VI – Korisnički interfejs

S A D R Ž A J

6.1 – Elementi 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***
- Stuktura ove datoteke prikazana je na sledećem slajdu
- 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.

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

6.1 – Elementi korisničkog interfejsa

```
<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>
```

6.1 – Elementi korisničkog interfejsa

- Postoje dva načina dizajniranja korisničkog interfejsa:
  1. **Proceduralno** – odnosi se na pisanje Java koda
  2. **Deklarativno** – odnosi se na pisanje XML koda
- Za kreiranje grafičkog korisničkog interfejsa uglavnom se koristi XML
- Kreiranjem interfejsa aktivnosti dobijaju svoju funkcionalnost, tj. vidljivost na ekranu uređaja koja ava interakciju sa korisnikom
- Osnovne jedinice korisničkog interfejsa su objekti:

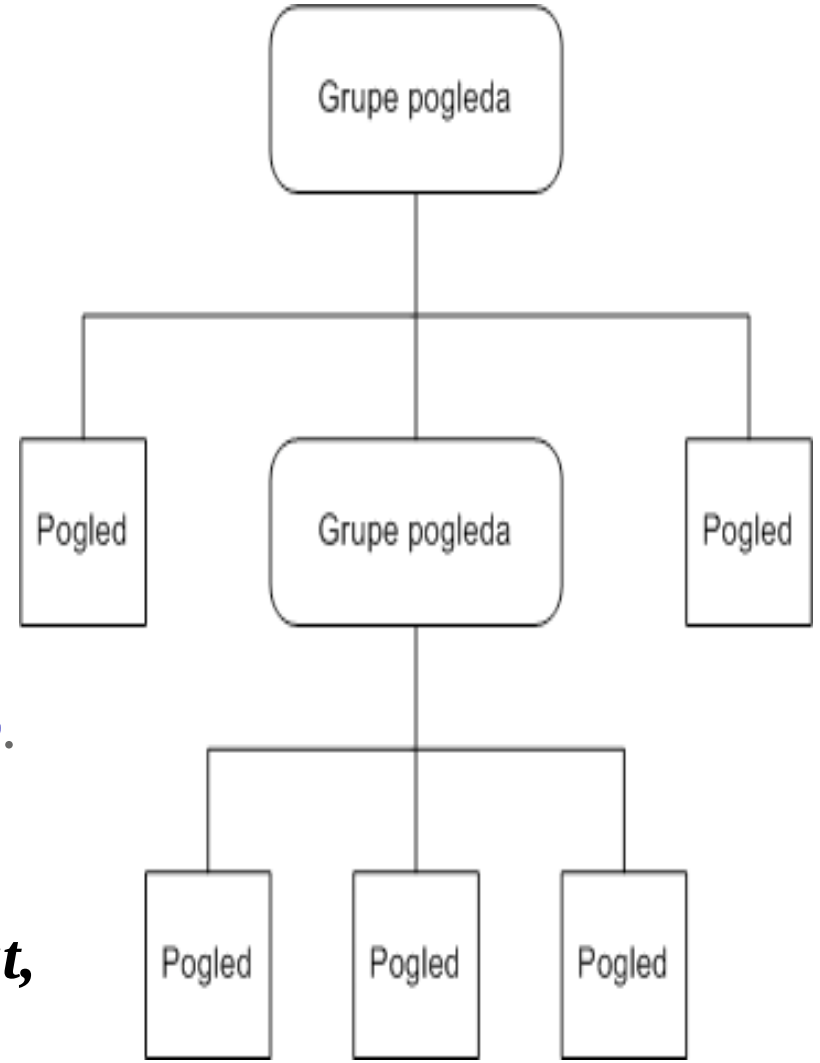
1. **pogled** (*View*)
   - ✓ Predstavlja objekat koji ima vlastitu reprezentaciju na ekranu
   - ✓ Njegova struktura u sebi nosi zapis izgleda i sadržaja određenog pravougaonog područja koje je vidljivo na ekranu,
   - ✓ Upravlja iscrtavanjem elemenata, pomeranjem sadržaja (*scrolling*) i ostalim faktorima koji utiču na izgled definisanog dela ekrana.
   - ✓ U hijerarhijskom stablu objekti **pogled** su listovi stabla.
   - ✓ Android raspolaže sa vec gotovim skupovima objekata ove vrste kao što su dugmad, labele, tekst polja, ***checkbox*** i slično.
   - ✓ Pogledi su definisani u baznoj Android klasi ***android.view.View***.

6.1 – Elementi korisničkog interfejsa

2. grupe pogleda (*ViewGroup*)

- ✓ posebna vrsta objekta pogled koja sadrži i upravlja grupom zavisnih objekata pogleda i grupe pogleda
- ✓ oni avaju kompleksnost prikaza korisničkog interfejsa.
- ✓ obezbeđuju način raspoređivanja komponenata korisničkog interfejsa na ekranu.
- ✓ Sve grupe pogleda izvedene su iz bazne klase ***android.view.ViewGroup***.
- ✓ U Android aplikacijama moguće je sresti sledeće grupe pogleda:

1. LinearLayout;
2. AbsoluteLayout;
3. TableLayout;
4. RelativeLayout;
5. FrameLayout,
6. List View
7. Grid View



6.1 – Elementi korisničkog interfejsa

- Iscrtavanje elemenata stabla započinje od korena stabla tako što aktivnost prvo poziva svoju **setContentView()** metodu i Android OS predaje referencu na koreni objekat.
- Svaki podčvor iscrtava se sam pozivanjem metode **draw()** i to pod uslovom da čvor sam određuje svoju lokaciju i veličinu.
- Roditeljski čvor (grupa pogleda) donosi konačnu odluku o veličini prostora za iscrtavanje podčvora i njegovom položaju na ekranu.
- Svaka grupa pogleda je odgovorna za podatke koji učestvuju u stvaranju (renderovanju) prikaza svojih podčvorova na ekranu.
- Podaci mogu sadržavati podatke o geometriji, tački gledišta, tekstu ili podatke o osvetljenju.
- Svaki pogled/grupa pogleda imaju određeni skup zajedničkih atributa.
- **Zajednički atributi** koji se najčešće koriste u pogledima ili grupama pogleda dati su sledećom tabelom:

6.1 – Elementi korisničkog interfejsae

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.

6.2 – XML layout

- Naša prva Android aplikacija koristila je najjednostavniji pogled ***TextView*** koji služi za prikaz teksta i bila je dizajnirana proceduralno.
- To znači da nije korišćen **XML kod** dizajniranja korisničkog interfejsa
- Sa obzirom da je poželjno koristiti deklarativan dizajn prilikom kreiranja složenijeg korisničkog interfejsa, u daljem tekstu biće opisan postupak kreiranja ***Hello world*** aplikacije korišćenjem **XML koda**
- Jedna od najvećih prednosti deklarativnog dizajna jeste potpuna odvojenost dizajna korisničkog interfejsa od aplikacione logike.
- To znači da male promene u kodu neće uticati na dizajn aplikacije
- Deklarisanje **TextView** pogleda pomoću **XML**-a obavlja se ovako:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="@string/zdravo"/>
```

6.2 – XML layout

- Ovo predstavlja XML layout fajl koji opisuje dizajn korisnič.interfejsa.
- U strukturi ovog fajla svaki čvor u stvari predstavlja pogled koji se koristi u aplikaciji (naziv **View** klase).
- U ovom primeru naveden je samo jedan **View** element i to **TextView**.
- U okviru XML layout fajla moguće je koristiti bilo koju klasu koja je nasleđena iz klase **View**, što podrazumeva i korisničke izvedene klase.
- Ovde je korišćen samo jedan element **TextView** sa svojih pet atributa:

1. **xmlns:android** - naziv XML imenskog prostora. Vrednost ovog atributa je *android*, što znači da će se koristiti atributi deklarisani u okviru Android imenskog prostora. Čvorovi najvišeg hijerarhijskog položaja u Android layout fajlu moraju da poseduju ovaj atribut.
2. **android:id** - dodeljuje jedinstveni identifikator elementu **TextView**. Dodeljeni ID je moguće koristiti za referenciranje ovog pogleda u izvornom kodu ili drugim XML deklaracijama.
3. **android:layout_width** – određuje širinu ekrana koju će ovaj pogled zauzeti. Kako je ovo jedini pogled koji se koristi, to je moguće zauzeti čitav ekran, što označava vrednost atributa "***fill_parent***".

6.2 – XML layout

3. **android:layout_height** - sličan atributu ***android:layout_width***. Razlika je u tome što se atribut ***android:layout_height*** odnosi na dostupnu visinu ekrana.
4. **android:text** - sadrži željeni tekst koji **TextView** treba da prikaže. Ovde se koristi resurs tipa **string** (**@string/zdravo** umesto direktno unete vrednosti stringa). Ovaj string je definisan u fajlu ***res/values/strings.xml***. Prilikom ubacivanja stringova u aplikaciju preporučuje se korišćenje resursa, a ne direktni unos vrednosti stringova. To je dobra praksa jer omogućava lokalizaciju aplikacije, odnosno mogućnost višejezičnosti, bez imena koda **layout** fajla.

- XML layout fajlovi nalaze se u folderu **/res/layout/** u okviru projekta.
- Podrazumevani layout fajl, **main.xml** se automatski formira
- U primeru ovaj fajl je ignorisan, a **layout** je kreiran direktno u kodu
- Da bi se on koristio, neophodno je načiniti odgovarajuće izmene.
- Najpre je potrebno otvoriti fajl **/res/layout/main.xml**, u njega ubaciti kod prikazan na prethodnom slajdu i sačuvati izmene.

6.2 – XML layout

- Zatim treba otvoriti fajl **/res/values/strings.xml,** gde se nalaze vrednosti stringova koji se koriste u aplikaciji, i ubaciti sledeći kod:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="zdravo">Hello world!</string>
<string name="app_name">Hello world application</string>
</resources>
```

- String **zdravo** se prikazuje u okviru **TextView**-a, a string **app_name** se odnosi na ime aplikacije koja će se prikazivati na Android uređaju
- Sada je potrebno promeniti sadržaj klase **Aktivnost** kako bi ona znala da treba da koristi kreirani XML layout.

```
package net.myelab.android.mojaaplikacija;
import android.app.Activity;
import android.os.Bundle;
public class Aktivnost extends Activity {
        @Override
         public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
         }
 }
```

6.2 - XML layout

- Odgovarajućim XML layout-ima se pristupa pozivanjem atributa **R.layout.naziv_layouta**, što predstavlja reprezentaciju kompajlovanog objekta odgovarajućeg XML layout-a.
- Ovi objekti se čuvaju u okviru fajla **/gen/paket/R.java**, koji ne treba ručno modifikovati.
- Izmenjenu aplikaciju je sada moguće pokrenuti na standardni način.
- U daljem tekstu biće prikazano više grupa pogleda (**layouta**) i pogleda (**views**) koji se mogu koristiti prilikom dizajniranja korisničkog interfejsa.
- Uz svaki pogled priložen je izvorni kod odgovarajućeg primera.

