IV – Struktura Android aplikacija

S A D R Ž A J

4.1 – Aktivnosti i klasa Activity

- Aktivnost (***Activity***) predstavlja komponentu aplikacije koja se uglavnom može poistovetiti sa jednim konkretnim prozorom aplikacije u kojem je korisnik u nosti da izvrši određenu radnju.
  Primer: *aktivnost služi da bi korisniku mobilnog telefona omogućila pozivanje određenog broja, slikanje fotografije korišćenjem ugrađene kamere, slanje e-mail poruke, pregled mape i dr.*
- Aplikacija može da sadrži jednu ili više definisanih aktivnosti, pri čemu je jedna od aktivnosti uvek definisana kao **primarna aktivnost**.
- Prelaz između aktivnosti vrši se tako što aktuelna aktivnost zove novu
- Iako više aktivnosti čini jedan kompaktan korisnički interfejs, treba imati na umu da su one međusobno potpuno nezavisne.
- Svaka aktivnost implementira se kao zasebna klasa koja nasleđuje klasu ***Activity***, pa je sama odgovorna za čuvanje svog stanja u životnom ciklusu aplikacije.
- Klasa aktivnosti je glavni deo svake Android aplikacije.
- Programer veći deo vremena provodi na definisanju i implementaciji aktivnosti za svaki ekran koji se pojavi u aplikaciji.

4.1 – Aktivnosti i klasa Activity

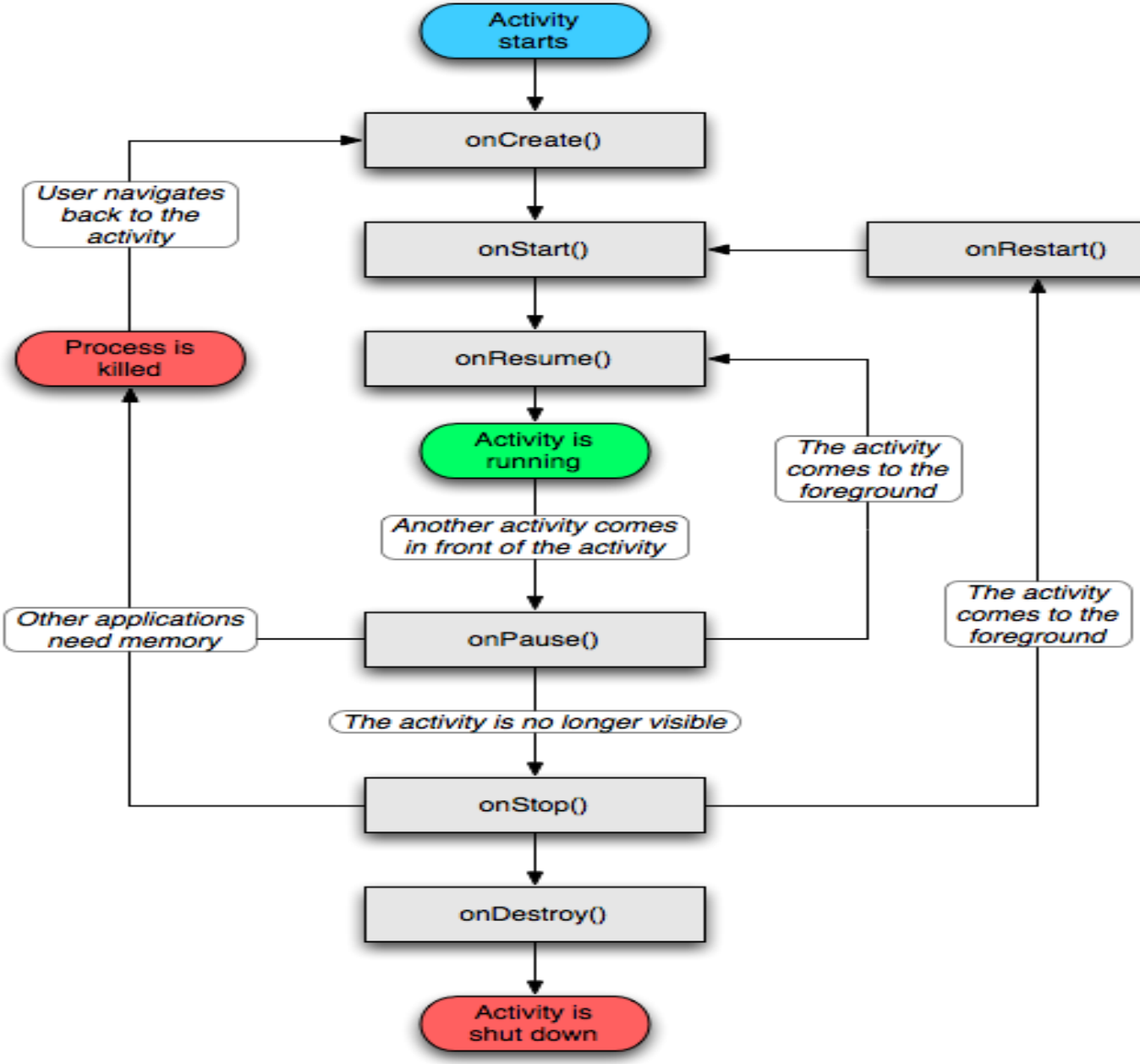
- Svakoj Android aplikaciji mora biti dodeljena **aktivnost** unutar **Android manifest fajla**.
- Druge klase aktivnosti mogu biti navedene u manifest fajlu kao i pod kojim određenim okolnostima one mogu da se pokrenu.
- Ovim sekundarnim ulaznim tačkama se upravlja pomoću Android manifest fajla određenim filterima.
- Aplikacije mogu biti prekinute zbog raznih događaja višeg prioriteta, kao što je telefonski poziv.
- U jednom trenutku može postojati **samo jedna aktivna aplikacija**.
- Android aplikacije su odgovorne za upravljanje njihovim stanjem, kao i korišćenom memorijom, resursima i podacima.
- Android OS može isključiti aktivnost koja je na pauzi, stopirana ili uništena kada nema dovoljno memorije (mala operativna memorija)
- Svaka aktivnost koja nije u prvom planu mora biti isključena
- Android aplikacija mora održavati stanje i biti spremna da bude prekinuta ili čak isključena u svakom trenutku.

