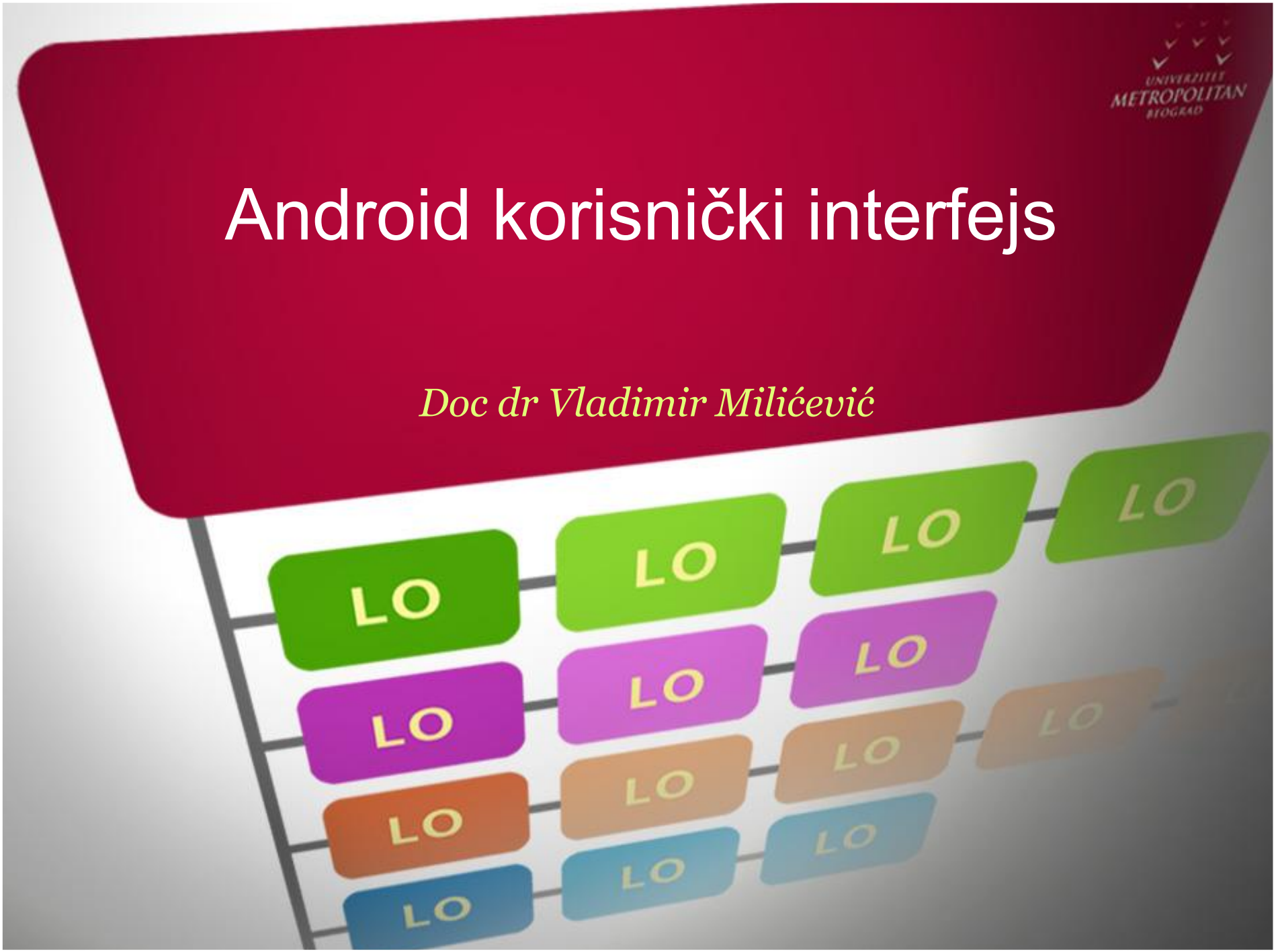
Android korisnički interfejs

Doc dr Vladimir Milićević



DATOTEKA 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:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    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=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>
```

Slika-1 Opšti oblik main.xml datoteke

Interfejs definisan ovom datotekom izvršava se pomoću metode *onCreate()* integrisane u odgovarajuću klasu aktivnosti. Nakon toga koristi se metoda setCont*entView()* klase aktivnosti. Tokom prevođenja Android aplikacije, svaki element main.xml datoteke prevodi se u odgovarajuću GUI klasu.

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

Slika-2 Implementacija metoda onCreate() i onContentView()

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:

- *LinearLayout;*
- *AbsoluteLayout;*
- *TableLayout;*
- *RelativeLayout;*
- *FrameLayout,*
- *List View*, i
- *Grid View*

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.

Atribut	Opis
layout_width	Širina pogleda ili grupe pogleda.
layout_height	Visina pogleda ili grupe pogleda.
layout_marginTop	Definiše dodatni prostor iznad pogleda ili grupe pogleda.
layout_MarginBottom	Definiše dodatni prostor ispod pogleda ili grupe pogleda.
layout_marginLaft	Definiše dodatni prostor levo od pogleda ili grupe pogleda.
layout_marginRight	Definiše dodatni prostor desno od pogleda ili grupe pogleda.
layout_gravity	Definiše kako se pozicioniraju izvedeni pogledi.
layout_weight	Definiše koliko dodatnog prostora u rasporedu treba alocirati za pogled.
layout_x	x koordinata pogleda ili grupe pogleda.
layout_y	y koordinata pogleda ili grupe pogleda.

Slika-4 Zajednički atributi različitih pogleda i grupa pogleda

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 Slikom 3. Na slici se vidi da je u main.xml datoteci grupa pogleda definisana xml tagom <LinearLayout> … </LinearLayout >, a element korisničkog interfejsa za prikazivanje teksta tagom <TextView… />.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    <!-- ili horizontal ukoliko zeliko orijentaciju po vrsti--> >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</LinearLayout>
```