6.3.1 – Linearni Layout

- **LinearLayout** predstavlja grupu pogleda koja prikazuje više pogleda, horizontalno/vertikalno tj. organizuje poglede u jednu kolonu ili vrstu
- Ukoliko se koristi ugnježdavanje više **LinearLayout**-a, bolje je umesto njega koristiti **RelativeLayout**.
- Opšti oblik ove grupe pogleda dat je sledećim kodom:

```
<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>
```

- 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… />**.

6.3.1 – Linearni Layout

- 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.

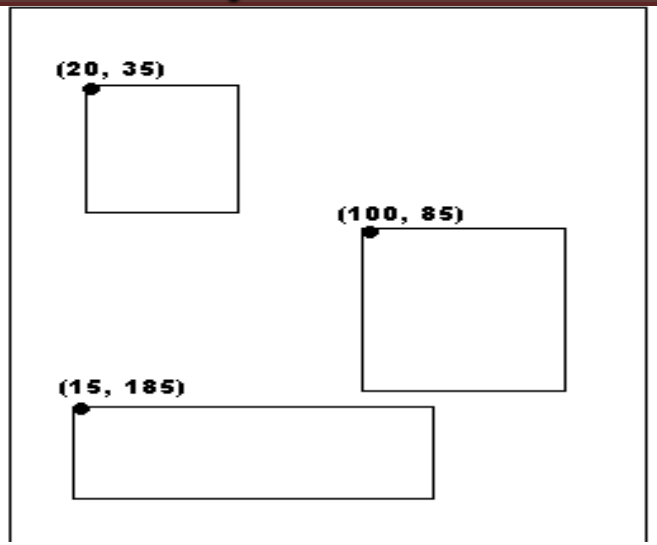
android.orientation = „vertical“

```
<?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>
```

android.orientation = „horizontal“

6.3.2 – Apsolutni Layout

- **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.
- Definisanjem **main.xlm** datoteke prikazanoj na slici 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>
```

6.3.3 – Relativni Layout

- 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.
- Pozicije su relativne u odnosu na element istog nivoa hijerarhije ili u odnosu na celo područje **RelativeLayout**-a
- Kao i svaka grupa pogleda, **RelativeLayout** se definiše **main.xml** datotekom kao što je to prikazano na slici.

```
<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>
```

6.3.4 – Tabelarni Layout

- **TableLayout** je grupa pogleda koja prikazuje svoje **View** elemente u redovima i kolonama
- Daje tabelarni prikaz
- 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>
```

6.3.4 – Tabelarni Layout

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

- Sledećom slikom grafički je prikazan jedan od mogućih načina organizacije kontrole korisničkog interfejsa primenom **TableLayout**-a

- Predhodni program koji implementira navedenu **main.xml** datoteku sa **TableLayout** organizacijom kontrola, na mobilnom uređaju imao bi sledeću reprezentaciju.

6.3.4 – Tabelarni Layout

- Primećujemo da način formiranja tabele u ovom layout-u podseća na formiranje tabele u HTML-u.
- Element **TableLayout** analogan je tagu **<table>** u HTML-u, a element **TableRow** analogan je elementu **<tr>**.
- Kao ćelije tabele mogu se koristiti bilo koji View elementi.
- U ovom slučaju se koristi **TextView**.
- Na kraju, ostaje kod početne aktivnosti, koji je isti kao u prethodna dva primera.
- Ovde je samo prikazana **onCreate** metoda ove aktivnosti:

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

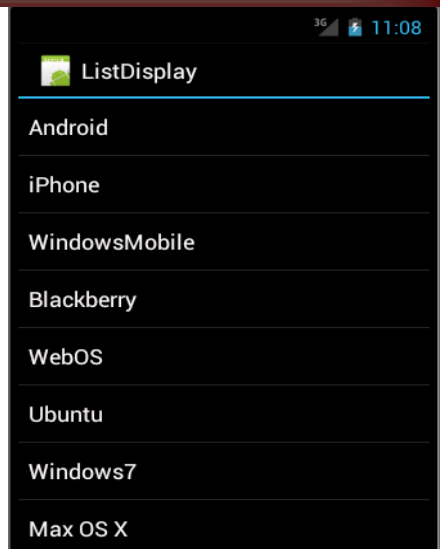
6.3.5 - Frame Layout

- **FrameLayout** se koristi sa ciljem izolovanja dela ekrana kako bi se prikazale pojedinačne stavke korisničkog interfejsa.
- Ovaj okvir trebalo bi da se koristi u svrhu prikazivanja pojedinačnih kontrola kako bi se prevazišli problemi različitih dimenzija ekrana.
- Primenom atributa **android:layout_gravity** moguće je dodati više stavki na ekran primenom ove grupe pogleda.
- **FrameLayout** organizacija korisničkog interfejsa 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>
```

6.3.6 – List View

- **ListView** pogled 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.
- Program koji implementira navedenu datoteku ima sledeću reprezentaciju:



klasa aktivnosti

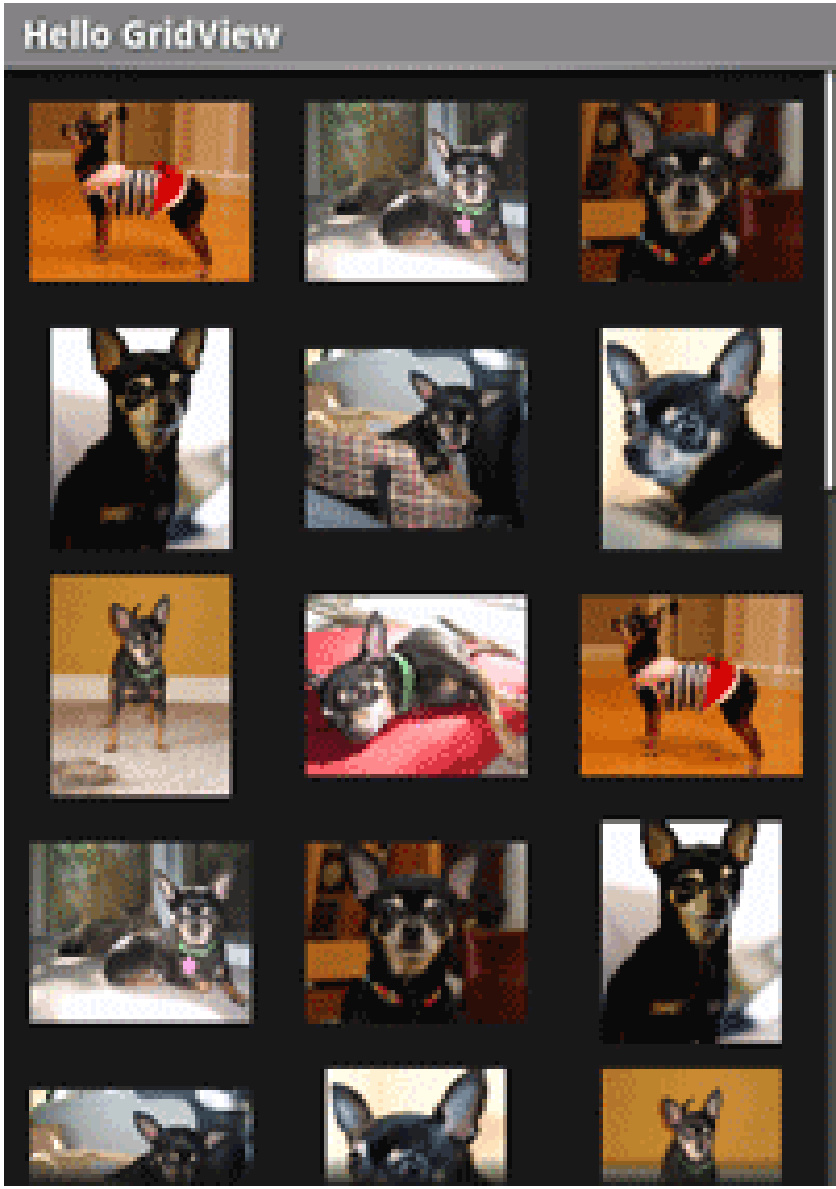
```
package com.example.ListDisplay;

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>
```

6.3.7 – GridView Layout

- **GridView** predstavlja grupu pogleda koja prikazuje elemente u dvodimenzionalnoj mreži (matrici) sa mogućnošću skrolovanja.
- Kada se izabere neka od njih, **toast** poruka će prikazati poziciju slike.
- Stavke korisničkog interfejsa automatski su raspoređene u grupu pogleda primenom adaptera pod nazivom **ListAdapter**.
- **GridView** grupa pogleda daje organizaciju korisničkog interfejsa kao što je prikazano sledećom slikom.
- Ovde je prikazano je kreiranje mreže sa malim slikama (**thumbnails**).



6.3.7 – GridView klase i XML podešavanja

- **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.

Modifikovana klasa aktivnosti

```
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;
   }
}
```

Modifikovana **main.xml**
Kreirana xml datoteka **strings.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>
```

6.3.7 – GridView Image adapter

- Za dodavanje slika u pogled neophodno je angažovati **ImageAdapter**.
- Na kraju, neophodno je kreirati JAVA klasu kojom se uvodi **ImageAdapter** u program.

```
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
   };
}
```

6.4 – Orijentacija ekrana

- Android OS podržava dve orijentacije ekrana: **portrait** i **landscape**.
- Prilikom promene orijentacije ekrana Android uređaja, ponovo se učitava trenutna aktivnost koja se povezuje sa sadržajem koji je prikazan
- Svaki put kada se orijentacija ekrana promeni, ponovo se poziva metoda ***onCreate()*** za iniciranje aktivnosti.
- Prilikom promene orijentacije ekrana tekuća aktivnost će biti uklonjena, a na njenom mestu biće kreirana nova aktivnost, tj. 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:

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

6.4 – Anchoring (Usidrenje)

- **RelativeLayout** prikaz jednostavno vrši usidravanje elemenata ekrana.
- Primena sledećih atributa vezuje interfejs elemente za 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 primeru **main.xml** datoteke sa **RelativeLayout** rasporedom za pet pogleda koji odgovaraju kontrolama ***dugmići (Button)***.

