item);
}

private void CreateMenu(Menu menu)
{
    MenuItem mnu1 = menu.add(0, 0, 0, "Stavka 1");
    {
        mnu1.setIcon(R.mipmap.ic_launcher);
        mnu1.setShowAsAction(
                MenuItem.SHOW_AS_ACTION_IF_ROOM |
                        MenuItem.SHOW_AS_ACTION_WITH_TEXT);
    }
    MenuItem mnu2 = menu.add(0, 1, 1, "Stavka 2");
    {
        mnu2.setIcon(R.mipmap.ic_launcher);
        mnu2.setShowAsAction(
                MenuItem.SHOW_AS_ACTION_IF_ROOM |
                        MenuItem.SHOW_AS_ACTION_WITH_TEXT);
    }
    MenuItem mnu3 = menu.add(0, 2, 2, "Stavka 3");
    {
        mnu3.setIcon(R.mipmap.ic_launcher);
        mnu3.setShowAsAction(
                MenuItem.SHOW_AS_ACTION_IF_ROOM |
                        MenuItem.SHOW_AS_ACTION_WITH_TEXT);
    }
    MenuItem mnu4 = menu.add(0, 3, 3, "Stavka 4");
    {
        mnu4.setShowAsAction(
                MenuItem.SHOW_AS_ACTION_IF_ROOM |
                        MenuItem.SHOW_AS_ACTION_WITH_TEXT);
```

```
        }
        MenuItem mnu5 = menu.add(0, 4, 4, "Stavka 5");
        {
            mnu5.setShowAsAction(
                    MenuItem.SHOW_AS_ACTION_IF_ROOM |
                            MenuItem.SHOW_AS_ACTION_WITH_TEXT);
        }
    }

    private boolean MenuChoice(MenuItem item)
    {
        switch (item.getItemId()) {
            case  android.R.id.home:
                Toast.makeText(this,
                        "Kliknuli ste na ikonu aplikacije!!!",
                        Toast.LENGTH_LONG).show();

                Intent i = new Intent(this, MainActivity.class);
                i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(i);

                return true;
            case 0:
                Toast.makeText(this, "Kliknuli ste na Stavka 1",
                        Toast.LENGTH_LONG).show();
                return true;
            case 1:
                Toast.makeText(this, "Kliknuli ste na Stavka 2",
                        Toast.LENGTH_LONG).show();
                return true;
            case 2:
                Toast.makeText(this, "Kliknuli ste na Stavka 3",
                        Toast.LENGTH_LONG).show();
                return true;
            case 3:
                Toast.makeText(this, "Kliknuli ste na Stavka 4",
                        Toast.LENGTH_LONG).show();
                return true;
            case 4:
                Toast.makeText(this, "YKliknuli ste na Stavka 5",
                        Toast.LENGTH_LONG).show();
                return true;
        }
        return false;
    }
}
```

NAČIN FUNCIONISANJA ACTIONBAR KOMPONENTE

Atribut android:theme omogućava prikazivanje i uklanjanje komponente ActionBar.

Atribut android:theme omogućava prikazivanje i uklanjanje komponente ActionBar. Definisanjem njegove vrednosti na sledeći način: ***android:theme="@style/Theme.AppCompat.Light.NoActionBar*** ", ActionBar neće biti prikazan. Sledećom slikom prikazan je izgled ekrana u kojeg je učitana glavna aktivnost kreirane aplikacije sa ActionBar kontrolom.

Slika 4.2 ActionBar kontrola

Međutim, bolji način je kroz programski kod iskoristiti referencu na ActionBar i pozivom metode hide() izvršiti skrivanje komponente. Pozivom metode show(), komponenta ActionBar će ponovo biti dostupna na ekranu.

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

        ActionBar actionBar = getSupportActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);
        actionBar.hide();
    }
```

Slika 4.3 ActionBar je uklonjen

DODAVANJE NOVIH STAVKI U ACTIONBAR

Prečice za izvesne akcije u aplikaciji nazivaju se stavke akcija.

Stavke akcija dodaju se u ActionBar u formi menija. Kao takve, one predstavljaju prečice za izvesne akcije u aplikaciji koje se često izvršavaju. Sledećom slikom prikazan je meni stavki. Prve dve stavke su smeštele levo na ActionBaru, a ostale u padajućem meniju.

Prikazani programski kod klase aktivnosti nosi sledeće značajne informacije:

1. Stavke akcija, u ActionBar-u, se prikazuju izvršavanjem metode onCreateOptionMenu();
2. Prikazivanje liste stavki menija omogućeno je izvršavanjem metode CreateMenu();
3. Da bi kao stavka menija bila prikazana stavka neke akcije, neophodno je implementirati odgovarajuću setShowAsAction() metodu;
4. Kao odgovor na izabranu stavku menija, izvršava se metoda onOptionItemSelected() .

Stavke ActionBar-a prikazane su sledećom slikom. Moguće je primetiti da stavke koje nisu stale na traku, formirale su padajući meni.

Slika 4.4 Stavke u ActionBar kontroli

PODEŠAVANJE AKCIONIH STAVKI I IKONE APLIKACIJE

Korisnik aplikacije može da klikne na stavku akcije ili ikonu aplikacije pri čemu se izvršavaju odgovajuće metode.

Kako je primer tekao stavke menija prikazivane su bez dopunskog teksta koji bi ukazao na šta se one odnose. Ukoliko postoji potreba za prikazivanjem stavke akcije zajedno sa odgovarajućim tekstom, to je moguće učiniti primenom konstante *SHOW_AS_ACTION_WITH_TEXT*. Kombinacija ove konstante sa konstantom ***SHOW_AS_ACTION_IF _ROOM*** i operatorom „|“ (vidi kod) prikazaće stavku, zajedno sa odgovarajućim tekstom, ukoliko na ekranu ima dovoljno prostora (sledeća slika). Takođe, pojedinim stavkama je pridružena i ikona koja reaguje na klik (videti kod).
Klik na ikonu aplikacije omogućen je izvršavanjem metode setDisplayHomeAsEnabled(). Nakon klika, korisnici se usmeravaju na glavnu aktivnost aplikacije. Da bi navedeno bilo omogućeno, neophodno je kreirati Intent objekat i u njegovu definiciju dodati sledeće: *intent.FLAG_ACTIVITY_CLEAR_TOP* (videti kod). Ovaj indikator obezbeđuje uklanjanje svih aktivnosti sa steka prilikom vraćanja na početnu.
Na kraju ove teme priložen je i mali video tutorijal koji pokazuje dodavanje kontrole ActionBAr u Android aplikaciju.

Slika 4.5 Prikazivanje stavki sa tekstom

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

PRIMER 11 - VEŽBANJE KONTROLE ACTIONBAR I KREIRANJE MENIJA

Proširivanje znanja o ActionBar kontroli

Slika 4.6 Traženi izgled ekrana

Zadatak:

1. Kreirati Android aplikaciju koja implementira ActionBar;
2. U ActionBar kontroli se nalaze skraćenice fakulteta našeg Univerziteta zajedno sa ikonama (logo Univerziteta), broj telefona centrale i adresa web sajta;
3. Preuzeti logo Univerziteta i u res/mipmap snimiti ga u .png fajlove za odgovarajuće rezolucije: mdpi (48px x 48px), hdpi (72px x 72px) i tako dalje;
4. Tu istu sliku snimiti u res/drawable u punoj rezoluciji i iskoristiti je za pozadinu glavne aktivnosti;
5. Za temu u AndroidManifest.xml izabrati DayNight.ActionBar;
6. Klikom na neku od opcija, program izbacuje odgovarajući komentar.

Slika 4.7 Lokacija ikona i pozadine

PRIMER 11 - LISTING

Sledi listing datoteka activity_main.xml, AndroidManifest.xml i MainActivity.java

Listing: 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="vertical"
    android:background="@drawable/metpozadina">

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Univerzitet Metropolitan Beograd" />

</LinearLayout>
```

Listing: AndroidManifest.xml

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/metlogo"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.DayNight.DarkActionBar">

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

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

</manifest>
```

Listing: MainActivity.java

```
package com.metropolitan.actionbardemo;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    /** Poziva se kada se kreira aktivnost. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ActionBar actionBar = getSupportActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        CreateMenu(menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        return MenuChoice(item);
    }

    private void CreateMenu(Menu menu)
    {
        MenuItem mnu1 = menu.add(0, 0, 0, "FIT");
        {
            mnu1.setIcon(R.mipmap.metlogo);
            mnu1.setShowAsAction(
                    MenuItem.SHOW_AS_ACTION_IF_ROOM |
                            MenuItem.SHOW_AS_ACTION_WITH_TEXT);
        }
        MenuItem mnu2 = menu.add(0, 1, 1, "FAM");
        {
            mnu2.setIcon(R.mipmap.metlogo);
            mnu2.setShowAsAction(
                    MenuItem.SHOW_AS_ACTION_IF_ROOM |
                            MenuItem.SHOW_AS_ACTION_WITH_TEXT);
        }
        MenuItem mnu3 = menu.add(0, 2, 2, "FDU");
        {
```

```
            mnu3.setIcon(R.mipmap.metlogo);
            mnu3.setShowAsAction(
                    MenuItem.SHOW_AS_ACTION_IF_ROOM |
                            MenuItem.SHOW_AS_ACTION_WITH_TEXT);
        }
        MenuItem mnu4 = menu.add(0, 3, 3, "TEL: +381 69 2030885");
        {
            mnu4.setShowAsAction(
                    MenuItem.SHOW_AS_ACTION_IF_ROOM |
                            MenuItem.SHOW_AS_ACTION_WITH_TEXT);
        }
        MenuItem mnu5 = menu.add(0, 4, 4, "Sajt: www.metropolitan.ac.rs");
        {
            mnu5.setShowAsAction(
                    MenuItem.SHOW_AS_ACTION_IF_ROOM |
                            MenuItem.SHOW_AS_ACTION_WITH_TEXT);
        }
    }

    private boolean MenuChoice(MenuItem item)
    {
        switch (item.getItemId()) {
            case  android.R.id.home:
                Toast.makeText(this,
                        "Kliknuli ste na ikonu aplikacije!!!",
                        Toast.LENGTH_LONG).show();

                Intent i = new Intent(this, MainActivity.class);
                i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(i);

                return true;
            case 0:
                Toast.makeText(this, "Izabrali ste FIT",
                        Toast.LENGTH_LONG).show();
                return true;
            case 1:
                Toast.makeText(this, "Izabrali ste FAM",
                        Toast.LENGTH_LONG).show();
                return true;
            case 2:
                Toast.makeText(this, "Izabrali ste FDU",
                        Toast.LENGTH_LONG).show();
                return true;
            case 3:
                Toast.makeText(this, "Izabrali ste kontakt telefon",
                        Toast.LENGTH_LONG).show();
                return true;
            case 4:
                Toast.makeText(this, "Izabrali ste web sajt",
                        Toast.LENGTH_LONG).show();
                return true;
        }
```

```
        return false;
    }
}
```

ZADATAK 4 - IMPLEMENTIRAJTE MENI I KONTROLU ACTIONBAR

Pokušajte sami

Iskoristite primer koji ste kreirali tokom savladavanja gradiva ove lekcije i na pokazane načine implementirajte menije i kontrolu ActionBar.

Poglavlje 5

Programsko kreiranje korisničkog interfejsa

KREIRANJE DINAMIČKOG UI

Dinamički korisnički interfejs kreira se primenom JAVA klasa.

Za razliku od dosadašnjeg pristupa, gde je UI kreiran kroz XML datoteke, u nastavku će biti pokazano kako se korisnički interfejs može kreirati JAVA programskim kodom. Ovaj pristup je posebno značajan u situacijama kada dolazi do dinamičke promene komponenata UI u toku izvršavanja aplikacije.

Od posebnog značaja, za ovakav način kreiranja korisničkog interfejsa, jeste kvalitetno kodiranja klasa aktivnosti. Kroz sledeći primer biće prikazan kod kojim je omogućeno dinamičko kreiranje korisničkog interfejsa za određenu aktivnost.

MainActivity.java je klasa kojom je omogućeno dodavanje komponenata u UI.

```
package com.metropolitan.dinamickiguidemo;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.ViewGroup.LayoutParams;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.main);
        //---param za poglede---
        LayoutParams params =
                new LinearLayout.LayoutParams(
                        LayoutParams.FILL_PARENT,
                        LayoutParams.WRAP_CONTENT);

        //---kreiranje izgleda---
        LinearLayout layout = new LinearLayout(this);
        layout.setOrientation(LinearLayout.VERTICAL);
```

```
        //---kreiranje a textview---
        TextView tv = new TextView(this);
        tv.setText("Ovo je polje za tekst");
        tv.setLayoutParams(params);

        //---kreiranje button---
        Button btn = new Button(this);
        btn.setText("Ovo je dugme");
        btn.setLayoutParams(params);


        //---dodaje textview---
        layout.addView(tv);

        //---dodaje button---
        layout.addView(btn);

        //---kreira parametre izgleda---
        LinearLayout.LayoutParams layoutParam =
                new LinearLayout.LayoutParams(
                        LayoutParams.FILL_PARENT,
                        LayoutParams.WRAP_CONTENT );
        this.addContentView(layout, layoutParam);
    }

}
```

Ekran mobilnog uređaja koji pokazuje dinamički kreirane kontrole dat je sledećom slikom.