Slika-3 xml kod za LinearLayout grupu pogleda

LINEARLAYOUT - NASTAVAK

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.

```
<?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/btnStartService"
      android:layout_width="270dp"
      android:layout_height="wrap_content"
      android:text="start_service"/>

   <Button android:id="@+id/btnPauseService"
      android:layout_width="270dp"
      android:layout_height="wrap_content"
      android:text="pause_service"/>

   <Button android:id="@+id/btnStopService"
      android:layout_width="270dp"
      android:layout_height="wrap_content"
      android:text="stop_service"/>

</LinearLayout>
```

Slika5- LinearLayout sa vertikalnim rasporedom

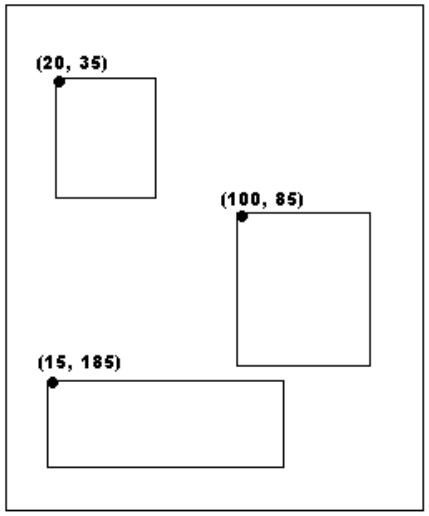
Zamenom navedene komande komandom android.orientation = „horizontal“ dobija se horizontalna organizacija komponenata korisničkog interfejsa.

Slika6- LinearLayout sa horizontalnim rasporedom

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.



Slika7- AbsoluteLayout

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:layout_width="100dp"
      android:layout_height="wrap_content"
      android:text="OK"
      android:layout_x="50px"
      android:layout_y="361px" />
   <Button
      android:layout_width="100dp"
      android:layout_height="wrap_content"
      android:text="Cancel"
      android:layout_x="225px"
      android:layout_y="361px" />

</AbsoluteLayout>
```

Slika7- AbsoluteLayout organizacija korisničkog interfejsa

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. U main.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. Sledećom slikom je pokazano kako se main.xml datotekom definiše *TableLayout* grupa pogleda.

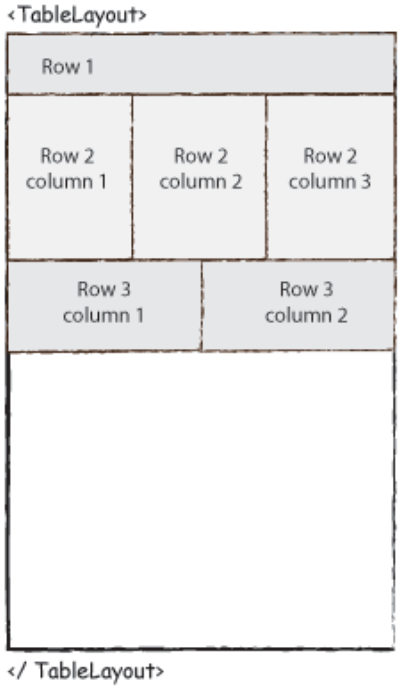
```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">

   <TableRow
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">
   <TextView
      android:text="Time"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_column="1" />
   <TextClock
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/textClock"
      android:layout_column="2" />
   </TableRow>

   <TableRow>
   <TextView
      android:text="First Name"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_column="1" />
   <EditText
      android:width="200px"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
   </TableRow>

   <TableRow>
   <TextView
      android:text="Last Name"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_column="1" />
   <EditText
      android:width="100px"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
   </TableRow>
      <TableRow
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">
   <RatingBar
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/ratingBar"
      android:layout_column="2" />
   </TableRow>
      <TableRow
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"/>
   <TableRow
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">
   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Submit"
      android:id="@+id/button"
      android:layout_column="2" />
   </TableRow>
```

Slika8- TableLayout organizacija korisničkog interfejsa - main.xml

TABLELAYOUT - NASTAVAK

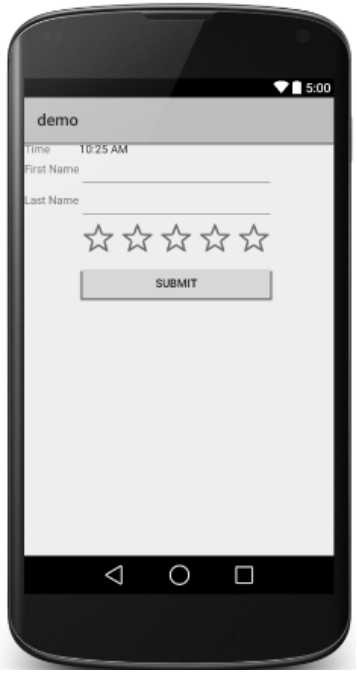
TableLayout kontejner ne prikazuje linije između vrsta i kolona koje čine grupu pogleda.

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



Slika9- TableLayout - grafički prikaz

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

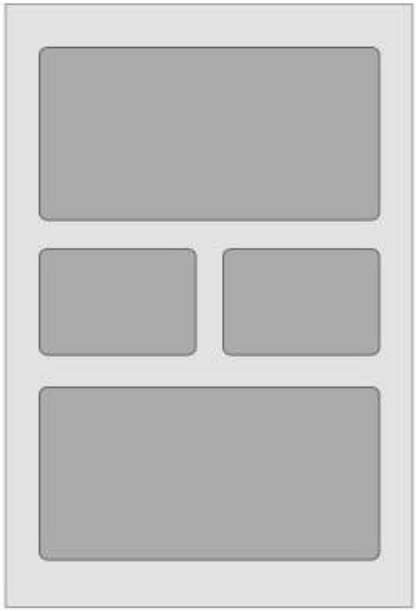


Slika10- TableLayout organizacija kontrola korisničkog interfejsa

RELATIVELAYOUT

RelativeLayout grupa pogleda pakuje komponente korisničkog interfejsa po relativnom rasporedu.

RelativeLayout 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.



Slika11- RealativeLayout grupa pogleda

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"
   android:paddingLeft="16dp"
   android:paddingRight="16dp" >

   <EditText
      android:id="@+id/name"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:hint="@string/reminder" />

   <LinearLayout
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_alignParentStart="true"
      android:layout_below="@+id/name">

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="New Button"
      android:id="@+id/button" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="New Button"
      android:id="@+id/button2" />

   </LinearLayout>

</RelativeLayout>
```