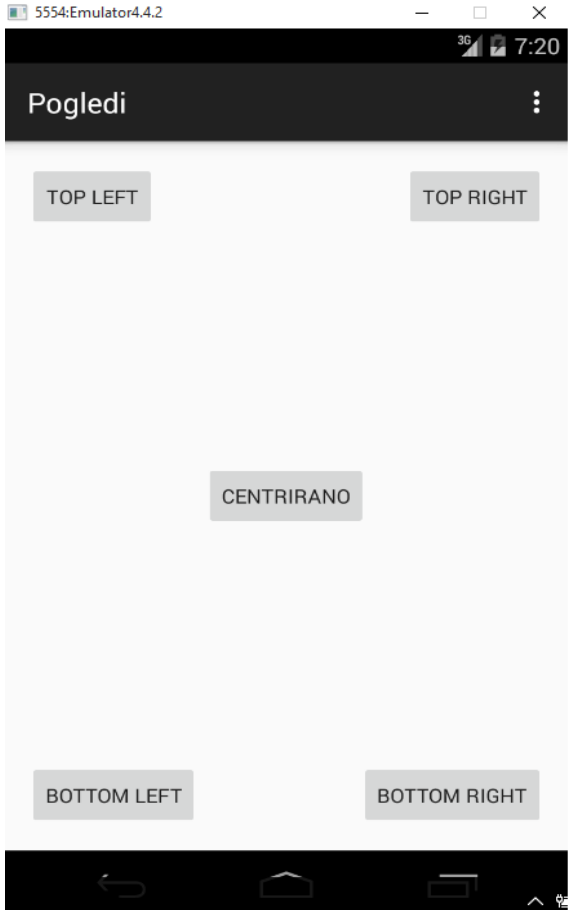
6.4 - Anchoring (Usidrenje)

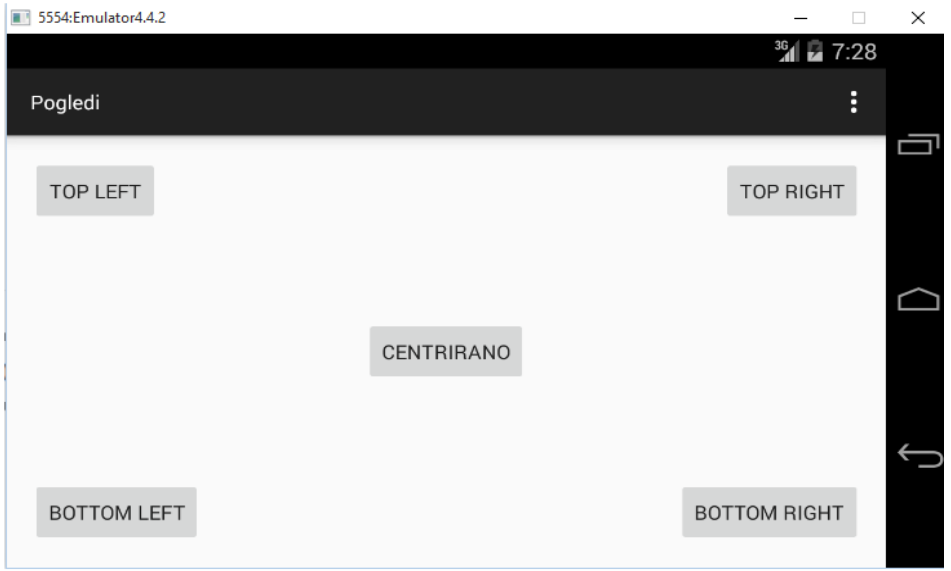
```
<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bottom Left"
    android:layout_alignParentLeft="true"
    android:layout_alignParentBottom="true"
    />
<Button
    android:id="@+id/button4"
    android:layout_width="wrap_content"
    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"
    />
```

6.4-Usidrenje-promena orijentacije AVD

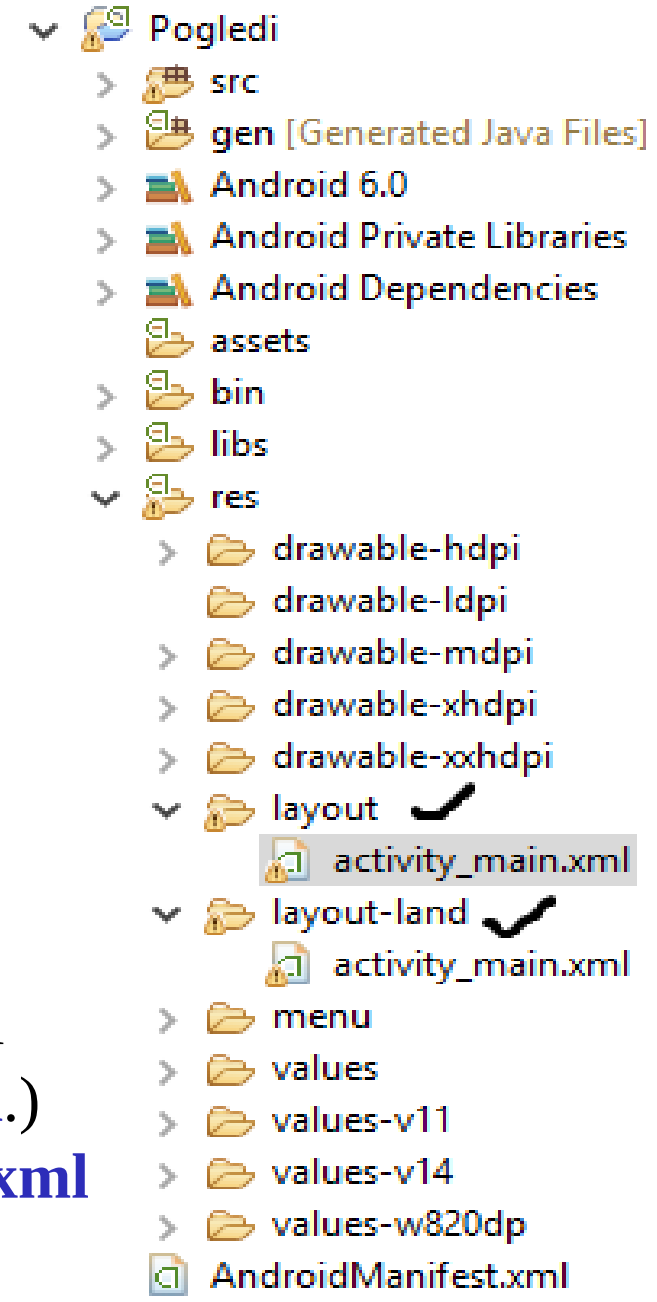
- Usidrenjem kreirane kontrole se pričvršćuju za ivice i središte ekrana.
- Upotrebom navedenih atributa,odgovarajuće kontrole biće raspoređene po ivicama ekrana ili centrirane, prilikom promene orijentacije ekrana
- Sledećom slikom je prikazana promena u orijentaciji emulatora i primena tehnike usidrenje.