Slika 5.1 Dinamički kreirane GUI kontrole

KREIRANJE DINAMIČKOG UI – FUNKCIONISANJE PROGRAMA

Isključivanjem setContentView() onemogućava se učitavanje UI iz main.xml.

Kada se pogleda priloženi kod, prvo što je moguće uočiti da je naredba setContentView() pretvorena u komentar, a to znači da će biti ignorisana tokom izvršavanja programa. Ovom akcijom onemogućeno je učitavanje interfejsa određenog main.xml datotekom. Od posebnog značaja su sledeći koraci:

- kreiran je LayoutParam()objekat čiji je zadatak omogućavanje parametara izgleda koje će koristiti drugi pogledi koji će naknadno biti kreirani;
- Kreiran jeLinearLayout() objekat koji sadrži sve poglede aktivnosti;
- Definisani su TextView i Button pogledi;
- Kreirani pogledi su dodati uLinearLayout() objekat;
- Kreiran je LayoutParam()objekat kojeg koristi *LinearLayout()* objekat;

- Na kraju, LinearLayout()objekat je uključen u aktivnost na sledeći način: *this.addContentView(layout,layoutParam);*

Odavde je moguće zaključiti da je korišćenje JAVA koda za kreiranje korisničkog interfejsa težak posao. Otuda, UI se dinamički generiše isključivo u situacijama kada je to neophodno. Sledećim video tutorijalom studenti mogu da pogledaju proceduru dinamičkog kreiranja GUI komponenata.

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

OBAVEŠTENJA UI – REDEFINISANJE METODA AKTIVNOSTI

Na nivou aktivnosti Activity klasa sadrži metode koje je moguće redefinisati za konkretne potrebe.

Korisnici komuniciraju sa Android aplikacijom korišćenjem UI u dva nivoa:

- Nivo aktivnosti;
- Nivo pogleda.

Na nivou aktivnosti Activity klasa sadrži metode koje je moguće redefinisati za konkretne potrebe. Opšte metode koje je moguće redefinisati u ovom kontekstu su:

- onKeyDown() – Izvršava se kada je pritisnut taster i ne rukuje se ni jednim pogledom konkretne aktivnosti;
- onKeyUp() – Izvršava se kada pusti pritisnuti taster i ne rukuje se ni jednim pogledom konkretne aktivnosti;
- onMenuItemSelected()– Izvršava se kada korisnik selektuje stavku menija;
- onMenuOpened()– Izvršava se kada korisnik otvori meni.

REDEFINISANJE METODA AKTIVNOSTI

U klasi aktivnosti aplikacije se redefinišu metode bazne klase Activity pri čemu im se dodeljuju konkretni zadaci.

U sledećem primeru biće pokazano redefinisanje metoda iz super klase aktivnosti (android.support.v7.app.AppCompatActivity). Prvo će biti prikazan UI definisan datotekom 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="vertical" >
```

```
    <TextView
        android:layout_width="214dp"
        android:layout_height="wrap_content"
        android:text="Naziv Univerziteta?"
        />
    <EditText
        android:id="@+id/txt1"
        android:layout_width="214dp"
        android:layout_height="wrap_content"
        />
    <Button
        android:id="@+id/btn1"
        android:layout_width="106dp"
        android:layout_height="wrap_content"
        android:text="OK"
        />
    <Button
        android:id="@+id/btn2"
        android:layout_width="106dp"
        android:layout_height="wrap_content"
        android:text="Cancel"
        />

</LinearLayout>
```

Nakon uključivanja paketa za klasu aktivnosti aplikacije: ***android.view.KeyEvent*** **i** ***android.widget.Toast*** **,** moguće je redefinisati metodu onKeyDown() sledećim kodom.

```
@Override
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        switch (keyCode)
        {
            case KeyEvent.KEYCODE_DPAD_CENTER:
                Toast.makeText(getBaseContext(),
                        "Centar je kliknut",
                        Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_LEFT:
                Toast.makeText(getBaseContext(),
                        "Leva strelica je kliknuta",
                        Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_RIGHT:
                Toast.makeText(getBaseContext(),
                        "Desna strelica je kliknuta",
                        Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_UP:
                Toast.makeText(getBaseContext(),
                        "Gornja strelica je kliknuta",
                        Toast.LENGTH_LONG).show();
                break;
```

```
            case KeyEvent.KEYCODE_DPAD_DOWN:
                Toast.makeText(getBaseContext(),
                        "Donja strelica je kliknuta",
                        Toast.LENGTH_LONG).show();
                break;
        }
        return false;
    }
```

NAČIN FUNKCIONISANJA REDEFINISANE METODE

Redefinisana metoda zamenjuje konkretnim akcijama praznu metodu iz Super klase.

Pokretanjem programa inicira se glavna aktivnost. Tada će pokazivač ukazivati na EditText polje na koje je postavljen fokus. U Android operativnom sistemu, kontrola koja je pod fokusom pokušava ga upravlja generisanim događajem. Konkretno, u tekst polju će, nakon pritiska na odgovarajući taster, biti prikazan karakter koji mu odgovara. Međutim, ako se pritisnu tasteri koji odgovaraju strelicama, EditText polje neće reagovati na ovaj način. Tada se, u ovom slučaju, izvršava metoda *onKeyDown()* i akcija koja odgovara kliku na odgovarajuću strelicu. Sledeće što će se desiti jeste prenošenje fokusa na dugme *OK*.

Posebno, ako pogled EditText već sadrži neki string i pokazivač je na kraju teksta, klikom na strelicu *levo* ne inicira se *onKeyDown()* metoda već se pokazivač pomera za jedno mesto ulevo. Razlog je činjenica da je EditText već rukovao tim događajem. Klik na strelicu *desno* dovešće do izvršavanja metode *onKeyDown()*.

Metoda *onKeyDown()* vraća vrednosti *true* ili *false.* Ukoliko se ukazuje sistemu da je završeno obrađivanje događaja i da sistem ne mora da nastavi obradu, metoda vraća vrednost true.

Sledećom slikom prikazan je deo u izvršenju posmatranog primera.

Slika 5.2 Demonstracija primene redefinisanih metoda

REGISTROVANJE DOGAĐAJA ZA POGLEDE.

Anonimnim klasama i anonimnim unutrašnjim klasama moguće je registrovati događaje koji se odnose na poglede korisničkog interfejsa.

Kada korisnik programa pritisne određeni taster, neophodno je upravljati događajem da bi došlo do realizovanja određene akcije. Da bi ovo bilo omogućeno, neophodno je jasno registrovati događaje za pojedinačne poglede.

U prethodnom primeru, za aktivnost su definisana dva dugmeta. Tada je moguće registrovati događaje, koji se odnose na klikove, primenom *anonimne klase* (videti kod uz odgovarajući komentar).

Postoji još jedan način za upravljanje događajima. Moguće je definisati u *unutrašnju anonimnu klasu* za rukovanje događajima (videti kod uz odgovarajući komentar).

Koju metodu koristiti?

1. Ukoliko postoji više pogleda koji se odnose na jednog rukovaoca događajima koristi se anonimna klasa.
2. Ako se jedan rukovalac događajima odnosi na jedan pogled, koristiće se anonimna unutrašnja klasa.

```
package com.metropolitan.uiaktivnsotidemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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


        Button btn1 = (Button)findViewById(R.id.btn1);
        btn1.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //---radi nešto---
            }
        });

        Button btn2 = (Button)findViewById(R.id.btn2);
        btn2.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //---radi nešto---
            }
        });

        //---anonimna inner class kao onfocus osluškivač---
        EditText txt1 = (EditText)findViewById(R.id.txt1);
        txt1.setOnFocusChangeListener(new View.OnFocusChangeListener()
        {
            @Override
            public void onFocusChange(View v, boolean hasFocus) {
                Toast.makeText(getBaseContext(),
                        ((EditText) v).getId() + " ima fokus - " + hasFocus,
                        Toast.LENGTH_LONG).show();
```

```
            }
        });
    }

    //---anonimna inner class kao onfocus osluškivač---
    private OnClickListener btnListener = new OnClickListener()
    {
        public void onClick(View v)
        {
            Toast.makeText(getBaseContext(),
                    ((Button) v).getText() + " je kliknut",
                    Toast.LENGTH_LONG).show();
        }
    };

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        switch (keyCode)
        {
            case KeyEvent.KEYCODE_DPAD_CENTER:
                Toast.makeText(getBaseContext(),
                        "Centar je kliknut",
                        Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_LEFT:
                Toast.makeText(getBaseContext(),
                        "Leva strelica je kliknuta",
                        Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_RIGHT:
                Toast.makeText(getBaseContext(),
                        "Desna strelica je kliknuta",
                        Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_UP:
                Toast.makeText(getBaseContext(),
                        "Gornja strelica je kliknuta",
                        Toast.LENGTH_LONG).show();
                break;
            case KeyEvent.KEYCODE_DPAD_DOWN:
                Toast.makeText(getBaseContext(),
                        "Donja strelica je kliknuta",
                        Toast.LENGTH_LONG).show();
                break;
        }
        return false;
    }

}
```

PRIMER 12 - VEŽBANJE DINAMIČKOG KREIRANJA GUI

U Java kodu se generišu elementi korisničkog interfejsa

Kombinovanjem dinamičkog i statičkog pristupa kreirati Android aplikaciju po uzoru na prethodnu sliku.

Statički elementi dati su sa 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="vertical"
    android:background="@drawable/metpozadina">

    <TextView
        android:id="@+id/textv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Demonstracija dinamičkog kreiranja GUI kontrola" />

</LinearLayout>
```

```
<?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"
    android:background="@drawable/metpozadina">

    <TextView
        android:id="@+id/textv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Demonstracija dinamičkog kreiranja GUI kontrola" />

</LinearLayout>
```

Dinamički GUI elementi su kreirani klasom glavne aktivnosti:

```
package com.metropolitan.dinamickiguidemo;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.ViewGroup.LayoutParams;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
```

```
public class MainActivity extends AppCompatActivity {

    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //---param za poglede---
        LayoutParams params =
                new LinearLayout.LayoutParams(
                        LayoutParams.FILL_PARENT,
                        LayoutParams.WRAP_CONTENT);

        //---kreiranje izgleda---
        LinearLayout layout = new LinearLayout(this);
        layout.setOrientation(LinearLayout.VERTICAL);
        //---kreiranje a textview---
        TextView tv = new TextView(this);
        tv.setText("Ovo je polje za tekst");
        tv.setLayoutParams(params);

        //---kreiranje button---
        Button btn = new Button(this);
        btn.setText("Ovo je dugme");
        btn.setLayoutParams(params);

        //---dodaje textview---
        layout.addView(tv);

        //---dodaje button---
        layout.addView(btn);

        //---kreira parametre izgleda---
        LinearLayout.LayoutParams layoutParam =
                new LinearLayout.LayoutParams(
                        LayoutParams.FILL_PARENT,
                        LayoutParams.WRAP_CONTENT );
        this.addContentView(layout, layoutParam);
    }

}
```

Slika 5.3 Izgled ekrana

ZADATAK 5- IMPLEMENTIRAJTE KONTOLE IZ KLASE AKTIVNOSTI

Pokušajte sami

Iskoristite primer koji ste kreirali tokom savladavanja gradiva ove lekcije i na pokazane načine implementirajte GUI kontrole na dinamički način iz Java koda glavne klase aktivnosti.

PRIMER 12 - VEŽBANJE GRUPA POGLEDA

Kreiranje aplikacije sa zadatim layout - om

Kreirati Android aplikaciju sa pozadinom našeg Univerziteta, koja u res/mipmap čuva slike našeg Univerziteta i njegove okoline. Slike se u glavnoj klasi pakuju u GridView raspored. Klikom na slike, Toast klasom se realizuje povratna informacija po želji.

Slika 5.4 Željeni izgled ekrana

GUI listing:

```
<?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"
    android:background="@drawable/metpozadina">

    <GridView
        android:id="@+id/gridview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:numColumns="auto_fit"
        android:verticalSpacing="10dp"
        android:horizontalSpacing="10dp"
        android:columnWidth="90dp"
        android:stretchMode="columnWidth"
        android:gravity="center" />

</LinearLayout>
```

Java listing:

```
import android.os.Bundle;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.Toast;
```

```
public class MainActivity extends AppCompatActivity {

    //---Slike za prikazivanje---
    Integer[] imageIDs = {
            R.mipmap.pic1,
            R.mipmap.pic2,
            R.mipmap.pic3,
            R.mipmap.pic4,
            R.mipmap.pic5,
            R.mipmap.pic6,
            R.mipmap.pic7
    };
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        GridView gridView = (GridView) findViewById(R.id.gridview);
        gridView.setAdapter(new ImageAdapter(this));

        gridView.setOnItemClickListener(new OnItemClickListener(){
            public void onItemClick(AdapterView<?> parent,
                                    View v, int position, long id){
                Toast.makeText(getBaseContext(),
                        "Slika " + (position + 1) + " je kliknuta",
                        Toast.LENGTH_SHORT).show();
            }
        });

    }

    public class ImageAdapter extends BaseAdapter{
        private Context context;

        public ImageAdapter(Context c){
            context = c;
        }

        //---vraća broj slika---
        public int getCount() {
            return imageIDs.length;
        }

        //---vraća stavku---
        public Object getItem(int position) {
            return position;
        }

        //---vraća ID stavke---
        public long getItemId(int position) {
            return position;
```

```
        }

        //---vraća ImageView pogled---
        public View getView(int position, View convertView,
                            ViewGroup parent){
            ImageView imageView;
            if (convertView == null) {
                imageView = new ImageView(context);
                imageView.setLayoutParams(new
                        GridView.LayoutParams(85, 85));
                imageView.setScaleType(
                        ImageView.ScaleType.CENTER_CROP);
                imageView.setPadding(5, 5, 5, 5);
            } else {
                imageView = (ImageView) convertView;
            }
            imageView.setImageResource(imageIDs[position]);
            return imageView;
        }
    }

}
```

ZADATAK 6 - PRIMENITE DRAG AND DROP FUNKCIONALNOSTI RAZVOJNOG OKRUŽENJA

Probajte sami!!!