Slika12- RealativeLayout grupa pogleda - main.xml

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.

Slika13- FrameLayout grupa pogleda (izvor: tutorialspoint.com)

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">

   <ImageView
      android:src="@drawable/ic_launcher"
      android:scaleType="fitCenter"
      android:layout_height="250px"
      android:layout_width="250px"/>

   <TextView
      android:text="Frame Demo"
      android:textSize="30px"
      android:textStyle="bold"
      android:layout_height="fill_parent"
      android:layout_width="fill_parent"
      android:gravity="center"/>
</FrameLayout>
```

Slika14- FrameLayout grupa pogleda - main.xml

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.

klasa aktivnosti

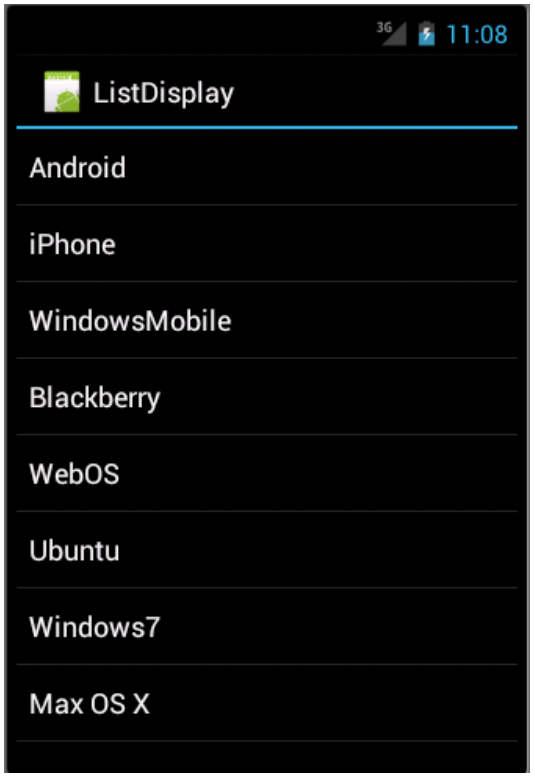
```
package com.example.ListDisplay;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class ListDisplay extends Activity {

   String[] mobileArray = {"Android","IPhone","WindowsMobile","Blackberry","WebOS","Ubunt

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

      ArrayAdapter adapter = new ArrayAdapter<String>(this, R.layout.activity_listview, m

      ListView listView = (ListView) findViewById(R.id.mobile_list);
      listView.setAdapter(adapter);
   }
}
```

main.xml

```
<LinearLayout 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:orientation="vertical"
   tools:context=".ListActivity" >

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

</LinearLayout>
```

Slika-15 ListView - primena Adaptera i main.xml datoteka

Program koji implementira navedene datoteku ima sledeću reprezentaciju.

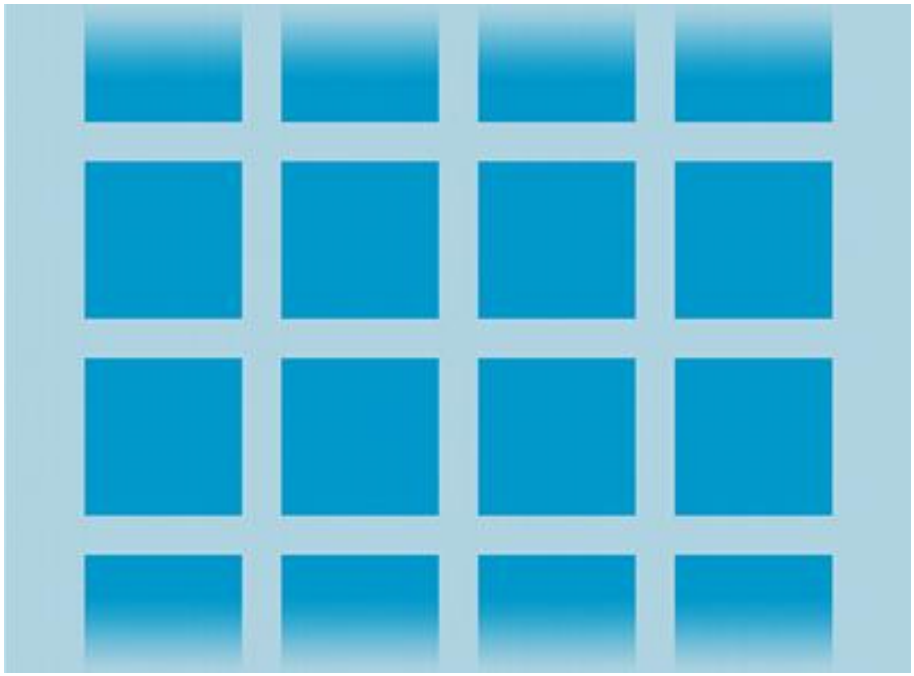


Slika-16 ListView raspored

GRIDVIEW

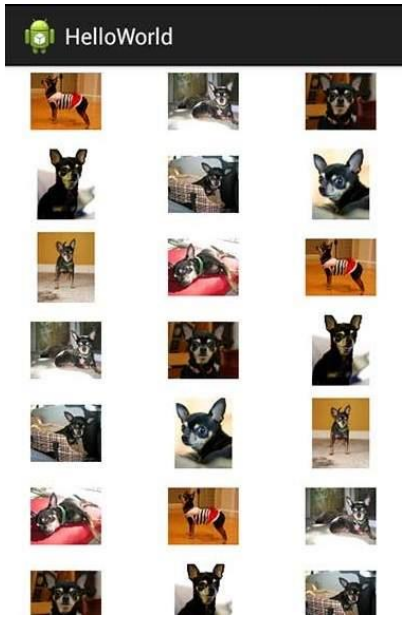
GridView prikazuje stavke korisničkog interfejsa u formi matrice.

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-18 GridView raspored

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



Slika-19 Primer primene GridView rasporeda

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.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.GridView;

public class MainActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

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

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }
}
```