4.2 – Životni ciklus aktivnosti

- Životnim ciklusom aktivnosti jedne Android aplikacije upravlja se implementacijom odgovarajućih metoda.
- Svaka aktivnost ima tri osnovna stanja:

1. ***Resumed (Running)*** - aktivnost je pokrenuta i fokusirana.
2. ***Paused*** - druga aktivnost je pokrenuta i fokusirana (u prvom planu), ali je tekuća aktivnost i dalje pokrenuta i vidljiva. Na primer, druga aktivnost je u prvom planu, dok je tekuća aktivnost vidljiva, ali ne zauzima ceo ekran. Aktivnost čije je stanje ***paused*** je potpuno „živa“ (objekat klase ***Activity*** se nalazi u memoriji, zadržava sva stanja i ostaje u ***window manager***-u), međutim ova aktivnost može da bude sklonjena u slučaju male slobodne memorije.
3. ***Stopped*** - aktivnost čije je stanje ***stopped*** je i dalje „živa“, ali radi u pozadini. Objekat klase ***Activity*** i dalje se nalazi u memoriji, čuva sva stanja, ali se više ne nalazi u okviru ***window manager***-a. Ovu aktivnost više **ne vidi korisnik** i u slučaju potrebe za dodatnom memorijom sistem može da je ukloni.

4.2 - Životni ciklus aktivnosti



1.onCreate()

2.onStart()

3.onResume()

4.onPause()

5.onStop()

6.onDestroy()

7.onRestart()