6.4 – Promena veličine i položaja

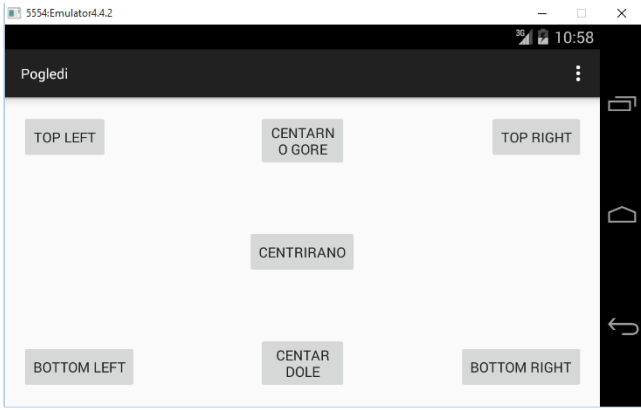
- Upravljanje promenama ekrana moguće je kreiranjem posebnih **xml** datoteka za svaku pojedinačnu orijentaciju.
- Promena veličine i položaja predstavlja napredniju tehniku upravljanja promenama u orijentaciji ekrana.
- Tehnika podrazumeva kreiranje posebnog **res/layout** foldera koji sadrži **xml** datoteke pojedinačnih orijentacija ekrana.
- Neka podrazumevani **res/layout** folder sadrži **main.xml** datoteku koja odgovara vertikalnoj orijentciji ekrana.
- Ako želimo da obezbedimo mogućnosti rada uređaja u horizontalnom režimu, treba kreirati novi folder (u našem primeru **res/layout-land**.)
- Takođe, ovaj folder će imati sopstvenu **main.xml** datoteku.



6.4 – Promena veličine i položaja

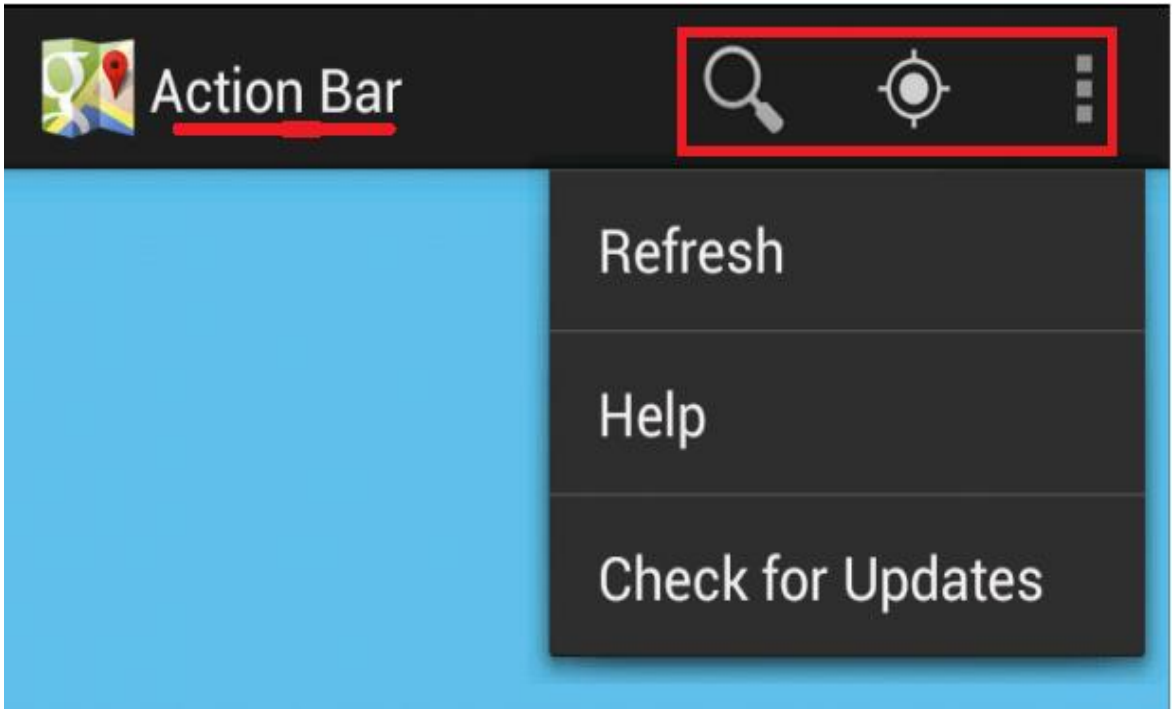
- Za svaku orijentaciju automatski se aktivira pripadajući xml dokument
- U novom **res/layout-land** folderu kreiraćemo **main.xml** datoteku koja se od **main.xml** datoteke iz **res/layout** foldera razlikuje po dodatim linijama xml koda prikazanih sledećom slikom.
- Učitavanje aktivnosti u vertikalnom režimu prikazuje pet dugmića.
- Okretanjem telefona, ili pritiskom na **CTRL-F11** za emulator, dolazi do promene u orijentaciji ekrana u horizontalni raspored.
- Sada će biti učitan iz novog **xml** fajla i biće prikazano sedam dugmića

```
<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"
    />
```