Slika-19 GridView primer - glavna klasa aktivnosti

Biće kreirana nova xml datoteka strings.xml i modifikovana main.xml.

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/gridview"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:columnWidth="90dp"
   android:numColumns="auto_fit"
   android:verticalSpacing="10dp"
   android:horizontalSpacing="10dp"
   android:stretchMode="columnWidth"
   android:gravity="center"
/>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">HelloWorld</string>
   <string name="action_settings">Settings</string>
</resources>
```

Slika-20 xml datoteke primera

GRIDVIEW – IMAGEADAPTER

Za dodavanje slika u pogled neophodno je angažovati ImageAdapter.

Na kraju, neophodno je kreirati JAVA klasu kojom se uvodi ImageAdapter u program. Navedeno je realizovano sledećim kodom.

```
package com.example.helloworld;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;

public class ImageAdapter extends BaseAdapter {
   private Context mContext;

   // Constructor
   public ImageAdapter(Context c) {
      mContext = c;
   }

   public int getCount() {
      return mThumbIds.length;
   }

   public Object getItem(int position) {
      return null;
   }

   public long getItemId(int position) {
      return 0;
   }
```

```
   public View getView(int position, View convertView, ViewGroup parent)
      ImageView imageView; // kreira novi ImageView

      if (convertView == null) {
         imageView = new ImageView(mContext);
         imageView.setLayoutParams(new GridView.LayoutParams(85, 85));
         imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
         imageView.setPadding(8, 8, 8, 8);
      }
      else
      {
         imageView = (ImageView) convertView;
      }
      imageView.setImageResource(mThumbIds[position]);
      return imageView;
   }

   public Integer[] mThumbIds = {  //čuva slike u nizu
      R.drawable.sample_2, R.drawable.sample_3,
      R.drawable.sample_4, R.drawable.sample_5,
      R.drawable.sample_6, R.drawable.sample_7,
      R.drawable.sample_0, R.drawable.sample_1,
      R.drawable.sample_2, R.drawable.sample_3,
      R.drawable.sample_4, R.drawable.sample_5,
      R.drawable.sample_6, R.drawable.sample_7,
      R.drawable.sample_0, R.drawable.sample_1,
      R.drawable.sample_2, R.drawable.sample_3,
      R.drawable.sample_4, R.drawable.sample_5,
      R.drawable.sample_6, R.drawable.sample_7
   };
}
```

Slika-21 ImageAdapter

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.

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.