Pokušajte priložene primereda kreirate primenom Drag and Drop funkcionalnosti razvojnog okruženja.

Poglavlje 6

Domaći zadatak 3 - 1

ZADATAK 1 – VEŽBANJE KREIRANJA UI I UPRAVLJANJA ORIJENTACIJOM EKRANA

Cilj ovog zadatka je kreiranje aplikacije sa zadatim rasporedom UI i mogućnosti prilagođavanja sadržaja promenama u orijentaciji ekrana.

1. U RelativeLayout prikazu kreirati UI sa slike;
2. U zavisnosti od rotacije ekrana, prikazati na ekranu odgovarajuće obaveštenje;
3. Klikom na prvo dugme prikazati naziv našeg univerziteta;
4. Klikom na drugo dugme prikazati Google Map lokaciju našeg univerziteta.
5. Pomoć u radu potražiti na linku: http://www.tutorialspoint.com/android/android_relative_layout.htm

Zadati interfejs izgleda ovako:

Slika 6.1 Zadati interfejs

ZADATAK 2 – VEŽBANJE KREIRANJA UI I UPRAVLJANJA ORIJENTACIJOM EKRANA

Cilj ovog zadatka je kreiranje aplikacije sa zadatim rasporedom UI, operacijama sa ActionBar-om i mogućnosti prilagođavanja sadržaja promenama u orijentaciji ekrana.

1. Kreirati u LinearLayout (vertikalno) korisnički interfejs koji sadrži: tekst polje i dva dugmeta (DA i NE);
2. Glavnu aktivnost nazvati Metropolitan Univerzitet;
3. Na ActionBaru dodati tri stavke: FIT, Fakultet za menadžment i Fakultet digitalnih umetnosti;
4. Izborom stavke i klikom na dugme (DA ili NE) potvrditi da li ste student izabranog fakulteta;
5. Pomoć u izradi rešenja potražiti u primeru iz ove lekcije i na linku: http://www.tutorialspoint.com/android/android_linear_layout.htm

ZADATAK 3 – SAMOSTALNO KREIRANJE APLIKACIJE NA OSNOVU URAĐENIH REŠENJA

Studenti da pokažu samostalnost u radu.

Kreirati po sopstvenoj želji Android aplikacije po sledećim uzorcima:

1. TableLayout: http://www.tutorialspoint.com/android/android_table_layout.htm
2. AbsoluteLayout: http://www.tutorialspoint.com/android/android_user_interface_layouts.htm
3. FrameLayout: http://www.tutorialspoint.com/android/android_frame_layout.htm;
4. ListView: http://www.tutorialspoint.com/android/android_list_view.htm
5. GridView: http://www.tutorialspoint.com/android/android_grid_view.htm

Poglavlje 7

Upotreba osnovnih pogleda

OSNOVNI POGLEDI - UVOD

Osnovni pogledi su elementi svakog Android korisničkog interfejsa.

Osnovnim pogledima omogućen je realizovanje osnovnih programskih akcija, kao što su: unos tekstualnih informacija, osnovno selektovanje i pokretanje akcija. Ovi pogledi se nazivaju osnovnim jer se u najvećem broju slučajeva, u praksi, vrši njihovo kombinovanje u složenije poglede, u takozvane grupe pogleda. Primena ovih GUI elemenata je karakteristična za sve savremene programske jezike pri čemu se samo razlikuje terminologija po kojoj se prepoznaju ove kontrole. Budući da se razvoj Android aplikacija oslanja na JAVA tehnologije i ovde imamo jasno definisane osnovne GUI elemente koji se kreiraju i koriste na dva načina: statički - definisanjem u XML datoteci i dinamički - definisanjem u JAVA klasi aktivnosti.

U svrhu kreiranje UI najčešće se koriste sledeći pogledi:

- TextView;
- EditText;
- Button;
- ImageButton;
- CheckBox;
- ToggleButton;
- RadioButton;
- RadioGroup.

U nastavku izlaganja biće demonstrirani navedeni pogledi i mogućnosti njihovog integrisanja u složenije poglede sa odgovarajućom organizacijom na ekranu.

TEXTVIEW POGED

TextView pogled se koristi za prikazivanje teksta korisnicima aplikacije.

TextView pogled se koristi za prikazivanje teksta korisnicima aplikacije i na drugim platformama se najčešće naziva labela. Radi se osnovnom pogledu koji se skoro uvek koristi u Android aplikacijama. Prilikom kreiranja svakog novog Android projekta, Eclipse IDE ili Android Studio IDE razvojnim okruženjem, kreira se activity_main.xml datoteka koja obavezno sadrži <TextView> element, a to je prikazano kodom datim sledećim listingom. Ako je neophodno da se omogući korisnicima aplikacije da menjaju tekst, biće korišćen *EditText* klasa, izvedena iz TextView klase. O ovom pogledu biće više reči u narednom izlaganju.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.metropolitan.osnovnipogledidemo.MainActivity">

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

OSTALI OSNOVNI POGLEDI

Osnovni pogledi se najčešće kreiraju statički activity_main.xml datotekom.

TextView pogled ima zadatak da istakne neku informaciju na ekranu ili da obeleži neku drugu osnovnu GUI kontrolu (pogled). Android podržava sve poznate osnovne GUI elemente za kreiranje bogatih i funkcionalnih grafičkih korisničkih interfejsa.

Biće obrađeni sledeći osnovni pogledi:

- ***EditText*** – omogućava dodavanje i ažuriranje teksta;
- ***Button*** – predstavlja dugme koje je moguće pritisnuti;
- ***ImageButton*** – dugme sa slikom;
- ***CheckBox*** – specijalni tip tastera sa dva stanja: selektovan (čekiran) i neselektovan (nečekiran);
- ***ToggleButton*** – dugme sa svetlosnim indikatorom za prikazivanje stanja selektovan/ neselektovan;
- ***RadioButton*** – ima dva stanja selektovan i neselektovan;
- ***RadioGroup*** – Grupa radio dugmadi u kojoj samo jedan, u datom trenutku, može biti selektovan.

Kao što je već napomenuto, ovde će biti kreirana XML datoteka koja čuva elemente korisničkog interfejsa. Ovi elementi prikazani su 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/btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/save"
        android:onClick="btnSaved_clicked"/>

    <Button android:id="@+id/btnOpen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open" />

    <ImageButton android:id="@+id/btnImg1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />

    <EditText android:id="@+id/txtName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <CheckBox android:id="@+id/chkAutosave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Autosave" />

    <CheckBox android:id="@+id/star"
        style="?android:attr/starStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />


    <RadioGroup android:id="@+id/rdbGp1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <RadioButton android:id="@+id/rdb1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Option 1" />
        <RadioButton android:id="@+id/rdb2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Option 2" />
    </RadioGroup>

    <ToggleButton android:id="@+id/toggle1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

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

UPRAVLJANJE OSNOVNIM POGLEDIMA

Akcije koje su rezultat interakcije korisnika sa osnovnim pogledima definisane su JAVA klasom aktivnosti.

U projektu, za demonstraciju upotrebe osnovnih pogleda, neophodno je modifikovati klasu aktivnosti tako da kreirane kontrole, activity_main.xml datotekom, imaju konkretne zadatke. Prvi zadataka je učitavanje korisničkog interfejsa ***onCreate()*** metodom(). Nakon toga klasa implementira osluškivače za odgovarajuće događaje kontrola i ponašanja koja su rezultat interakcije korisnika sa konkretnom kontrolom iz kreiranog korisničkog interfejsa, aktuelnog Android projekta.

Sledećim listingom prikazana je klasa aktivnosti koja implementira gore pomenutu programsku logiku i funkcionalnosti.

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.RadioGroup.OnCheckedChangeListener;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {

    public void btnSaved_clicked (View view) {
        DisplayToast("Kliknuli ste na dugme Save");
    }

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


        //---Button view---
        Button btnOpen = (Button) findViewById(R.id.btnOpen);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                DisplayToast("Kliknuli ste na dugme Open");
            }
        });


        //---CheckBox---
        CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
        checkBox.setOnClickListener(new View.OnClickListener()
```

```
        {
            public void onClick(View v) {
                if (((CheckBox)v).isChecked())
                    DisplayToast("CheckBox je čekiran");
                else
                    DisplayToast("CheckBox nije čekiran");
            }
        });

        //---RadioButton---
        RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
        radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener()
        {
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                RadioButton rb1 = (RadioButton) findViewById(R.id.rdb1);
                if (rb1.isChecked()) {
                    DisplayToast("Option 1 je čekiran!");
                } else {
                    DisplayToast("Option 2 je Nečekiran!");
                }
            }
        });

        //---ToggleButton---
        ToggleButton toggleButton =
                (ToggleButton) findViewById(R.id.toggle1);
        toggleButton.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v) {
                if (((ToggleButton)v).isChecked())
                    DisplayToast("Toggle button je uključen");
                else
                    DisplayToast("Toggle button je isključen");
            }
        });
    }

    private void DisplayToast(String msg)
    {
        Toast.makeText(getBaseContext(), msg,
                Toast.LENGTH_SHORT).show();
    }

}
```

FUNKCIONISANJE OSNOVNIH POGLEDA

Interakcijom sa različitim pogledima moguće je primetiti odgovarajuće akcije aktivnosti.

Grafički korisnički interfejs projekta je osmišljen tako da koristeći vertikalni LinearLayout raspored, pogledi su postavljeni jedan iznad drugog:

- Prvo dugme je definisano tako da je njegov element *layout_width* postavljen na fill_parent, a to znači da zauzima celu širinu ekrana;
- Drugo dugme ima vrednost wrap_content za naznačeni atribut, a to znači da je njegova širina definisana prostorom za prikazivanje sadržaja;
- ImageButton pomoću src atributa postavlja sliku;
- EditText pokazuje tekst polje za unos i / ili ažuriranje teksta;
- CheckBox definiše polje koje je moguće selektovati ili deselektovati. Upotrebom atributa *style (*videti activity_main.xml) moguće je promeniti izgled polja (u zvezdicu u ovom primeru);
- Radio grupa sadrži dva dugmeta koji se međusobno isključuju;
- ToggleButton prikazuje taster koji ima dva stanja on/off i praktično predstavlja atraktivnu reprezentaciju CheckBox kontrole.

Svaki pogled ima vlastiti identifikator definisanid atributom, kojeg je moguće koristiti pomoću metoda View.findViewById() i Activity.findViewById().

Sledećom slikom prikazana je aktivnost u koju je učitan prethodno kreirani korisnički interfejs. Aktivnost se pokreće izborom opcije Run app ili Shift + F10 u Android Studio IDE razvojnom okruženju.

Slika 7.1 Učitana aktivnost sa kreiranim GUI pogledima

UPRAVLJANJE DOGAĐAJIMA ZA ANDROID POGLEDE

Da bi upravljanje konkretnim pogledom bilo moguće, neophodno ga je programski locirati.

Za svaki pogled, kreiran tokom ***onCreate()***događaja, upravljanje je moguće ukoliko je pogled programski lociran. Navedeno je omogućeno prosleđivanjem idatributa metodi findViewById() koja pripada osnovnoj klasi Activity (AppCompatActivity za najnovije API Android verzije). Za dugme, koje je korišćeno u primeru, lociranje je rešeno na sledeći način:

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

Metodom setOnClickListener() omogućeno je dobijanje povratne informacije koja će biti prezentovana prilikom obraćanja pogledu. Drugim rečima postavljen je osluškivač na ovu kontrolu koji očekuje interakciju između korisnika i dugmeta.

```
btnOpen.setOnClickListener(new View.OnClickListener() {
 ublic void onClick(View v) {
 DisplayToast("Kliknuli ste na dugme Open");
 }
});
```

StanjeCheckBox-a proverava se pomoću argumenta za metodu onClick() kojim se, potom, poziva metodaisChecked() (sledeći listig).

```
//---CheckBox---
        CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
        checkBox.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v) {
                if (((CheckBox)v).isChecked())
                    DisplayToast("CheckBox je čekiran");
                else
                    DisplayToast("CheckBox nije čekiran");
            }
        });
```

Metoda setOnCheckedChangeListener() se koristi kada u radio grupi treba da se dobije informacija o promeni stanjaradio dugmeta. Za svako dugme postoji metoda isChecked() kojom se proverava stanje kontrole (sledeći listing).

ToggleButton funkcioniše potpuno isto kao CheckBox.