4.2 – Životni ciklus aktivnosti

- Fazama životnog ciklusa aktivnosti upravljaju 7 metoda: onCreate(), onRestart(), onStart(), onResume(), onPause(), onStop() i onDestroy().
- Aktivnosti započinje metodom ***onCreate()***, a završava sa ***onDestroy()***.
- U okviru metode ***onCreate()***, programer bi trebalo da definiše izgled i globalno stanje aktivnosti, kao raspored elemenata korisničk.interfejsa
- Implementacijom metode ***onDestroy()*** korišćeni resursi se oslobađaju.
  Primer: *ako neka aktivnost koristi pozadinsku nit koja preuzima podatke sa Interneta, nit bi trebalo kreirati u okviru metode* ***onCreate()****, a zaustaviti je u okviru metode* ***onDestroy()*****.**
- Aktivnost je vidljiva između poziva metoda ***onStart()*** i ***onStop()***
- Dok je aktivnost vidljiva, korisnik je vidi na ekranu i može da vrši interakciju sa njom (da čita ili unosi podatke).
- Metoda ***onStop()*** se poziva kada se startuje nova aktivnost, i time tekuća aktivnost više neće biti vidljiva.
- Aktivnost je fokusirana između poziva metoda onResume()i onPause()
- Aktivnost može često da prelazi iz prvog plana u pozadinu i obrnuto.
- Metoda ***onPause()*** se poziva kada uređaj prelazi u stanje mirovanja.

4.2 – Osnovne metode životnog ciklusa

Metoda	Opis	Može da se „ubije“ posle?	Sledeća metoda
`onCreate()`	Poziva se kada se aktivnost prvi put kreira. Ovde bi trebalo kreirati poglede (views), povezati (bind) podatke sa listama i sl. Ovoj metodi prosleđuje se objekat tipa Bundle koji sadrži prethodno stanje aktivnosti.	Ne	**`onStart()`**
`onRestart()`	Poziva se posle stopiranja aktivnosti, ali neposredno pre njenog ponovnog startovanja.	Ne	**`onStart()`**
`onStart()`	Poziva se neposredno pre nego što aktivnost postane vidljiva korisniku. Metoda koja se poziva posle ove može da bude `onResume()` u slučaju da aktivnost prelazi u prvi plan, ili `onStop()` u slučaju da postaje sakrivena.	Ne	**`onResume()`** **ili** **`onStop()`**
`onResume()`	Poziva se neposredno pre nego što aktivnost započne interakciju sa korisnikom.	Ne	**`onPause()`**

4.2 – Osnovne metode životnog ciklusa

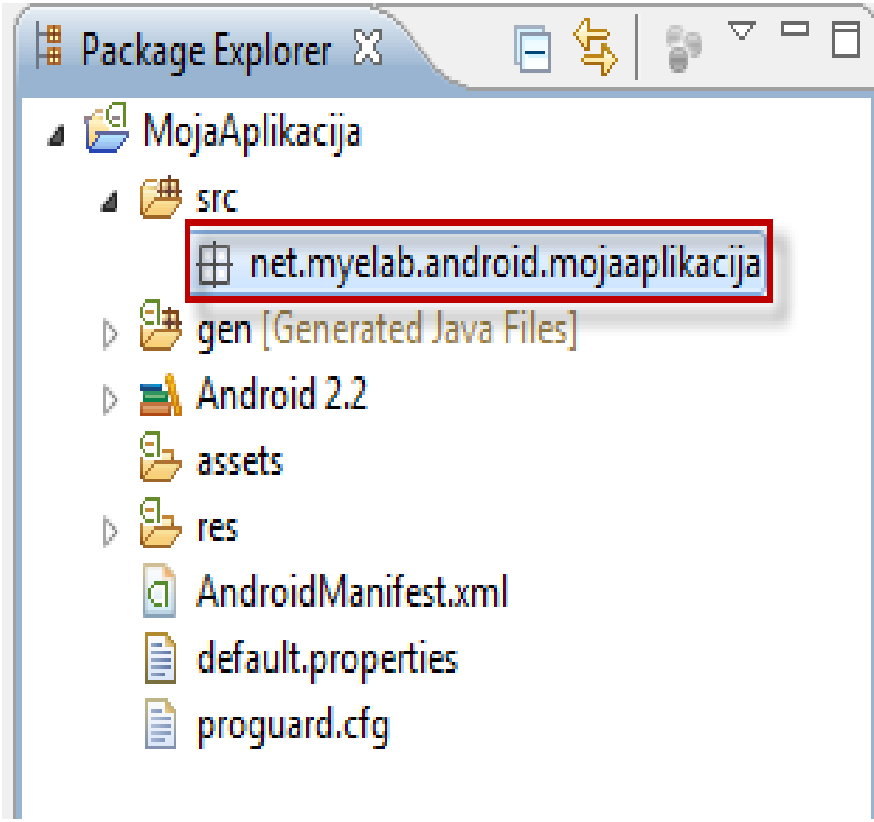
Metoda	Opis	Može da se „ubije“ posle?	Sledeća metoda
`onPause()`	Poziva se neposredno pre nego što sistem nastavi izvršavanje druge aktivnosti. Ova metoda se obično koristi da sačuva podatke, zaustavi animacije i prekine sa izvršavanjem procesa koji koriste procesorsko vreme. Ova metoda se mora izvršiti što je brže moguće, zato što sledeća aktivnost neće biti nastavljena dok se ova metoda završi. Metoda koja se poziva posle ove može da bude `onResume()` u slučaju da aktivnost prelazi u prvi plan, ili `onStop()` u slučaju da postaje sakrivena.	Da	**`onResume()`** **ili** **`onStop()`**
`onStop()`	Poziva se kada korisnik više ne vidi aktivnost. Ovo se može dogoditi kada je u toku uništavanje aktivnosti ili zbog nastavljanja izvršavanja neke druge aktivnosti.	Da	**`onRestart()`** **ili** **`onDestroy()`**
`onDestroy()`	Poziva se neposredno pre uništavanja aktivnosti. Ovo je poslednji poziv koji će aktivnost da primi. Metoda `onDestroy()` može biti pozvana kada se završava sa izvršavanjem aktivnosti (neko je pozvao metodu `finish()`) ili zato što sistem privremeno uništava instancu aktivnosti da bi oslobodio prostor.	Da	**nema**