```
<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"
    />
```

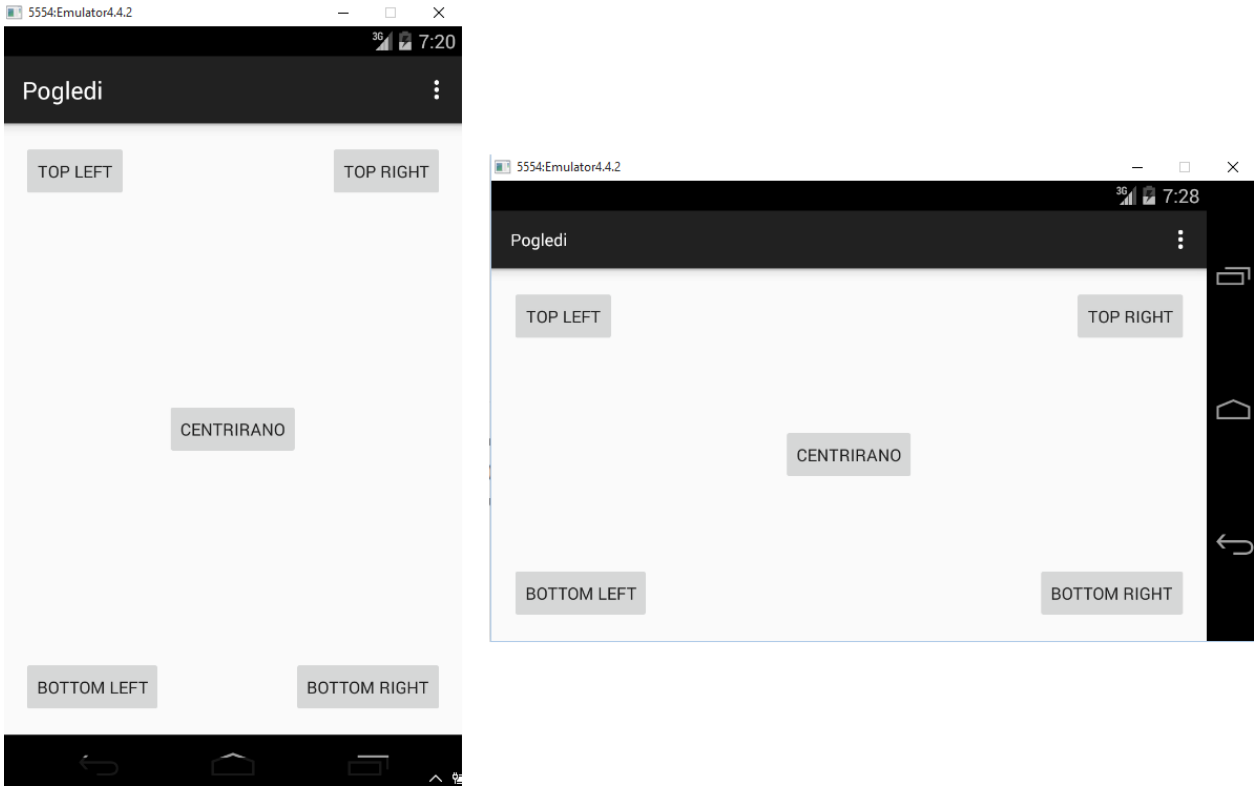
```
<Button
    android:id="@+id/button4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bottom Right"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true"
    />
<Button
    android:id="@+id/button5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Centrirano"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    />
```

Slika-22 Usidrenje

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 promena u orijentaciji emulatora i primena tehnike usidrenje.



Slika-1 Promena orijentacije ekrana i usidrenje

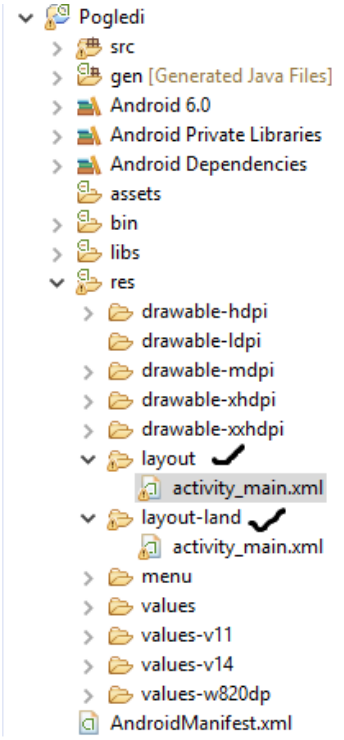
PROMENA VELIČINE I POLOŽAJA

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 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 main.xml datotekom.

Sledećom slikom je prikazana hijerarhija projekta sa navedenim folderima.



Slika-2 Hijerarhija projekta sa dva res/layout foldera

PROMENA VELIČINE I POLOŽAJA – PROMENA ORIJENTACIJE EKRANA

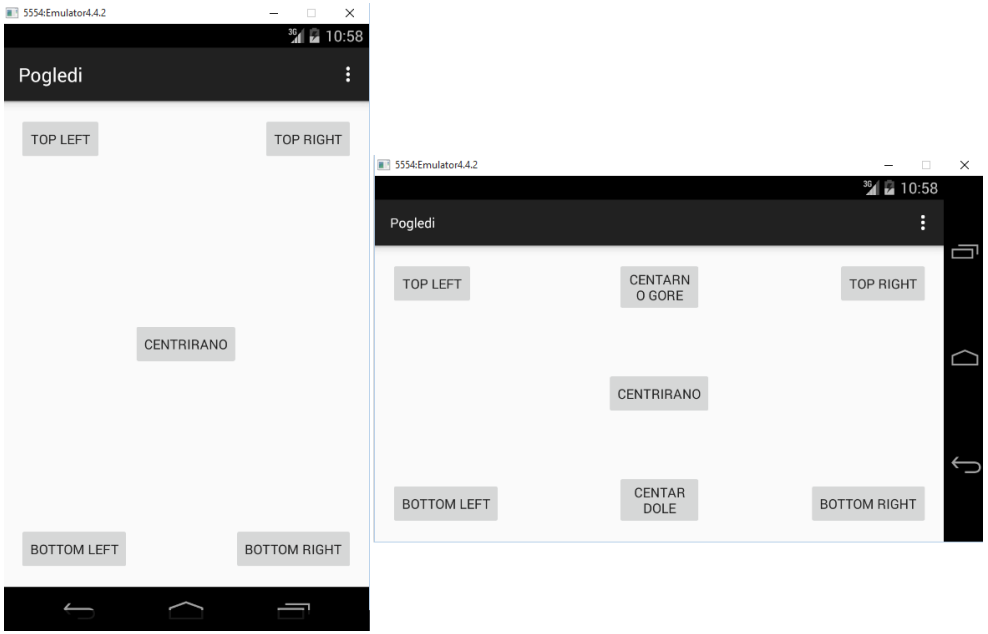
Za svaku orijentaciju Android automatski aktivira odgovarajući xml dokument.

U novom res/layout-land folderu kreiraćemo main.xml datoteku koja se od main.xml datoteke iz res/layout foldera razlikuje po dodatim linijama xml koda prikazanih sledećom slikom.

```
<Button
    android:id="@+id/button6"
    android:layout_width="180px"
    android:layout_height="wrap_content"
    android:text="Centarno 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"
    />
```

Slika-3 Dodatni kod za res/layout-land/main.xml

Učitavanje aktivnosti u vertikalnom režimu već je prikazano i uočeno je da je došlo do prikazivanja pet dugmića. Okretanjem telefona, ili pritiskom na ctrl – F11 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-4 Uporedni prikaz obe orijentacije 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. U tu svrhu biće uveden sledeći primer.