```
//---RadioButton---
        RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
        radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener()
        {
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                RadioButton rb1 = (RadioButton) findViewById(R.id.rdb1);
                if (rb1.isChecked()) {
                    DisplayToast("Option 1 je čekiran!");
                } else {
                    DisplayToast("Option 2 je Nečekiran!");
                }
            }
        });
```

PRIMER 13 - VEŽBANJE DODAVANJA OSNOVNIH POGLEDA U GUI

Praktičan rad sa osnovnim GUI elementima

Zadatak:

1. Koristeći LinearLayout sa vertikalnim rasporedom kreirati GUI sa sledeće slike.
2. Dodeliti funkcionalnost kontrolama.

Slika 7.2 GUI zadatka 1

Datoteka korisničkog interfejsa data je 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:background="@drawable/metpozadina"
    android:orientation="vertical" >

    <Button android:id="@+id/btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/save"
        android:onClick="btnSaved_clicked"/>

    <Button android:id="@+id/btnOpen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open" />

    <ImageButton android:id="@+id/btnImg1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@mipmap/metlogo" />

    <EditText android:id="@+id/txtName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <CheckBox android:id="@+id/chkAutosave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Autosave" />

    <CheckBox android:id="@+id/star"
        style="?android:attr/starStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />


    <RadioGroup android:id="@+id/rdbGp1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <RadioButton android:id="@+id/rdb1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Option 1" />
        <RadioButton android:id="@+id/rdb2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Option 2" />
    </RadioGroup>

    <ToggleButton android:id="@+id/toggle1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

```
</LinearLayout>
```

Glavna klasa aktivnosti je priložena sledećim kodom:

```
package com.metropolitan.osnovnipogledidemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.RadioGroup.OnCheckedChangeListener;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {

    public void btnSaved_clicked (View view) {
        DisplayToast("Kliknuli ste na dugme Save");
    }

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


        //---Button view---
        Button btnOpen = (Button) findViewById(R.id.btnOpen);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                DisplayToast("Kliknuli ste na dugme Open");
            }
        });


        //---CheckBox---
        CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
        checkBox.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v) {
                if (((CheckBox)v).isChecked())
                    DisplayToast("CheckBox je čekiran");
                else
                    DisplayToast("CheckBox nije čekiran");
            }
        });

        //---RadioButton---
```

```
        RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
        radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener()
        {
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                RadioButton rb1 = (RadioButton) findViewById(R.id.rdb1);
                if (rb1.isChecked()) {
                    DisplayToast("Option 1 je čekiran!");
                } else {
                    DisplayToast("Option 2 je Nečekiran!");
                }
            }
        });

        //---ToggleButton---
        ToggleButton toggleButton =
                (ToggleButton) findViewById(R.id.toggle1);
        toggleButton.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v) {
                if (((ToggleButton)v).isChecked())
                    DisplayToast("Toggle button je uključen");
                else
                    DisplayToast("Toggle button je isključen");
            }
        });
    }

    private void DisplayToast(String msg)
    {
        Toast.makeText(getBaseContext(), msg,
                Toast.LENGTH_SHORT).show();
    }

}
```

POGLED PROGRESSBAR

ProgressBar obezbeđuje vizuelne informacije o akcijama čije je izvršavanje u toku.

Ukoliko se tokom izvršavanja aplikacije pojavljuje zadatak čije izvršavanje traje, moguće je kontrolom ProgressBar vizuleno ukazati na informaciju u vezi sa trajanjem date akcije. Kao primer moguće je navesti informaciju u vezi sa statusom instalacije aplikacije ili o preuzimanju sadržaja sa Interneta. Primenu kontrole ProgressBar veoma često je moguće sresti u brojnim Android aplikacijama koje komuniciraju sa Internetom.

ProgresBar je pogled koji, kao i svi ostali pogledi, deklarisanjem u XML datoteci korisničkog interfejsa postaje dostupan za učitavanje u glavnu aktivnost ili neku njenu podaktivnost.

```
<?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">

      <ProgressBar android:id="@+id/progressbar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="@android:style/Widget.ProgressBar.Horizontal" />

</LinearLayout>
```

Kontrola ProgresBar se učitava u glavnu aktivnost, zajedno sa celokupnim korisničkim interfejsom, pozivom metode onCreate() u glavnoj klasi aktivnosti. Kod ove klase prezentovan je sledećim listingom.

```
package com.metropolitan.osnovnipogledidemo2;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.ProgressBar;

public class MainActivity extends AppCompatActivity {

    static int progress;
    ProgressBar progressBar;
    int progressStatus = 0;
    Handler handler = new Handler();

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

        progress = 0;
        progressBar = (ProgressBar) findViewById(R.id.progressbar);
        progressBar.setMax(200);

        //---radi nešto u pozadinskoj niti---
        new Thread(new Runnable()
        {
            public void run()
            {
                //—-i ovde radi nešto—-
                while (progressStatus < 100)
                {
                    progressStatus = doSomeWork();
```

```
                    //—-Ažurira ProgresBar—-
                    handler.post(new Runnable()
                    {
                        public void run() {
                            progressBar.setProgress(progressStatus);
                        }
                    });
                }

                //---Sakriva ProgressBar---
                handler.post(new Runnable()
                {
                    public void run()
                    {
                        //---0 - VISIBLE; 4 - INVISIBLE; 8 - GONE---
                        progressBar.setVisibility(View.GONE);
                    }
                });
            }

            //---duga aktivnost---
            private int doSomeWork()
            {
                try {
                    //---simulacija zadatka---
                    Thread.sleep(50);
                } catch (InterruptedException e)
                {
                    e.printStackTrace();
                }
                return ++progress;
            }
        }).start();
    }

}
```

PROGRESSBAR - FUNKCIONISANJE

Inicijalno, ProgressBar je realizovan kao ciklična animacija.

Kontrola *ProgessBar* je veoma korisna za zadatke koji nemaju specifično vreme izvršavanja, poput: send i response komunikacije sa web serverom. Postavljanjem xml taga <ProgressBar> u activity_main.xml datoteku, biće obezbeđeno prikazivanje ikone koja se okreće. Obaveza programera je da obezbedi zaustavljanje kontrole kada se završi obavljanje datog zadatka.

ProgressBarkontrola, u ovom primeru, prati pozadinsku nit za simuliranje zadatka koji dugo traje. Otuda je neophodno imlementiranje klase Thread i Runnable objekta. Metodom run() započinje izvršavanje niti, koja u konkretnom primeru izvršava metodu, simbolično nazvanu,

doSomeWork(). Po završetku zadatka, upotrebljen je Handler objekat za slanje poruke do niti za prekidanje ProgressBar-a. Za navedeno videti priloženi kod klase aktivnosti.

Inicijalno, ProgressBar je realizovan kao horizontalna animacija. Međutim, upotrebom instrukcije:

style **= "@android:style/Widget.ProgressBar.Large„**

kontrola ProressBar će biti prikazan kao ciklična animacija.

Ovde bi trebalo napomenuti da postoje brojne podržani stilovi za prikazivanje ove kontrole.

Sledećom slikom je prikazan pogled ProgressBar kao horizontalna i ciklična animacija.

Slika 7.3 ProgressBar kao ciklična i horizontalna animacija

PRIMER 14 - VEŽBANJE DODAVANJA PROGRESSBAR KONTROLE

ProgressBar kontrola se utvrđuje praktičnim zadacima

Zadatak:

1. Implementirati u glavnu aktivnost kontrolu koja simulira izvršavanje dugotrajnog zadatka;
2. Za ProgressBar izaberite stil po vlastitoj želji.

Rešenje tražiti po uzoru na sledeću sliku.

Slika 7.4 ProgressBar

Datoteka korisničkog interfejsa je data sledećim programskim 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"
    android:background="@drawable/metpozadina">
    <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Nešto se dešava..." />
    <ProgressBar android:id="@+id/progressbar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="@android:style/Widget.Material.ProgressBar.Large" />

</LinearLayout>
```

Glavna klasa aktivnosti data je sledećim kodom.

```
package com.metropolitan.osnovnipogledidemo2;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.ProgressBar;

public class MainActivity extends AppCompatActivity {

    static int progress;
    ProgressBar progressBar;
    int progressStatus = 0;
    Handler handler = new Handler();

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

        progress = 0;
        progressBar = (ProgressBar) findViewById(R.id.progressbar);
        progressBar.setMax(200);

        //---radi nešto u pozadinskoj niti---
        new Thread(new Runnable()
        {
            public void run()
            {
                //—i ovde radi nešto—
                while (progressStatus < 100)
                {
                    progressStatus = doSomeWork();

                    //—Ažurira ProgresBar—
                    handler.post(new Runnable()
                    {
                        public void run() {
                            progressBar.setProgress(progressStatus);
                        }
                    });
                }

                //---Sakriva ProgressBar---
                handler.post(new Runnable()
                {
                    public void run()
                    {
                        //---0 - VISIBLE; 4 - INVISIBLE; 8 - GONE---
                        progressBar.setVisibility(View.GONE);
                    }
                });
```

```
            }

            //---duga aktivnost---
            private int doSomeWork()
            {
                try {
                    //---simulacija zadatka---
                    Thread.sleep(50);
                } catch (InterruptedException e)
                {
                    e.printStackTrace();
                }
                return ++progress;
            }
        }).start();
    }

}
```

ZADATAK 7 - REDEFINIŠITE PRIMER 13

Pokušajte sami

- Otvorite Primer 13;
- Promenite izgled u GridLayout;
- Nastavite da dodajete kontrole po vlastitom izboru;
- Dajte konkretnu funkcionalnost dodatim kontrolama.
- U primer integrišite i ProgressBar za simulaciju učitavanja programa sa neke veb lokacije.

Poglavlje 8

Upotreba Picker Pogleda

TIMEPICKER POGLED

TimePicker pogled omogućava korisnicima selektovanje vremena.

Izbor datuma i vremena su akcije koje se veoma često izvode Android aplikacijama. Android obezbeđuje dva alata kojima su ove funkcionalnosti dostupne, a to su *TimePicker i DatePicker*.

TimePicker pogled omogućava korisnicima da izaberu vreme, i to, u dva režima 24h ili AM/PM. Upravo iz razloga što se veoma često sreću i koriste u Android aplikacijama ove poglede je neophodno detaljno analizirati i prezentovati u narednom izlaganju.

U prvom koraku biće pokazano kako se kontrola TimePicker deklariše u XML datoteci korisničkog interfejsa.

Slika 8.1 TimePicker pogled

Sledećim listingom je prikazana activity_main.xml datoteka sa pripadajućim TimePicker pogledom.

```
<?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" >

    <TimePicker android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

TIMEPICKER - GLAVNA KLASA AKTIVNOSTI I FUNKCIONAISANJE

TimePicker u standardnom Android UI omogućava korisniku da podesi vreme.

Da bi funkcionisanje pogleda TimePicker bilo prezentovano, neophodno je uvesti u diskusiju i glavnu klasu aktivnosti. Ovde je neophodno napomenuti da će ova klasa sadržati i funkcionalnosti koje se odnose na sledeći pogled, DatePicker, sa ciljem da u narednom izlaganju ne bi došlo do ponavljanja koda.

```
package com.metropolitan.osnovnipogledidemo3;

import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.DatePicker;
import android.widget.TimePicker;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class MainActivity extends AppCompatActivity {

    TimePicker timePicker;
    DatePicker datePicker;
    int hour, minute;
    int yr, month, day;
    static final int TIME_DIALOG_ID = 0;
    static final int DATE_DIALOG_ID = 1;
    /** Poziva se kada se kreira aktivnost */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true);

        // showDialog(TIME_DIALOG_ID);
        datePicker = (DatePicker) findViewById(R.id.datePicker);

        //---uzima tekući datum---
        Calendar today = Calendar.getInstance();
        yr = today.get(Calendar.YEAR);
        month = today.get(Calendar.MONTH);
        day = today.get(Calendar.DAY_OF_MONTH);

        showDialog(DATE_DIALOG_ID);
    }
    @Override
    protected Dialog onCreateDialog(int id)
    {
        switch (id) {
```

```
            case TIME_DIALOG_ID:
                return new TimePickerDialog(
                        this, mTimeSetListener, hour, minute, false);
            case DATE_DIALOG_ID:
                return new DatePickerDialog(
                        this, mDateSetListener, yr, month, day);

        }
        return null;
    }
    private DatePickerDialog.OnDateSetListener mDateSetListener =
            new DatePickerDialog.OnDateSetListener()
            {
                public void onDateSet(
                        DatePicker view, int year, int monthOfYear, int dayOfMonth)
                {
                    yr = year;
                    month = monthOfYear;
                    day = dayOfMonth;
                    Toast.makeText(getBaseContext(),
                            "Izabrali ste : " + (month + 1) +
                                    "/" + day + "/" + year,
                            Toast.LENGTH_SHORT).show();
                }
            };
    private TimePickerDialog.OnTimeSetListener mTimeSetListener =
            new TimePickerDialog.OnTimeSetListener()
            {
                public void onTimeSet(
                        TimePicker view, int hourOfDay, int minuteOfHour)
                {
                    hour = hourOfDay;
                    minute = minuteOfHour;

                    SimpleDateFormat timeFormat = new SimpleDateFormat("hh:mm aa");
                    Date date = new Date(0,0,0, hour, minute);
                    String strDate = timeFormat.format(date);