6.5 – Action Bar funkcija

- 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
- **ActionBar** prikazuje ikonu aplikacije i naziv aktivntosti.
- Sa desne strane,opciono, moguće je naći i određene **ActionBar** stavke.
- U Android aplikacijama **ActionBar** može biti prikazan ili sakriven



6.5 – Action Bar funkcija

- Upravljanje ActionBar funkcijom biće prikazano na sledećem primeru:

1. Kreirajte nov Android projekat *MojActionBar*;
2. Dodavanje i skrivanje f-je definisati datotekom AndroidManifest.xml
3. Modifikovati klasu ***MojActionBar.java***.
4. Izvršiti prevođenje i pokretanje programa putem emulatora.
5. Datoteka main.xml će nositi podatak 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>
```

6.5 - Action Bar funkcija

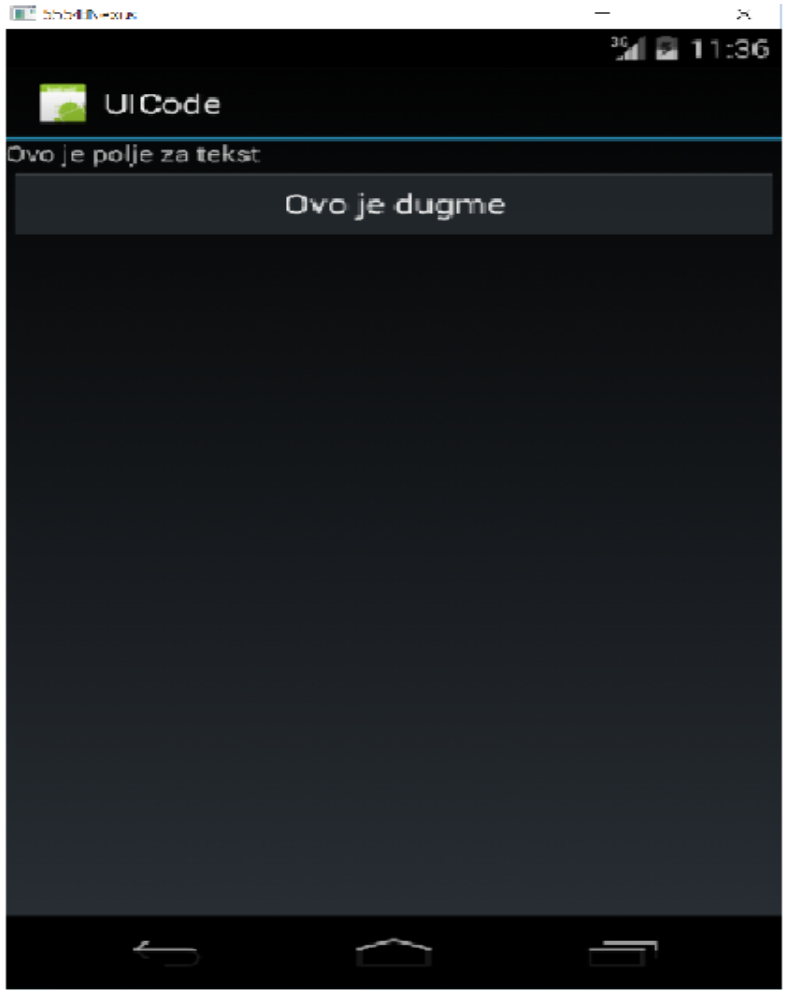
- Sledećim kodom prikazana je **AndroidManifest.xml** datoteka.
- Umetanjem sledeće instrukcije u blok **<activity>** **ActionBar** može biti skiven: **Android:theme=„android:style/Theme.Holo.NoActionBar“**

```
<?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>
```

6.6 – Kreiranje dinamičkog U/I

- Korisnički interfejs se može kreirati JAVA programskim kodom.
- Ovaj pristup je značajan kada dolazi do dinamičke promene U/I
- Od posebnog značaja, za ovakav način kreiranja korisničkog interfejsa, jeste kvalitetno kodiranja klasa aktivnosti.