```
<?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>
```

Slika-1 main.xml datoteka primera

Glavna klasa aktivnosti data je sledećim programskim kodom.

```
package com.metropolitan.Orijentacije;

import android.app.Activity;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.util.Log;
import android.view.Display;
import android.view.WindowManager;

public class OrijentacijeActivity extends Activity {
   /** poziva se kada se aktivnost kreira. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.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---
      //---snimanje svega što želite da sačuvate; koristi Object type
         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();
      }
   }
```

Slika-2 OrijentacijeActivity.java

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. Klikom na ctrl+F11 menja se orijentacija ekrana i u LogCat prozoru se prikazuje sledeće.

```
StateInfo          onPause
StateInfo          onStop
StateInfo          onDestroy
Orientation        Landscape mode
StateInfo          onStart
StateInfo          onResume
```

Slika-3 Praćenje aktivnosti

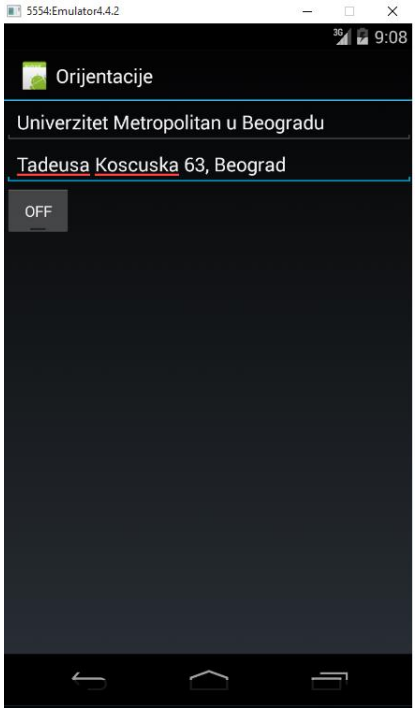
Moguće je primetiti da je promenom orijentacije došlo do potpunog gašenja inicijalne aktivnosti i pokretanja nove. Ovo je veoma važno da se zna u slučaju da postoji potreba za čuvanjem određenih podataka koji bi promenom orijentacije bili izgubljeni. U tom slučaju je neophodno sačuvati stanje aktivnosti 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.

POKRETANJE PRIMERA I PRAĆENJE PERZISTENTNOSTI PODATAKA

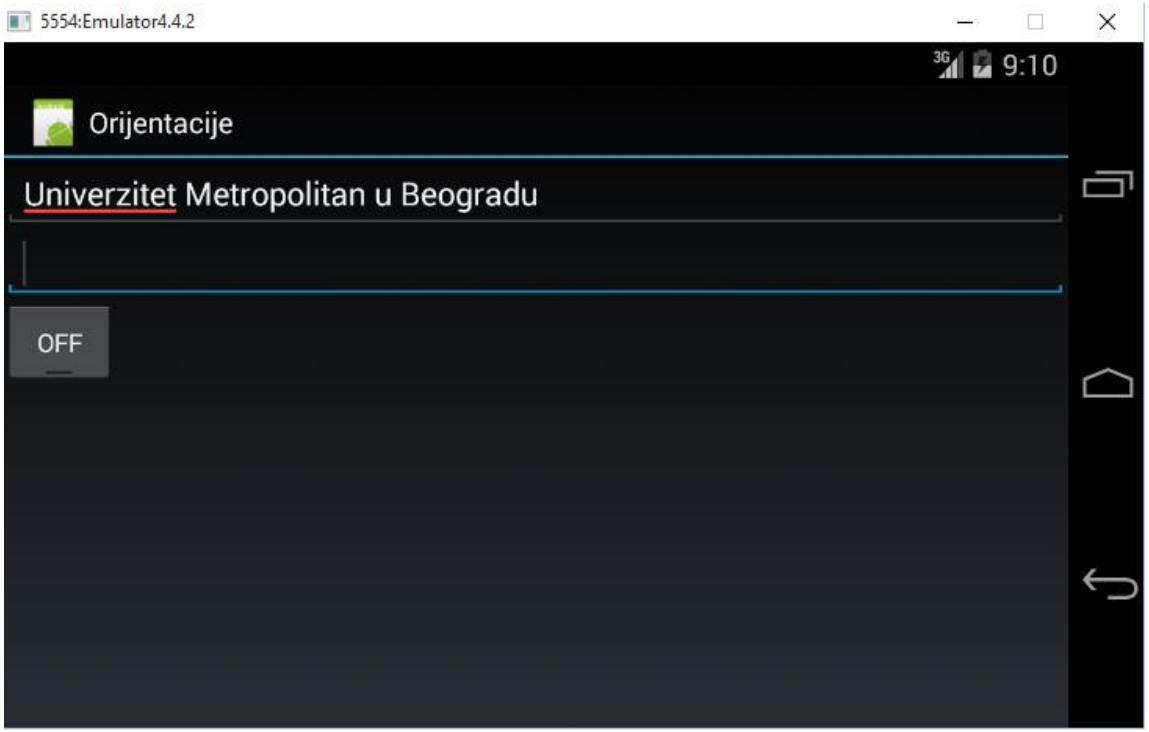
Pogled koji odgovara kontroli čiji je naziv definisan android:id atributom biće učitan sa promenom orijentacije uređaja.

Sledećom slikom dat je emulator sa vertikalnom orijentacijom ekrana i sa pokrenutim primerom.



Slika-4 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.



Slika-5 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.

```
@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");
}
```

Slika-6 Metode za čuvanje informacija

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.