                    Toast.makeText(getBaseContext(),
                            "Izabrali ste " + strDate,
                            Toast.LENGTH_SHORT).show();
                }
            };
    public void onClick(View view) {
        Toast.makeText(getBaseContext(),
                "Izabrani datum:" + (datePicker.getMonth() + 1) +
                        "/" + datePicker.getDayOfMonth() +
                        "/" + datePicker.getYear() + "\n" +
                        "Izabrano vreme:" + timePicker.getCurrentHour() +
                        ":" + timePicker.getCurrentMinute(),
                Toast.LENGTH_SHORT).show();
    }
```

```
}
```

TimePicker koristi standardni korisnički interfejs u kome je korisniku aplikacije omogućeno da podesi vreme. Po osnovnim podešavanjima, vreme se prikazuje u formatuAM/FM. Ukoliko se želi prikaz vremena u 24h formatu, neophodno je koristiti metodu setIs24HourView() (videti metodu onCreate()priloženog koda). Da bi vreme bilo programski prikazano, neophodno je upotrebiti metode getCurrentHour() i getCurrentMinute(), na sledeći način:

```
Toast.makeText(getBaseContext(),
"Izabrano vreme:" + timePicker.getCurrentHour() +
":" + timePicker.getCurrentMinute(),
Toast.LENGTH_SHORT).show();
```

PAKOVANJE TIMEPICKER POGLEDA U OKVIR DIJALOGA

Prosleđivanjem odgovarajućeg indikatora metodi showDialog, TimePicker pogled se smešta u okvir dijaloga.

Do sada je poznato da poziv dijaloga okvira inicira izvršavanje metode showDialog(). U konkretnom slučaju (videti priloženi kod) ovoj metodi je prosleđen atribut TIME_DIALOG_ID kojim je identifikovan izvor dijaloga okvira. Izvršavanjem metode showDialog(), za prikazivanje okvira, izvršava se i metoda onCreateDialog() u okviru koje se kreira instanca klase TimePickerDialog. Navedena instanca će preuzeti informacije od značaja, a to su u konkretnom slučaju sati i minuti. Na kraju, kao što je u kodu naznačeno, podaci će biti prezentovani u 24h formatu.

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

Sledećom slikom je prikazan okvir dijaloga za *TimePicker pogled.*

Slika 8.2 Dijalog okvir za TimePicker pogled (jedna od mogućih reprezentacija)

DATEPICKER POGLED

DatePicker pogled omogućava korisnicima selekciju datuma.

DatePicker pogled je po načinu korišćenja veoma sličan TimePickerpogledu. Primenom ovog pogleda, u određenoj aktivnosti, korisnicima je omogućeno da izaberu određeni datum, a to je aktivnost koju korisnici veoma često sreću i koriste u Android aplikacijama. Takođe, activity_main.xml datotekom definiše se DatePicker pogled odgovarajućim XML tagom.

```
<?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" >

  <DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Sledećom slikom prikazan je DatePicker pogled u Android aplikaciji.

Slika 8.3 DatePicker pogled

DATEPICKER POGLED - FUNKCIONISANJE I PAKOVANJE U DIJALOG OKVIR

DatePicker pogled koristi specifične metode za prikazivanje podataka o datumu.

Na sličan način kao TimePicker, DatePicker pogled koristi izvesne metode za prikazivanje podataka od značaja: dan, mesec i godina. Za prikazivanje ovih podataka odgovorne su metode getDayOfMonth(), getMonth() i getYear(), respektivno. Trebalo bi napomenuti da metoda *getMonth()* vraća vrednost 0 za mesec januar. Otuda, prilikom upotrebe ove metode vrši se inkrementacija rezultata da bi on imao valjanu vrednost. Primena ovih metoda je ugrađena u priloženi kod, a ovde će biti prezentovan samo konkretan deo poziva metoda.

```
"Izabrani datum:" + (datePicker.getMonth() + 1) +
"/" + datePicker.getDayOfMonth() +
"/" + datePicker.getYear() + "\n"
```

Kao i TimePicker pogled, DatePicker pogled može biti prikazan u okviru za dijalog. Identifikator za DatePicker definisan je kao DATE_DIALOG_ID koji se predaje metodi showDialog(). Po ovome, DatePicker i TimePicker funkcionišu na isti način. Nakon definisanja datuma, poziva se metoda onSetDate() koja kao rezultat vraća datum koji je korisnik izabrao (videti priloženi kod klase aktivnosti.

Kao jedinstveni primer, moguće je uklopiti obrađene TimePicker i DatePicker poglede u jedinstvenu activity_main.xml datoteku, koja bi u kombinaciji sa navedenom klasom aktivnosti uspešno zaokružila posmatrani i analizirani primer.

Sledećom slikom pokazan je TimePicker, ugrađen u konkretan dijalog okvir.

Slika 8.4 DatePicker dijalog okvir

PRIMER 15 - PODEŠAVANJE VREMENA I DATUMA

Praktičan rad na aplikacijama sa funkcionalnostima TimePicker i DatePicker

Zadatak:

1. Kreirati Android aplikaciju koja implementira poglede DatePicker i TimePicker.
2. Koristiti specifične dijaloge za ove poglede.

Zadatak uraditi po uzoru na sledeću sliku.

Slika 8.5 DatePicker i TimePicker

Datoteka korisničkog interfejsa data je 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"
    android:background="@drawable/metpozadina">

    <Button android:id="@+id/btnSet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Potvrdite podešavanja!!!"
        android:onClick="onClick" />

    <DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TimePicker android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Kod klase glavne aktivnosti dat je priloženim kodom:

```
package com.metropolitan.osnovnipogledidemo3;

import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.DatePicker;
```

```
import android.widget.TimePicker;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class MainActivity extends AppCompatActivity {

    TimePicker timePicker;
    DatePicker datePicker;
    int hour, minute;
    int yr, month, day;
    static final int TIME_DIALOG_ID = 0;
    static final int DATE_DIALOG_ID = 1;
    /** Poziva se kada se kreira aktivnost */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true);

        // showDialog(TIME_DIALOG_ID);
        datePicker = (DatePicker) findViewById(R.id.datePicker);

        //---uzima tekući datum---
        Calendar today = Calendar.getInstance();
        yr = today.get(Calendar.YEAR);
        month = today.get(Calendar.MONTH);
        day = today.get(Calendar.DAY_OF_MONTH);

        showDialog(DATE_DIALOG_ID);
    }
    @Override
    protected Dialog onCreateDialog(int id)
    {
        switch (id) {
            case TIME_DIALOG_ID:
                return new TimePickerDialog(
                        this, mTimeSetListener, hour, minute, false);
            case DATE_DIALOG_ID:
                return new DatePickerDialog(
                        this, mDateSetListener, yr, month, day);

        }
        return null;
    }
    private DatePickerDialog.OnDateSetListener mDateSetListener =
            new DatePickerDialog.OnDateSetListener()
            {
                public void onDateSet(
```

```
                        DatePicker view, int year, int monthOfYear, int dayOfMonth)
                {
                    yr = year;
                    month = monthOfYear;
                    day = dayOfMonth;
                    Toast.makeText(getBaseContext(),
                            "Izabrali ste : " + (month + 1) +
                                    "/" + day + "/" + year,
                            Toast.LENGTH_SHORT).show();
                }
            };
    private TimePickerDialog.OnTimeSetListener mTimeSetListener =
            new TimePickerDialog.OnTimeSetListener()
            {
                public void onTimeSet(
                        TimePicker view, int hourOfDay, int minuteOfHour)
                {
                    hour = hourOfDay;
                    minute = minuteOfHour;

                    SimpleDateFormat timeFormat = new SimpleDateFormat("hh:mm aa");
                    Date date = new Date(0,0,0, hour, minute);
                    String strDate = timeFormat.format(date);

                    Toast.makeText(getBaseContext(),
                            "Izabrali ste " + strDate,
                            Toast.LENGTH_SHORT).show();
                }
            };
    public void onClick(View view) {
        Toast.makeText(getBaseContext(),
                "Izabrani datum:" + (datePicker.getMonth() + 1) +
                        "/" + datePicker.getDayOfMonth() +
                        "/" + datePicker.getYear() + "\n" +
                        "Izabrano vreme:" + timePicker.getCurrentHour() +
                        ":" + timePicker.getCurrentMinute(),
                Toast.LENGTH_SHORT).show();
    }

}
```

ZADATAK 8 - DODAJTE U VAŠ PRIMER KONTROLE ZA PODEŠAVANJE DATUMA I VREMENA

Pokušajte sami.

Nastavite rad na primeru iz Zadatka 7 i dodajte u vaš primer kontrole za podešavanje datuma i vremena

Poglavlje 9

Primena List pogleda

LISTVIEW POGLED

Prikazivanje liste stavki u vertikalnoj skrol listi omogućeno je ListView pogledom.

List pogledi omogućavaju prikazivanje dugačkih listi u Android mobilnim aplikacijama. U Android operativnom sistemu podrška prikazivanju dugih listi realizovana je primenom specifičnih pogleda pod nazivima ***ListView*** i ***SpinnerView.*** Prvi zadatak je analiza mogućnosti i davanje rešenja za prikazivanje liste stavki u vertikalnoj skrol listi. Ovaj zahtev je moguće realizovati primenom pogleda ***ListView***.

U prvom koraku neophodno je pokazati kako se ovaj tip pogleda dodaje u korisnički interfejs Android aplikacije. Sledećim listingom je pokazan kod iz XML datoteke korisničkog interfejsa kojim je prikazana ***ListView*** kontrola.

```
<?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"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Prikaži izabrane stavke"
        android:onClick="onClick"/>
    <ListView
        android:id="@+id/android:list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Prikazivanje liste, kao specifičnog Android pogleda, ne bi bilo moguće bez glavne klase aktivnosti koja učitava korisnički interfejs, u glavnu aktivnost, u koji je ugrađen navedeni pogled. Klasa koja nasleđuje ListActivity i sama sadrži listu. Sledećim kodom prikazana je glavna klasa aktivnosti ovog primera.

```
package com.metropolitan.osnovnipogledidemo4;

import android.os.Bundle;
import android.app.ListActivity;
```

```
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends ListActivity {

    String[] predmeti;
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ListView lstView = getListView();
        lstView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
        lstView.setTextFilterEnabled(true);

        predmeti =
                getResources().getStringArray(R.array.predmeti_array);

        setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_checked, predmeti));
    }
    public void onListItemClick(
            ListView parent, View v, int position, long id)
    {
        Toast.makeText(this,
                "Izabrali ste " + predmeti[position],
                Toast.LENGTH_SHORT).show();
    }
    public void onClick(View view) {
        ListView lstView = getListView();
        String itemsSelected = "Izabrana stavka: \n";
        for (int i=0; i<lstView.getCount(); i++) {
            if (lstView.isItemChecked(i)) {
                itemsSelected += lstView.getItemAtPosition(i) + "\n";
            }
        }
        Toast.makeText(this, itemsSelected, Toast.LENGTH_LONG).show();
    }

}
```

LISTVIEW POGLED - FUNKCIONISANJE

Klasa za implementaciju ListView pogleda nasleđuje klasu ListActivity.

Kada se pogleda priloženi kod primera moguće je uočiti da klasa aktivnosti, koja je kreirana za prikaz liste stavki, nasleđuje klasu *ListActivity* koja je, pak, potklasa bazne klase Activity (AppCompatActivity za sve najnovije Android API verzije). Takođe, datoteku activity_main.xml nije potrebno modifikovati da bi ***ListView***pogled bio uključen. Klasa koja nasleđuje

ListActivity već sadrži ListView.Zbog toga, uonCreate() metodi nije potrebno pozvati metodu setContentView() sa ciljem učitavanja korisničkog interfejsa iz datoteke activity_main.xml. U onCreate() metodi se koristi setListAdapter() za popunjavanje ekrana aktivnošću koja odgovara ***ListView*** pogledu. Komponenta ArrayAdapter upravlja nizom stringova prezentovanih ListView pogledom. Ako se pogleda priloženi kod, *ListView* pogled je definisan da bude prikazan u jednostavnom simple_list_item_checkedrežimu.

Na kraju, metoda onListItemClick() se inicira uvek kada korisnik klikne na neku stavku iz liste. Sledećom slikom prikazan je korisnički interfejs koji odgovara tekućem primeru.

Slika 9.1 ListView pogled sa izabranim stavkama

PODEŠAVANJE LISTVIEW POGLEDA

ListView pogled podržava različite prikaze koji se mogu posebno podešavati.

ListView pogled podržava različite prikaze koji se mogu posebno podešavati. U konkretnom primeru definisano je da iz liste, u svakom trenutku izvršavanja programa, može biti izabrano više stavki korišćenjem konstante ***CHOICE_MODE_MULTIPLE***. Ovo je moguće prilagoditi na još dva načina: nije dozvoljeno selektovanje iz liste ili moguće je selektovati samo jednu stavku liste.

Kreiranje liste i izbor scenarija biranja iz liste prikazan je sledećom onCreate () metodom čiji je kod izolovan iz glavne klase aktivnbosti.