4.2 - Životni ciklus aktivnosti

```
public class MojaAktivnost extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      // Nova aktivnost se kreira.
    }
    @Override
    protected void onStart() {
      super.onStart();
      // Kreirana aktivnost postaje vidljiva.
    }
    @Override
    protected void onResume() {
      super.onResume();
      // Aktivnost je postala vidljiva (stanje "resumed").
    }
    @Override
    protected void onPause() {
      super.onPause();
      // Druga aktivnost uzima fokus (ova aktivnost je sada "paused").
    }
    @Override
    protected void onStop() {
      super.onStop();
      // Aktivnost više nije vidljiva (sada je u stanju "stopped").
    }
    @Override
    protected void onDestroy() {
      super.onDestroy();
      // Aktivnost se uništava.
    }
}
```

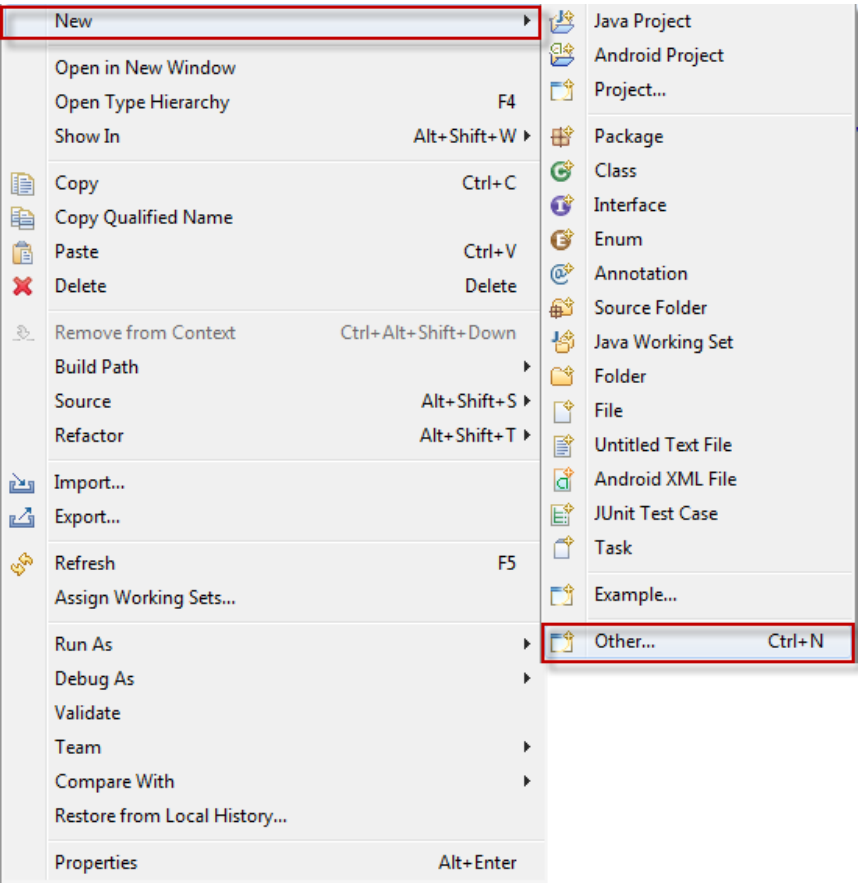
4.3 – Kreiranje aktivnosti

- Da bi se kreirala nova aktivnost u okviru Android aplikacije, kreira se nova klasa koja nasleđuje klasu **Activity** (ili njenu izvedenu klasu).
- U novokreiranoj klasi, neophodno je implementirati metode opisane u prethodnom poglavlju
- Najvažnije je implementiranje metoda **onCreate()** i **onPause()**.
- U okviru već kreiranog Android poglavlja potrebno je kreirati novu klasu.
- U okviru Package Explorera koji se najčešće nalazi sa leve strane u Eclipse razvojnom okruženju najpre treba izabrati folder ***src***, a zatim kliknuti desnim tasterom miša na paket koji je kreiran zajedno sa Android projektom.
- U našem slučaju, to je ***net.myelab.android.mojaaplikacija.***