```
import android.view.Display;
import android.view.WindowManager;

public class OrientationsActivity extends Activity {
    /** poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.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");
        }
    }
```

Slika-7 Detekcija promene orijentacije

KONTROLA ORIJENTACIJE

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

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 Activity. Navedeno je prikazano sledećim kodom.

```
import android.content.pm.ActivityInfo;

public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
//fiksiranje horizontalnog režima
setRequestOrientation (ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
}
```

Slika-8 Kontrola orijentacije

Fiksiranje u vertikalni raspored izvodi se instrukcijom ActivityInfo.SCREEN_ORIENTATION_PORTRAIT.

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

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.Orientations"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="Orijentacije" >
        <activity
            android:label="Orijentacije"
            android:name=".OrientationsActivity"
            android:screenOrientation="landscape" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

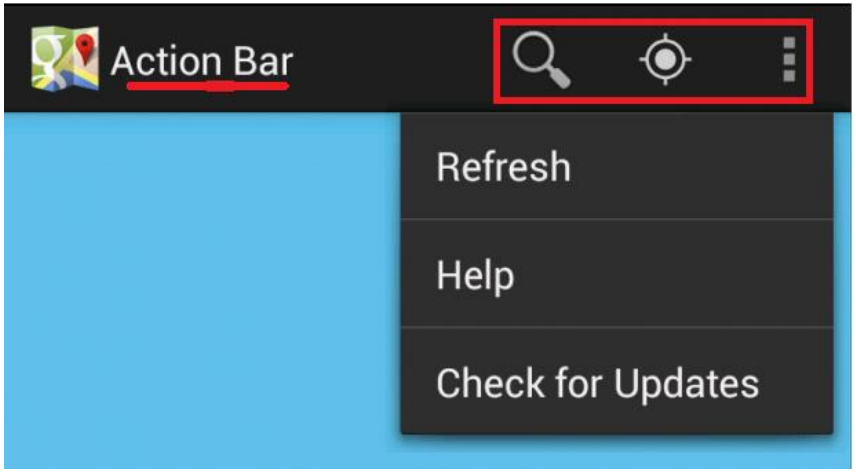
Slika-9 Kontrola orijentacije u AndroidManifest.xml

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.

Sledećom slikom prikazan je ActionBar sa dodatnim stavkama.



Slika-1 ActionBar sa dodatnim stavkama

PRIKAZIVANJE I SKRIVANJE ACTIONBAR FUNKCIJE

U Android aplikacijama ActionBar može biti softverski prikazan ili sakriven.

Upravljanje ActionBar funkcijom biće prikazano na sledećem primeru:

1. Kreirati nov Android projekat *MojActionBar*;
2. Dodavanje i skrivanje funkcije definisati datotekom AndroidManifest.xml;
3. Modifikovati klasu *MojActionBar.java.*
4. Izvršiti prevođenje i pokretanje programa za demonstraciju emulatorom.
5. Datoteka main.xml će nositi podatak o nazivu našeg univeriteta koji će biti prikazan na ekranu.

```
<?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>
```

Slika-2 main.xml za demostraciju ActionBar

Sledećim kodom prikazana je AndroidManifest.xml datoteka. Opciono, umetanjem sledeće instukcije u blok <activity> ActionBar može biti skiven:

Android:theme=„android:style/Theme.Holo.NoActionBar“

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.metropolitan.MojActionBar"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="13" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".MyActionBarActivity">
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Slika-3 AndroidManifest.xml za demostraciju ActionBar

GLAVNA JAVA DATOTEKA AKTIVNOSTI

Dodavanje stavki akcija dešava se u klasi aktivnosti.

Glavna klasa aktivnosti nosi podatke o načinu funkcionisanja glavne aktivnosti ali i o stavkama koje se dodaju u ActionBar komponentu. Za ovaj primer kod klase dat je na sledeći način. Takođe, ovde je moguće sakriti ActionBar dodavanjem instrukcije actionBar.hide();

```
package net.learn2develop.MyActionBar;

import android.app.ActionBar;

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

        ActionBar actionBar = getActionBar();
        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)
{   //kreiranje menija
    MenuItem mnu1 = menu.add(0, 0, 0, "Stavka 1");
    {
        mnu1.setIcon(R.drawable.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.drawable.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.drawable.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);
    }
}
```

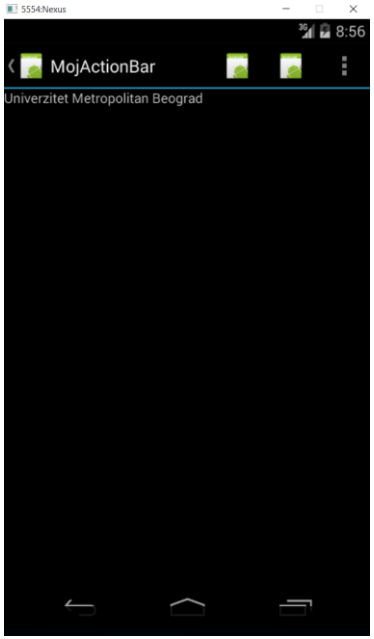
```
private boolean MenuChoice(MenuItem item)
{   //izbor stavki
    switch (item.getItemId()) {
    case  android.R.id.home:
        Toast.makeText(this,
            "Kliknuli ste na ikonu aplikacije!!!",
            Toast.LENGTH_LONG).show();