```
String[] predmeti;
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ListView lstView = getListView();
        lstView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
        lstView.setTextFilterEnabled(true);

        predmeti = getResources().getStringArray(R.array.predmeti_array);

        setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_checked, predmeti));
    }
```

Lista ne dozvoljava izbor ukoliko se u metodi setChoiceMode() kao argument javi konstanta ***CHOICE_MODE_NONE***, a iz liste je moguće izabrati samo jednu stavku ukoliko je ovoj metodi pridružena konstanta ***CHOICE_MODE_SINGLE***.

Da bi navedenu metodu bilo moguće koristiti, neophodno je kreirati objekat klase ListView , za čije je učitavanje neophodno pozvati metodu getListView().

Još jedan koristan element, koji je u primeru bio korišćen, je podrška za filtriranje. Nakon omogućavanja metodesetTextFilterEnabled(), otvara se mogućnost za unost teksta pomoću tastature, a***ListView*** prikaz će automatski biti modifikovan da se filtriraju samo one stavke koje odgovaraju unetim podacima: ListView.setTextFilterEnabled(true);

ČUVANJE STAVKI U STRING.XML DATOTECI

Stavke liste moguće je čuvati izvan klase aktivnosti aplikacije.

U jednostavnim aplikacijama moguće je čuvati podatke kao niz u JAVA klasi aktivnosti aplikacije. Međutim, u realnim uslovima, poželjnije je čuvati stavke u bazama podataka ili, pak, u datotekama. U konkretnom primeru, odabrana je datoteka strings.xml da sačuva članove liste do njihovog poziva.

Budući da su nazivi predmeta sačuvani ustring.xml datoteci, njih je moguće učitati, na veoma jednostavan način, upotrebom metode getResources() u klasi aktivnosti aplikacije (videti priloženi kod klase aktivnosti).

Kao što se vidi iz naziva datoteke, niz stavki koje će biti spakovane u listu, prilikom učitavanja glavne aktivnosti aplikacije, obezbeđene su primenom adekvatnih XML instrukcija koje su prikazane sledećim listingom.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">ListView demonstracija!</string>
    <string name="app_name">ListView</string>
    <string-array name="predmeti_array">
        <item>Razvoj mobilnih apliacija</item>
        <item>Web sistemi 1</item>
        <item>Web sistemi 2</item>
        <item>Distribuirani sistemi</item>
        <item>Skripting jezici</item>
        <item>Razvoj igara</item>
        <item>Java programiranje 1</item>
        <item>Java programiranje 2</item>
        <item>Programiranje u C#</item>
        <item>Operativni sistemi</item>
        <item>Matematika</item>
    </string-array>
</resources>
```

PROVERA SELEKTOVANIH STAVKI

Metoda isItemChecked() ispituje da li je stavka izabrana ili ne.

U ovom delu lekcije je istaknuto kako se prikazuje ***ListView*** pogled koji zauzima celu površinu za prikazivanje određene aktivnosti i nije bilo potrebe za dodavanjem***<ListView>*** elementa u activity_main.xml. Da bi prostor predviđen za celokupnu aktivnost bio delimično ispunjen,

neophodno je dodati ***<ListView>*** element u activity_main.xml datoteku na način prikazan priloženim programskim kodom za ovaj primer. Ovaj element mora da bude snabdeven i id atributom čija je vrednost: @+id/android:list. To znači da je u ovom slučaju neophodno učitati metodu setContentView() (videti priloženi kod klase aktivnosti) da bi interfejs iz datoteke activity_main.xml bio učitan.

Metodom isItemChecked() proverava se da li stavka u listi izabrana, ili ne, na sledeći način:

```
public void onClick(View view) {
        ListView lstView = getListView();
        String itemsSelected = "Izabrana stavka: \n";
        for (int i=0; i<lstView.getCount(); i++) {
            if (lstView.isItemChecked(i)) {
                itemsSelected += lstView.getItemAtPosition(i) + "\n";
            }
        }
```

Delimično prikazana lista, u učitanoj aktivnosti, prikazana je sledećom slikom.

Slika 9.2 Delimično učitana lista

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

SPINNER POGLED

SpinnerView pogled omogućava prikazivanje jedne po jedne stavke iz liste.

Ukoliko je neophodno da se pored liste stavki, u odgovarajućoj aktivnosti, prikažu i drugi pogledi, a da se ne zauzme cela površina ekrana kao kod ***ListView***pogleda, trebalo bi koristiti ***SpinnerView*** prikaz. Ovaj prikaz omogućava prikazivanje jedne po jedne stavke iz liste i odgovara opšte prihvaćenom terminu za ovakav tip kontrole - *padajuća lista.*

Padajući listu, kao kontrolu, na jednostavan način je moguće ugraditi u activity_main.xml datoteku. Navedeno je realizovano kodom iz sledećeg listinga.

```
<?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" >

    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawSelectorOnTop="true" />
```

```
</LinearLayout>
```

Takođe, za učitavanje padajuće liste, izbor njenih stavki i realizovanje odgovarajuće programske logike, zadužena je glavna klasa aktivnosti. Kod glavne klase aktivnosti prikazan je sledećim listingom.

```
package com.metropolitan.osnovnipogledidemo5;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    String[] predmeti;

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

        predmeti = getResources().getStringArray(R.array.predmeti_array);
        Spinner s1 = (Spinner) findViewById(R.id.spinner1);

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_single_choice, predmeti);

        s1.setAdapter(adapter);
        s1.setOnItemSelectedListener(new OnItemSelectedListener()
        {
            @Override
            public void onItemSelected(AdapterView<?> arg0,
                                       View arg1, int arg2, long arg3)
            {
                int index = arg0.getSelectedItemPosition();
                Toast.makeText(getBaseContext(),
                        "Izabrali ste stavku : " + predmeti[index],
                        Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onNothingSelected(AdapterView<?> arg0) { }
        });
    }
```

```
}
```

SPINNER POGLED - FUNKCIONISANJE

Za SpinnerView neophodno je implementirati njegovu metodu onNothingSelected().

Tekući primer funckioniše veoma slično prethodnom - primeru ***ListView*** pogleda. Za početak je neohodno implementirati metodu onNothingSelected().Ova metoda se izvršava kada korisnik pritisne tasterBackna mobilnom uređaju i na taj način prekida dalje prikazivanje liste.

Umesto stavki za ArrayAdapter, kojima se prikazuje jednostavna lista, moguće je prikazati elemente korišćenjem radio tastera. Da bi to bilo moguće neophodno je modifikovati drugi parametar u konstruktoru ArrayAdapterklase. Navedeno je moguće prikazati kroz sledeći deo koda koji je izolovan iz glavne klase aktivnosti.

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_single_choice, predmeti);
```

U nastavku je moguće testirati funkcionalnosti kreirane ***Spinner*** kontrole. Izborom opcije Run app (ili Shift + F10) u Android Studio razvojnom okruženju vrši se debagovanje i pokretanje aplikacije u odgovarajućem Android emulatoru, u ovom slučaju u Genymotion emulatoru.

Sledećom slikom je prikazan izgled kontrole Spinner koja je učitana u glavnu aktivnost aplikacije.

Slika 9.3 Spinner pogled u Android aplikaciji

PRIMER 16 - PRIKAZIVANJE PODATAKA U FORMI LISTE

Vežbanje kreiranja primera sa ListView pogledima.

Zadatak:

1. Kreirati Android aplikaciju koja u glavnu aktivnost učitava listu predmeta Fakulteta informacionih tehnologija.
2. Omogućiti višestruku mogućnost biranja iz liste;
3. Vratiti informaciju Toast klasom o izabranim stavkama iz liste.

Uzor za aplikaciju tražiti u sledećoj slici.

Slika 9.4 ListView

GUI datoteka data je sledćim 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"
    android:background="@drawable/metpozadina">
    <Button android:id="@+id/btn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Prikaži izabrane stavke"
        android:onClick="onClick"/>
    <ListView
        android:id="@+id/android:list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Klasa glavne aktivnosti data je sledećim kodom:

```
package com.metropolitan.osnovnipogledidemo4;

import android.os.Bundle;
import android.app.ListActivity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends ListActivity {

    String[] predmeti;
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ListView lstView = getListView();
        lstView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
        lstView.setTextFilterEnabled(true);

        predmeti = getResources().getStringArray(R.array.predmeti_array);

        setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_checked, predmeti));
    }
    public void onListItemClick(
            ListView parent, View v, int position, long id)
    {
        Toast.makeText(this,
                "Izabrali ste " + predmeti[position],
                Toast.LENGTH_SHORT).show();
    }
```

```
    public void onClick(View view) {
        ListView lstView = getListView();
        String itemsSelected = "Izabrana stavka: \n";
        for (int i=0; i<lstView.getCount(); i++) {
            if (lstView.isItemChecked(i)) {
                itemsSelected += lstView.getItemAtPosition(i) + "\n";
            }
        }
        Toast.makeText(this, itemsSelected, Toast.LENGTH_LONG).show();
    }

}
```

PRIMER 17 - PRIKAZIVANJE PODATAKA U FORMI PADAJUĆE LISTE

Vežbanje pogleda Spinner

Prethodni zadatak modifikovati tako da predmeti sa našeg fakulteta budu spakovani u padajuću listu. Obezbediti da je u svakom trenutku moguće izabrati samo jedan predmet.

Uzor za izradu zadatka tražiti u sledećoj slici.

Slika 9.5 Pogled Spinner

GUI datoteka data je sledećim XML 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"
    android:background="@drawable/metpozadina">

    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawSelectorOnTop="true" />

</LinearLayout>
```

Klasa glavne aktivnosti data je sledećim JAVA kodom:

```
package com.metropolitan.osnovnipogledidemo5;

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

```
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    String[] predmeti;

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

        predmeti = getResources().getStringArray(R.array.predmeti_array);
        Spinner s1 = (Spinner) findViewById(R.id.spinner1);

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_single_choice, predmeti);

        s1.setAdapter(adapter);
        s1.setOnItemSelectedListener(new OnItemSelectedListener()
        {
            @Override
            public void onItemSelected(AdapterView<?> arg0,
                                       View arg1, int arg2, long arg3)
            {
                int index = arg0.getSelectedItemPosition();
                Toast.makeText(getBaseContext(),
                        "Izabrali ste stavku : " + predmeti[index],
                        Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onNothingSelected(AdapterView<?> arg0) { }
        });
    }

}
```

ZADATAK 9 - DODAJTE LISTU U VAŠ PRIMER

Pokušajte sami

- Dodati u vašu Android aplikaciju mogućnost učitavanja liste stavki prema vašem izboru.
- Omogućiti višestruku mogućnost biranja iz liste;
- Vratiti informaciju Toast klasom o izabranim stavkama iz liste.

Poglavlje 10

Primena specijalizovanih fragmenata

LISTFRAGMENT KLASA

ListFragment klasa definiše fragment koji sadrži ListView pogled.

Kao što je naznačeno, u jednoj od prethodnih lekcija, fragmenti su mini-aktivnosti koje imaju sopstvene životne cikluse. Drugim rečima, fragment može biti tretiran kao podaktivnost neke glavne aktivnosti. Da bi fragment bio kreiran, on mora da bude podržan klasom koja nasleđuje baznu klasu Fragment. Osnovna klasa poseduje i različite specijalne varijante koje se koriste da bi realizovale brojne specifične programske zahteve. Pored osnovne klase omogućeno je i nasleđivanje izvesnih njenih potklasa sa ciljem kreiranja specijalizovanih fragmenata. Potklase klase *Fragment su: ListFragment, DialogFragment i PreferenceFragment.*

ListFragment klasa definiše fragment koji sadrži ListView pogled. *ListFragment* je veoma koristan alat jer omogućava prikazivanje liste stavki na ekranu zajedno sa ostalim elementima korisničkog interfejsa, a to može biti i još neka lista stavki. Da bi fragment za prikazivanje liste bio kreiran, neophodno je da ključna klasa nasledi klasu ListFragment i da budu realizovane XML datoteke glavnog korisničkog interfejsa i fragmenata.

```
<?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">

    <ListView
        android:id="@id/android:list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:drawSelectorOnTop="false"/>

</LinearLayout>
```

Fragmenti, kao i ostale kontrole, mogu da predstavljaju deo glavnog korisničkog interfejsa određenog datotekom activity_main.xml. Kod ove datoteke je prikazan 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="horizontal" >

    <fragment
        android:name="com.metropolitan.listfragmentdemo.Fragment1"
        android:id="@+id/fragment1"
        android:layout_weight="0.5"
        android:layout_width="0dp"
        android:layout_height="200dp" />

    <fragment
        android:name="com.metropolitan.listfragmentdemo.Fragment1"
        android:id="@+id/fragment2"
        android:layout_weight="0.5"
        android:layout_width="0dp"
        android:layout_height="300dp" />

</LinearLayout>
```

JAVA DATOTEKE APLIKACIJE ZA DEMONSTRACIJU LISTFRAGMENT

Pored glavne JAVA klase, koja je zadužena za aktivnosti, neophodno je kreirati i JAVA klasu koja nasleđuje klasu ListFragment.

Svaka Android aplikacija mora da sadrži glavnu JAVA klasu aktivnosti. Zadatak ove klase je, u velikom broju slučajeva, da obezbedi izvršavanje metode setContentView() kojom se učitava korisnički interfejs definisan u activity_main.xml datoteci.

U ovom primeru, klasa aktivnosti će imati isključivo navedeni zadatak, a to je prikazano sledećim kodom.

```
package com.metropolitan.listfragmentdemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    /** Poziva se kada se aktivnost kreira. */

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

Da bi fragment za prikazivanje listi bio implementiran, neophodno je kreirati njegovu klasu. Kao što je napomenuto, on mora da nasledi osnovnu ListFragment klasu, a to je pokazano sledećim kodom.