```
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);
    }
}
```



6.6 – Kreiranje dinamičkog U/I

- 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:
  1. kreiran je ***LayoutParam()*** objekat čiji je zadatak omogućavanje parametara izgleda koje će koristiti drugi pogledi koji će naknadno biti kreirani;
  2. Kreiran je ***LinearLayout()*** objekat koji sadrži sve poglede aktivnosti;
  3. Definisani su ***TextView*** i ***Button*** pogledi;
  4. Kreirani pogledi su dodati u ***LinearLayout()*** objekat;
  5. Kreiran je ***LayoutParam()*** objekat kojeg koristi ***LinearLayout()***;
  6. Na kraju, ***LinearLayout()*** objekat je uključen u aktivnost na sledeći način: **this.addContentView(layout, layoutParam);**
- Korišćenje JAVA koda za kreiranje korisničkog interfejsa težak posao.
- UI se dinamički generiše samo u situacijama kada je to neophodno

6.6 – Komuniciranje sa korisnikom

- Na nivou aktivnosti **Activity** klasa sadrži metode koje je moguće predefinisati za konkretne potrebe komunikacije sa korisnikom.
- Korisnici komuniciraju sa Android aplikacijom korišćenjem UI i to:
  1. Na nivou **aktivnosti**;
  2. Na nivu **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:
  1. ***onKeyDown()*** – Izvršava se kada je pritisnut taster i ne rukuje se ni sa jednim pogledom konkretne aktivnosti;
  2. ***onKeyUp()*** – Izvršava se kada se pusti pritisnuti taster i ne rukuje se ni sa jednim pogledom konkretne aktivnosti;
  3. ***onMenuItemSelected()*** – Izvršava se kada korisnik selektuje stavku menija;
  4. ***onMenuOpened()*** – Izvršava se kada korisnik otvori meni.

6.6 - Komuniciranje sa korisnikom

- 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>
```

6.6 – Komuniciranje sa korisnikom

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

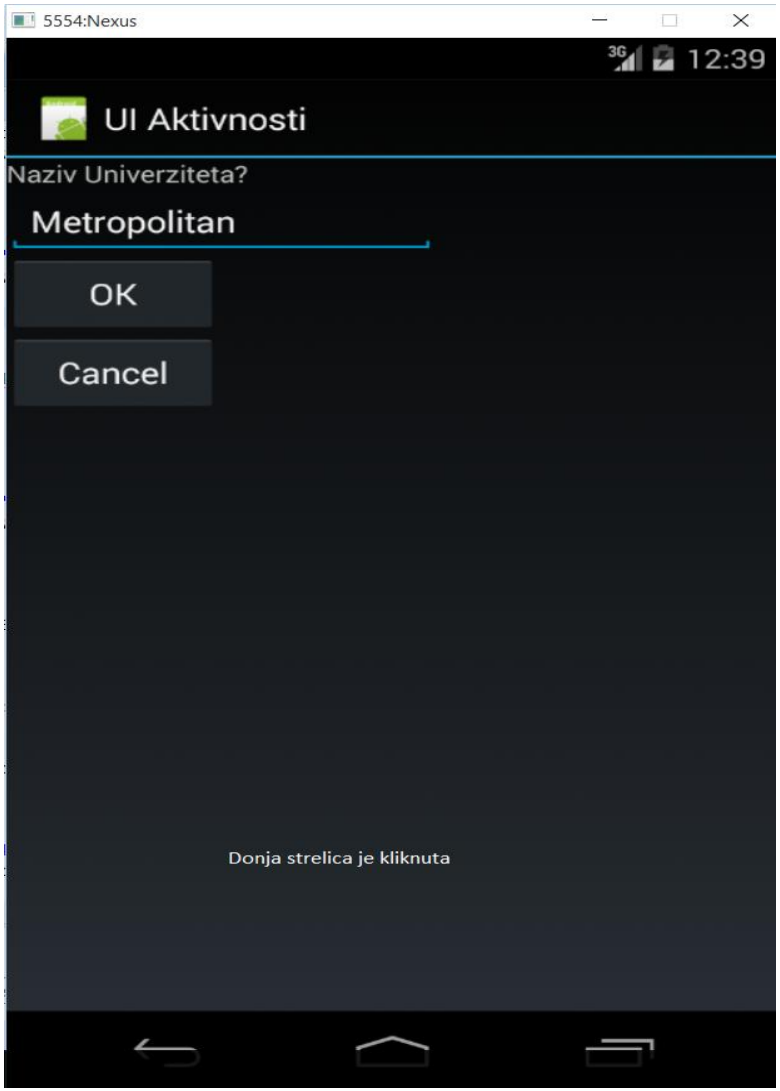
```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event)
{
    switch (keyCode)
    {
        case KeyEvent.KEYCODE_DPAD_CENTER:
            Toast.makeText(getBaseContext(),
                    "Centar je kliknut",
                    Toast.LENGTH_LONG).show();
            break;
        case 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;
}
```

6.6 – Komuniciranje sa korisnikom

- Pokretanjem programa inicira se glavna aktivnost i tada će pokazivač ukazivati na **EditText** polje na koje je postavljen fokus.
- U Androidu, fokusirana kontrola 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.
- U tom slučaju, izvršava se 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 sistemu vrednosti ***true*** ili ***false***.
- Vraća **true** ako je završena obrada događaja i sistem ne mora da nastavi

6.6 – Komuniciranje sa korisnikom

Sledećom slikom prikazan je deo u izvršenju posmatranog primera.



Hvala na pažnji !!!

Pitanja

? ? ?