        Intent i = new Intent(this, MyActionBarActivity.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;
}
}
```

Slika-4 Glavna klasa aktivnosti

NAČIN FUNCIONISANJA ACTIONBAR KOMPONENTE

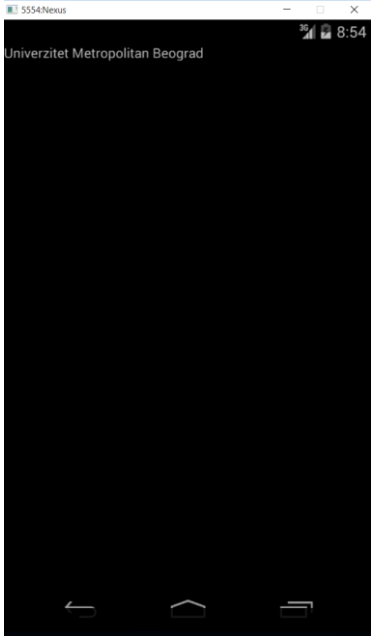
Atribut android:theme omogućava prikazivanje i uklanjanje komponente ActionBar.

Atribut android:theme omogućava prikazivanje i uklanjanje komponente ActionBar. Definisanjem njegove vrednosti na sledeći način: *android:theme/Theme.Holo.NoActionBar,* ActionBar neće biti prikazan.



Slika-5 ActionBar sa dopunskim stavkama

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.

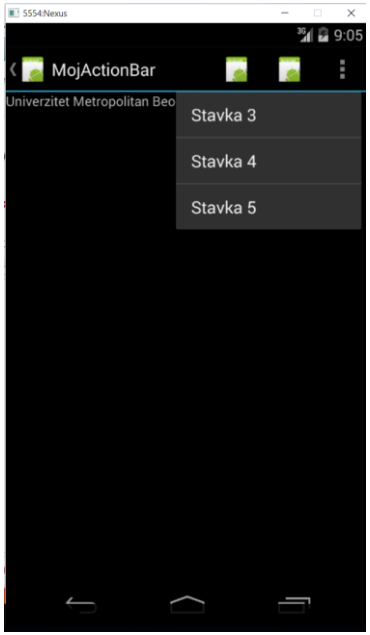


Slika-6 Implementacija metode hide()

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.



Slika-7 Stavke akcija

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()

KREIRANJE DINAMIČKOG UI

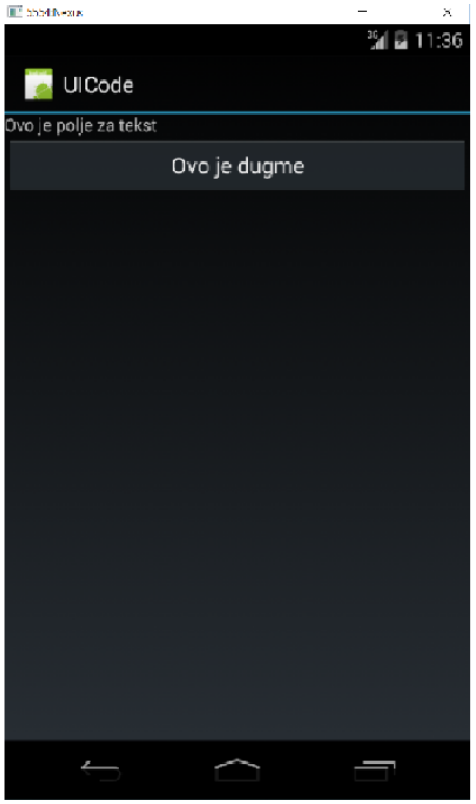
Dinamički korisnički interfejs kreira se primenom JAVA klasa.

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.

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

```
package com.metropolitan.UICode;
import android.app.Activity;
public class UICodeActivity extends Activity {
    /** 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);
    }
}
```



Slika-1 Klasa aktivnosti i dodavane UI komponenata

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 je *LinearLayout()* objekat koji sadrži sve poglede aktivnosti;
- Definisani su *TextView* i *Button* pogledi;
- Kreirani pogledi su dodati u *LinearLayout()* 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.

OBAVEŠTENJA UI – PREDEFINISANJE METODA AKTIVNOSTI

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

PREDEFINISANJE METODA AKTIVNOSTI

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

U sledećem primeru biće pokazano predefinisanje metoda iz super klase Activity. Prvo će biti prikazan UI definisan datotekom 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>
```

Slika-2 main.xml primera

Nakon uključivanja paketa za klasu aktivnosti aplikacije: **android.view.KeyEvent i android.widget.Toast,** moguće je predefinisati 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;
}
```

Slika-3 Predefinisanje onKeyDown() metode

NAČIN FUNKCIONISANJA PREDEFINISANE METODE

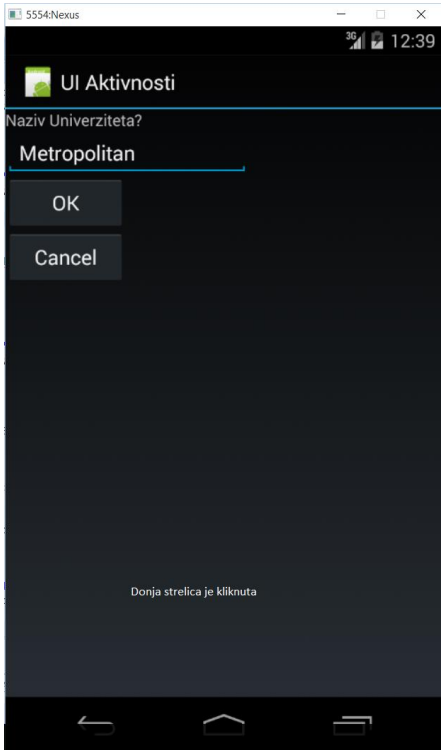
Pedefinisana metoda zamenjuje konkretnim akcijama praznu metodu iz Super klase.

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-4 Demonstracija primene predefinisanih metoda