```
package com.metropolitan.listfragmentdemo;

import android.app.ListFragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
/**
 * Created by Vladimir Milicevic on 27.10.2016..
 */

public class Fragment1 extends ListFragment{
    String[] predmeti = {
            "Razvoj mobilnih apliacija",
            "Web sistemi 1",
            "Web sistemi 2",
            "Distribuirani sistemi",
            "Skripting jezici",
            "Razvoj igara",
            "Java programiranje 1",
            "Java programiranje 2",
            "Programiranje u C#",
            "Operativni sistemi",
            "Matematika"
    };
    @Override
    public View onCreateView(LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment1, container, false);
    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setListAdapter(new ArrayAdapter<String>(getActivity(),
                android.R.layout.simple_list_item_1, predmeti));
    }
    public void onListItemClick(ListView parent, View v,
                                int position, long id)
    {
        Toast.makeText(getActivity(),
                "Izabrali ste " + predmeti[position],
                Toast.LENGTH_SHORT).show();
    }
}
```

APLIKACIJA ZA DEMONSTRACIJU LISTFRAGMENT - FUNKCIONISANJE

Aplikacija pokazuje dva fragmenta za prikaz listi, postavljena jedan pored drugog.

Kreiranje aplikacije, čiji je kod priložen, teče u sledećim koracima:

- Kreirana je XML datoteka za fragment koja sadrži <ListView> element;
- Kreirana je JAVA klasa koja nasleđuje klasu *ListFragment*;
- U ovoj klasi je kreiran niz predmeta (polje) koji će biti upisan u listu;
- Metoda onCreate() koristi metodu setListAdapter()čiji je zadatak punjenje liste sadržajem polja; *ArrayAdapter* upravlja nizom stringova koji će biti prikazani u *ListView* rasporedu jednostavnim režimom *simple_list_item_1* (videti kod fragment klase).
- Metoda onListItemClick() poziva se prilikom svakog klika na stavku u *ListView* prikazu;
- Na kraju, datotekom activity_main.xml je definisan korisnički interfejs koji poseduje dva fragmenta za prikazivanje listi. Za svaki fragment određena je drugačija visina prostora koji će zauzeti (videti kod datoteke activity_main.xml).

Klikom na Shift + F10, ili izborom opcijeRun app, u Android Studio razvojnom okruženju, primer se prevodi i pokreće emulatorom. Rezultat pokretanja aplikacije prikazan je sledećom slikom.

Slika 10.1 Prikaz dva fragmenta tipa ListFragment u aktivnosti

DIALOGFRAGMENT KLASA

Fragmenti za dijalog su veoma korisni kada bi trebalo dobiti odgovor korisnika pre nego što se nastavi izvršavanje Android aplikacije.

Postoji još jedan tip fragmenata koji je moguće kreirati i čiji je zadatak da od korisnika dobije neki odgovor pre nego što se nastavi izvršavanje Android aplikacije. Veoma često se sreće, u savremenim Android aplikacijama, situacija u kojoj se odvija "dijalog" između korisnika i aplikacije. Dijalog u velikom broju slučajeva predstavlja podaktivnost aktivnosti kojom se obavlja programska logika konkretne Android aplikacije. Ovakvi fragmeti su izvedeni iz klase DialogFragment i nazivaju se fragmentima za prikazivanje dijaloga.

Za analizu i demonstraciju funkcionisanja ovakvog tipa fragmenta biće neophodno uvesti odgovarajući primer. Neka prvo bude definisana XML datoteka glavnog korisničkog interfejsa pod nazivom activity_main.xml. Njen kod je prikazan sledećim jednostavnim 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="DialogFragment - primer" />

</LinearLayout>
```

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

DIALOGFRAGMENT KLASA – JAVA KLASE PRIMERA

Da bi fragment za prikazivanje dijaloga bio kreiran, neophodno je da ključna klasa nasledi klasu DialogFragment.

Svaki fragment da bi bio implementiran, mora da poseduje vlastitu klasu. Klasa fragmenta za prikazivanje dijaloga mora da nasleđuje baznu klasu DialogFragment. Sledećim listingom je prikazan JAVA programski kod klase Fragment 1.

```
package com.metropolitan.dialogfragmentdemo;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.content.DialogInterface;
import android.os.Bundle;

/**
 * Created by Vladimir Milicevic on 27.10.2016..
 */

public class Fragment1 extends DialogFragment {
    static Fragment1 newInstance(String title) {
        Fragment1 fragment = new Fragment1();
        Bundle args = new Bundle();
        args.putString("Naslov", title);
        fragment.setArguments(args);
        return fragment;
    }
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        String title = getArguments().getString("Naslov");
        return new AlertDialog.Builder(getActivity())
                .setIcon(R.mipmap.ic_launcher)
                .setTitle(title)
                .setPositiveButton("Da",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                                int whichButton) {
                                ((MainActivity)
```

```
                                        getActivity()).doPositiveClick();
                            }
                        })
                .setNegativeButton("Ne",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                                int whichButton) {
                                ((MainActivity)
                                        getActivity()).doNegativeClick();
                            }
                        }).create();
    }
}
```

Za funkcionisanje svake Android aplikacije najveće zasluge ima glavna klasa aktivnosti. Za zaokruživanje primera primene fragmenata dijaloga neophodno je u, aktuelnom projektu, definisati i glavnu klasu aktivnosti.

Glavna klasa aktivnosti aplikacije data je sledećim programskim kodom.

```
package com.metropolitan.dialogfragmentdemo;

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

public class MainActivity extends AppCompatActivity {

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

        Fragment1 dialogFragment = Fragment1.newInstance(
                "Da li želite da nastavite?");
        dialogFragment.show(getFragmentManager(), "dialog");
    }

    public void doPositiveClick() {
        //---Korisnik je kliknuo Da---
        Log.d("DialogFragment - primer", "Korisnik je kliknuo Da");
    }

    public void doNegativeClick() {
        //---Korisnik je kliknuo Ne---
        Log.d("DialogFragment - primer", "Korisnik je kliknuo Ne");
    }

}
```

FUNKCIONISANJE DIALOGFRAGMENT APLIKACIJA

Prozorom sa porukom i tasterima za reakciju, aplikacija čeka na odgovor korisnika

U nastavku je neophodno izvršiti testiranje kreirane aplikacije sa ciljem demonstriranja funkcionalnosti fragmenata sa dijalogom.

Funkcionisanje aplikacije, koja implementira fragment sa dijalogom, podrazumeva da su ispunjeni sledeći uslovi:

- Kreirana fragment klasa koja nasleđuje klasu *Fragment*;
- Kreiran okvir za dijalog koji prikazuje poruku i odgovarajuće tastere;
- Metodom newInstance() omogućeno je kreiranje nove instance fragmenta i prihvatanje string argumenta koji će biti prikazan kao poruka u okviru za dijalog (videti priloženi kod klase Fragment1);
- Postojanje onCreateDialog() metode, koja se izvršava nakon onCreate() metode, a preonCreateView() metode (videti priloženi kod klase Fragment1);
- Kreirana instanca fragmenta koja izvršava metodu show(): dialogFragment.show(getFragmentManager(), "dialog");
- Na kraju, neophodno je implementirati metode doPositiveClick() i doNegativeClick() za rukovanje aktivnostima koje su rezultat klika na dugmad *Da* i *Ne,* respektivno (videti priloženi kod klase Fragment1).

Kao poslednji korak, u Android Studio IDE razvojnom okruženju, neophodno je izabrati opciju Run app (ili Shift + F10) za pokretanje i testiranje aplikacije u izabranom Android emulatoru.

Izgled učitane aplikacije prikazan je sledećom slikom.

Slika 10.2 DialogFragment

PREFERENCEFRAGMENT KLASA

U Android operativnom sistemu moguće je koristiti PreferenceActivity baznu klasu za prikazivanje aktivnosti koja korisniku omogućava opcije za prilagođavanje.

Izbor opcija za personalizovanje aplikacija, vlastitim potrebama i navikama, omogućen je u Android operativnom sistemu. Bazna klasa, kojom su omogućene navedene funkcionalnosti, naziva se PreferenceActivity klasa. Novije verzije Android operativnog sistema donose još jednu klasu kojom je obezbeđeno da kreirana aplikacija ima navedene funkcionalnosti. Ova klasa je poznata pod nazivom PreferenceFragment klasa. Za demonstraciju, biće korišćen primer u kome će activity_main.xml datoteka ostati nepromenjena, ali će zato, u folderu koji će biti nazvan xml, biti kreirana datoteka preferences.xmlu kojoj će biti definisane kategorije i njihovi sadržaji namenjeni podešavanju u Android aplikaciji. Sledećom slikom prikazan je položaj foldera xml i njegove datoteke preferences.xml u hijerarhiji projekta.

Slika 10.3 Datoteka preferencija

Datoteka preferences.xml, kao što je napomenuto, prikazuje više kategorija na ekranu. U svakoj od kategorija postoje različiti sadržaji kojima se podešavaju vrednosti u skladu sa preferencijama korisnika. Kod datoteke preferences.xml prikazan je sledećim listingom.

```
<?xml version="1.0" encoding="utf-8"?>


        <CheckBoxPreference
            android:title="Checkbox"
            android:defaultValue="false"
            android:summary="True ili False"
            android:key="checkboxPref" />
    </PreferenceCategory>

        <EditTextPreference
            android:name="EditText"
            android:summary="Unesite string"
            android:defaultValue="[Unesite string ovde]"
            android:title="Promenite tekst"
            android:key="editTextPref" />
        <RingtonePreference
            android:name="Ringtone Preference"
            android:summary="Izaberite melodiju zvona"
            android:title="Melodija zvona"
            android:key="ringtonePref" />

            <EditTextPreference
                android:name="EditText"
                android:summary="Unesite string"
                android:title="Unesite string (drugi ekran)"
                android:key="secondEditTextPref" />
        </PreferenceScreen>
    </PreferenceCategory>
</PreferenceScreen>
```

PREFERENCEFRAGMENT KLASA – JAVA KLASE PRIMERA

Fragment klasa nasleđuje baznu klasu PreferenceFragment

Osnovna klasa aplikacije, klasa aktivnosti, neophodna je za funkcionisanje Android aplikacije koja se razvija. Njen konkretan zadatak će, pored učitavanja glavne aktivnosti, biti i da omogući upravljanje fragmentom preferencija pomoću objekata tipa: FragmentManager, FragmentTransaction i kreirane klase Fragment1.

```
package com.metropolitan.preferencefragmentdemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.app.FragmentManager;
import android.app.FragmentTransaction;

public class MainActivity extends AppCompatActivity {

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

        FragmentManager fragmentManager = getFragmentManager();
        FragmentTransaction fragmentTransaction =
                fragmentManager.beginTransaction();
        Fragment1 fragment1 = new Fragment1();
        fragmentTransaction.replace(android.R.id.content, fragment1);
        fragmentTransaction.addToBackStack(null);
        fragmentTransaction.commit();
    }

}
```

Fragment klasa mora da nasleđuje baznu klasu PreferenceFragment da bi funkcije prilagođavanja aplikacije korisničkim potrebama bile omogućene. Takođe, ova klasa ima zadatak da u fragment učita preferencije iz datoteke pod nazivom preferences.xml. Sledećim listingom priložen je kod klase Fragment1 koja realizuje navedene funkcionalnosti.

```
package com.metropolitan.preferencefragmentdemo;

import android.os.Bundle;
import android.preference.PreferenceFragment;
/**
 * Created by Vladimir Milicevic on 27.10.2016..
 */

public class Fragment1 extends PreferenceFragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //---Učitava preference iz XML datoteke---
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

PREFERENCEFRAGMENT KLASA – DEMONSTRACIJA PRIMERA

Na ekranu se nalaze opcije koje omogućavaju korisniku da izabere željena podešavanja.

Izborom opcije Run app (ili Shift + F10), u Android Studio IDE razvojnom okruženju, pokreće se testiranje aplikacije u izabranom Android emulatoru. Kao rezultat javlja se učitavanje sledeće aktivnost u ekran emulatora.

Slika 10.4 Glavna aktivnost sa kategorijama

Ako se izabere opcija Promenite tekst, otavara se dijalog okvir sa opcijom za unos željenog teksta. Nakon unosa ovog strinka i izbora opcije OK, u preferencijama se menja podešena vrednost za ovu opciju. Ovu aktivnost kreiranje aplikacije moguće je videti na sledećoj slici.

Slika 10.5 Izmena tekst preferencija

Ako se izabere opcija Drugi ekran preferencija, biće prikazan drugi ekran sa svojim opcijama. Među ovim opcijama postoji i opcija za izmenu teksta koja je već prikazana. Dalje, na prvom ekranu je moguće primetiti preferencije koje se veoma često koriste, a odnose se na podešavanje melodije zvona. Nov ekran se realizuje elementom <PreferenceScreen>, a kategorije u okviru ekrana elementima <PreferenceCategory>.

Slika 10.6 Drugi ekran preferencija

ČUVANJE I LOCIRANE PREFERENCIJA

Sva korisnička podešavanja biće sačuvana u posebnom dokumentu.

Sva korisnička podešavanja biće sačuvana u posebnom dokumentu. Ukoliko se iskoristi opcija File Exprorer (dostupna je u Dalvik Debug Monitor Service - DDMS prikazu) moguće je locirati datoteku sa snimljenim preferencijama. Potrebno je pratiti sledeću putanju: data/data, a zatim u tom folderu locirati naziv paketa (korišćeno je com.metropolitan) sa nazivom ciljanog projekta. Ulaskom u ovaj podfolder, locira se folder shered_prefs u kojem je spakovana datoteka sa sačuvanim preferencijama (slika broj 8).

Da bi File Explorer bio dostupan, neophodno je u opciji Tools, Android Studio IDE razvojnog okruženja, izabrati opciju Android, a zatim i Android Device Monitor, kao što je prikazano sledećom slikom.

Slika 10.7 Pristup File Exploreru u Android Studio razvojnom okruženju

Sledećom slikom prikazana je lokacija datoteka sa preferencijama koja se čuva u kreiranom emulatoru, a koja je dostupna u Android Studio IDE (ili Eclipse IDE) razvojnom okruženju preko prikazane FIle Explorer funkcionalnosti.

Slika 10.8 Datoteka sa preferencijama

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

PREFERENCEFRAGMENT KLASA - FUNKCIONISANJE

Za učitavanje datoteke sa preferencijama, u fragment, koristi se specijalna metoda.

Da bi preferencije bilo moguće kreirati, u nekoj Android aplikaciji, neophodno je prvo kreirati XML datoteku preferencija u kojoj se definišu različite perzistentne stavke aplikacije.

Da bi fragment preferencija bio kreiran, neophodno je kreirati klasu koja nasleđuje baznu klasu PreferenceFragment.

Učitavanje datoteke sa preferencijama u fragment omogućeno je metodom addPreferencesFromSource() koja je ugrađena u metodu onCreate()fragment klase aplikacije. Izolovan kod ove metode, klase Fragment1, moguće je prikazati sledećim listingom.

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //---Učitava preference iz XML datoteke---
        addPreferencesFromResource(R.xml.preferences);
    }
```

Na kraju, neophodno je pakovanje fragmenta preferencija na pozadinski stek pomoću metode addToBackStack()u klasi glavne aktivnosti. Na ovaj način, omogućeno je uklanjanje fragmenta klikom na taster Back. Navedena funkcionalnost je realizovana kodom iz sledećeg listinga, izdvojenog iz klase MainActivity.java.

```
FragmentManager fragmentManager = getFragmentManager();
        FragmentTransaction fragmentTransaction =
                fragmentManager.beginTransaction();
        Fragment1 fragment1 = new Fragment1();
        fragmentTransaction.replace(android.R.id.content, fragment1);
        fragmentTransaction.addToBackStack(null);
        fragmentTransaction.commit();
```

PRIMER 18 - VEŽBANJE KREIRANJA FRAGMENATA LISTE

Korišćenje specijalnih fragmenata za liste.

Zadatak: U dva fragmenta, različitih dimenzija, spakovati liste predmeta našeg fakulteta.

Sledećim kodom data je glavna GUI datoteka:

```
<?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"
    android:background="@drawable/metpozadina">

    <fragment
        android:name="com.metropolitan.listfragmentdemo.Fragment1"
        android:id="@+id/fragment1"
        android:layout_weight="0.5"
        android:layout_width="0dp"
        android:layout_height="200dp"
        />

    <fragment
        android:name="com.metropolitan.listfragmentdemo.Fragment1"
        android:id="@+id/fragment2"
        android:layout_weight="0.5"
        android:layout_width="0dp"
        android:layout_height="300dp" />

</LinearLayout>
```

Za fragment je moguće definisati sledeći GUI:

```
<?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">

    <ListView
        android:id="@id/android:list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:drawSelectorOnTop="false"/>

</LinearLayout>
```

Glavna klasa aktivnosti može da bude u podrazumevanom, osnovnom obliku:

```
package com.metropolitan.listfragmentdemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    /** Poziva se kada se aktivnost kreira. */

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

Klasa fragmenta ima sledeći kod:

```
package com.metropolitan.listfragmentdemo;

import android.app.ListFragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
/**
 * Created by Vladimir Milicevic on 27.10.2016..
 */

public class Fragment1 extends ListFragment{
    String[] predmeti = {
            "Razvoj mobilnih apliacija",
            "Web sistemi 1",
            "Web sistemi 2",
            "Distribuirani sistemi",
            "Skripting jezici",
            "Razvoj igara",
            "Java programiranje 1",
            "Java programiranje 2",
            "Programiranje u C#",
            "Operativni sistemi",
            "Matematika"
    };
    @Override
    public View onCreateView(LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment1, container, false);
    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setListAdapter(new ArrayAdapter<String>(getActivity(),
                android.R.layout.simple_list_item_1, predmeti));
    }
    public void onListItemClick(ListView parent, View v,
                                int position, long id)
    {
        Toast.makeText(getActivity(),
                "Izabrali ste " + predmeti[position],
                Toast.LENGTH_SHORT).show();
    }
}
```

PRIMER 19 - VEŽBANJE KREIRANJA FRAGMENATA DIJALOGA

Korišćenje specijalnih fragmenata za dijaloge.

Zadatak: Glavna aktivnost čeka da se obavi komunikacija korisnika sa fragmentom dijaloga.

Glavna GUI datoteka može da ima sledeći oblik:

```
<?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="DialogFragment - primer" />

</LinearLayout>
```

Glavna klasa aktivnosti data je sledećim kodom:

```
package com.metropolitan.dialogfragmentdemo;

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

public class MainActivity extends AppCompatActivity {

    /** Poziva se kada se kreira aktivnost. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

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

        Fragment1 dialogFragment = Fragment1.newInstance(
                "Da li želite da nastavite?");
        dialogFragment.show(getFragmentManager(), "dialog");
    }

    public void doPositiveClick() {
        //---Korisnik je kliknuo Da---
        Log.d("DialogFragment - primer", "Korisnik je kliknuo Da");
    }

    public void doNegativeClick() {
        //---Korisnik je kliknuo Ne---
        Log.d("DialogFragment - primer", "Korisnik je kliknuo Ne");
    }

}
```

Fragment klasa data je sledećim kodom:

```
package com.metropolitan.dialogfragmentdemo;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.content.DialogInterface;
import android.os.Bundle;

/**
 * Created by Vladimir Milicevic on 27.10.2016..
 */

public class Fragment1 extends DialogFragment {
    static Fragment1 newInstance(String title) {
        Fragment1 fragment = new Fragment1();
        Bundle args = new Bundle();
        args.putString("Naslov", title);
        fragment.setArguments(args);
        return fragment;
    }
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        String title = getArguments().getString("Naslov");
        return new AlertDialog.Builder(getActivity())
                .setIcon(R.mipmap.ic_launcher)
                .setTitle(title)
                .setPositiveButton("Da",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                                int whichButton) {
                                ((MainActivity)
                                        getActivity()).doPositiveClick();
                            }
```

```
                        })
                .setNegativeButton("Ne",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                                int whichButton) {
                                ((MainActivity)
                                        getActivity()).doNegativeClick();
                            }
                        }).create();
    }
}
```

PRIMER 20 - VEŽBANJE KREIRANJA I UPRAVLJANJA FRAGMENTIMA PREFERENCIJA

Utvrđivanje stečenog znanja o fragmentima preferencija sa časova predavanja

Zadatak:

Kreirati Android aplikaciju za kreiranje i upravljanje fragmentima preferencija:

1. Aplikacija ima glavnu klasu u kojoj se realizuje upravljanje fragmentima preferencija;
2. Aplikacija poseduje i klasu fragmenta preferencija;
3. Kreirati nov folder u res folderu za definisanje xml datoteke preferencija;
4. Definisati ekrane, kategorije i elemente preferencija u ovoj datoteci;
5. Testirati aplikaciju u ogovarajućem emulatoru.

Sledećim kodom data je klasa glavne aktivnosti.

```
package com.metropolitan.preferencefragmentdemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.app.FragmentManager;
import android.app.FragmentTransaction;

public class MainActivity extends AppCompatActivity {

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

        FragmentManager fragmentManager = getFragmentManager();
        FragmentTransaction fragmentTransaction =
                fragmentManager.beginTransaction();
        Fragment1 fragment1 = new Fragment1();
        fragmentTransaction.replace(android.R.id.content, fragment1);
```

```
        fragmentTransaction.addToBackStack(null);
        fragmentTransaction.commit();
    }

}
```

Klasa fragmenta data je sledećim kodom:

```
package com.metropolitan.preferencefragmentdemo;

import android.os.Bundle;
import android.preference.PreferenceFragment;
/**
 * Created by Vladimir Milicevic on 27.10.2016..
 */

public class Fragment1 extends PreferenceFragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //---Učitava preference iz XML datoteke---
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

PRIMER 20 - KRAJ

Definisanje XML datoteke preferencija.

Sledećim kodom se završava primer. Datoteka preferences.xml, iz res/xml foldera poseduje sledeći kod.

```
<?xml version="1.0" encoding="utf-8"?>


        <CheckBoxPreference
            android:title="Checkbox"
            android:defaultValue="false"
            android:summary="True ili False"
            android:key="checkboxPref" />
    </PreferenceCategory>

        <EditTextPreference
            android:name="EditText"
            android:summary="Unesite string"
            android:defaultValue="[Unesite string ovde]"
            android:title="Promenite tekst"
            android:key="editTextPref" />
        <RingtonePreference
```

```
            android:name="Ringtone Preference"
            android:summary="Izaberite melodiju zvona"
            android:title="Melodija zvona"
            android:key="ringtonePref" />

            <EditTextPreference
                android:name="EditText"
                android:summary="Unesite string"
                android:title="Unesite string (drugi ekran)"
                android:key="secondEditTextPref" />
        </PreferenceScreen>
    </PreferenceCategory>
</PreferenceScreen>
```

ZADATAK 10 - KREIRAJTE VLASTITU APLIKACIJU PODSETNIK

Pokušajte sami!!!

Pokušajte da na osnovu demonstriranih primera sa kreirate vlastitu aplikaciju Podsetnik.

Poglavlje 11

Domaći zadatak 3 -2

ZADATAK 2– VEŽBANJE KREIRANJA UI POMOĆU OSNOVNIH POGLEDA, PICKER POGLEDA I SPECIJALIZOVANIH FRAGMENATA.

Naučeni elementi lekcije biće primenjeni u sledećem zadatku.

Kreirati Android aplikaciju po sledećim instrukcijama:

1. Na početnom ekranu koristiti PreferenceFragment;
2. Postaviti dve kategorije: Test i Univerzitet Info
3. Prva kategorija sadrži jedan ButtonGroup pogled na kojem se nalazi pitanje i tri ponuđena odgovora. Izbor tačnog, ili netačnog, odgovora propratiti odgovarajućim komentarom na ekranu;
4. Druga kategorija podrazumava pokretanje novog ekrana. Na novom ekranu postaviti kontrole za unos teksta pomoću kojih će aplikacija preuzeti informacije o nazivu i adresi našeg Univerziteta.
5. Na ovom ekranu obezbediti izbor datuma i vremena.

Zaključak

PREGLED LEKCIJE03

Korisnički interfejs je moguće kreirati na dva načina, XML datotekom ili JAVA klasom aktivnosti aplikacije.

U ovoj lekciji je naučeno kako je moguće kreirati korisnički interfejs u Android operativnom sistemu. Posebno je diskutovano o različitim rasporedima elemenata koje je moguće iskoristiti sa ciljem definisanja pogleda u konkretnom Android korisničkom interfejsu.

Takođe, istaknuto je da Android uređaji podržavaju različite orijentacije ekrana o kojima se mora voditi računa prilikom kreiranja aplikacija.

U nastavku, elaborirana je funkcija ActionBar koja je uvedena u upotrebu sa novijim verzijama Androida. Pokazano je kako funkcioniše i kako se dodaju nove stavke u ActionBar.

U nastavku je pokazano da korisnički interfejs može da se kreira i JAVA programskim kodom, u klasi aktivnosti, kada je neophodno dinamičko generisanje komponenata korisničkog interfejsa. Posebno je istaknuto da dinamičko generisanje elemenata korisničkog interfejsa u Android aplikacijama može da bude veoma težak zadatak. Iz navedenog razloga je eleborirano da se taj zadatak mora obavljati veoma pažljivo i samo u situacijama kada za tim postoji potreba.

PREGLED LEKCIJE03 - NASTAVAK

Obrađeni su osnovni i specijalizovani pogledi.

U lekciji su, takođe, obrađeni načini prikaza koji se sreću u brojnim Android aplikacijama i predstavljaju dobru osnovu za samostalno kreiranje korisničkog interfejsa Android aplikacija.

Obrađeni su osnovni pogledi koji su predstavljeni kontrolama korisničkog interfejsa pomoću kojih aplikacija komunicira sa korisnikom. Obrađeni su sledeći osnovni pogledi: ***TextView, EditText, Button, ImageButton, CheckBox, ToggleButton, RadioButton i RadioGroup*** i ***ProgressBar***.

U nastavku je bilo govora o ***Picker*** pogledima pomoću kojih korisnici mogu da podešavaju vreme i datum u Android aplikacijama. Obrađeni su pogledi: ***TimePicker*** i ***DatePicker***.

Konačno, obrađeni su ***List*** pogledi koji omogućavaju prikazivanje dugačkih listi. U Android operativnom sistemu podrška prikazivanju dugih listi realizovana je primenom pogleda ***ListView*** i ***SpinnerView***.
Na kraju, diskutovano je o primenama specijalizovanih fragmenata u Android aplikacijama. Svaka od navedenih tema praćena je odgovarajućim pokaznim primerom.

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