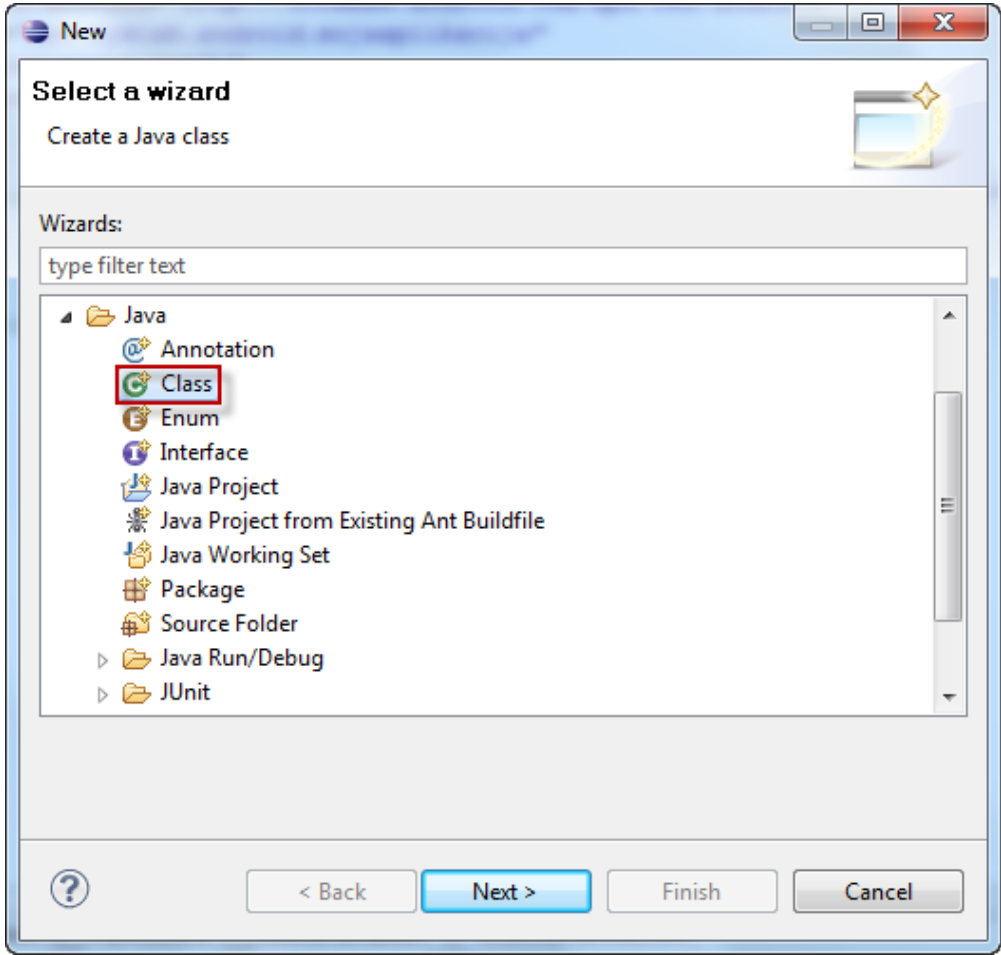


4.3 - Kreiranje aktivnosti

- Treba izabrati opciju ***New***, pa ***Other***

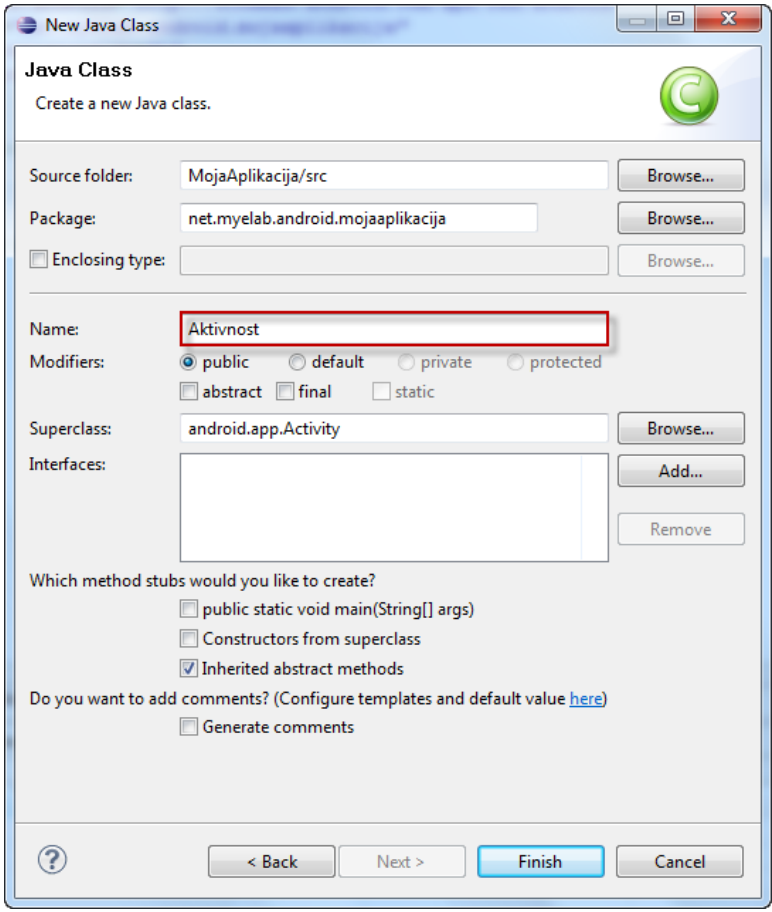


- Potrebno izabrati opciju za dodavanje nove klase. To je opcija ***Java -> Class***
- Zatim treba kliknuti na taster **Next**.



4.3 – Kreiranje aktivnosti

- Pojaviće se ekran gde je neophodno uneti:
- **Source folder** – folder u kome se čuvaju izvorni fajlovi projekta. Obično je to ***src***.
- **Package** – naziv paketa u kome se kreira klasa. Ovde bi trebalo da stoji naziv paketa koji je kreiran zajedno sa novim Android projektom. U ovom slučaju, to je ***net.myelab.android.mojaaplikacija***.
- **Name** – naziv klase. Ovde unosimo željeni naziv aktivnosti koju kreiramo. Ovde neka to bude ***Aktivnost.***
- **Modifiers** – modifikatori pristupa. Za aktivnost, ova vrednost je ***public***.
- **Superclass** – nadređena klasa. Za aktivnost, to bi trebalo da bude ***android.app.Activity***.



4.3 – Kreiranje aktivnosti

- Aktivnost koju ćemo kreirati predstavljaće ***Hello World*** primer.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    TextView tv = new TextView(this);
    tv.setText("Hello world");
    setContentView(tv);
}
```

- Jedini ulazni parametar ove metode je objekat tipa Bundle - predstavlja neko prethodno stanje aktivnosti ukoliko je aktivnost ranije pokrenuta.
- Na početku se poziva metoda **onCreate()** nadređene klase (klasa **Activity**), kojoj se prosleđuje parametar tipa Bundle.
- Kreira se novi pogled (**View**), koji predstavlja osnovnu jedinicu korisničkog interfejsa Android aplikacije.
- Android aktivnost može da koristi više različitih pogleda.
- Tekstualni sadržaj se ubacuje korišćenjem metode setText().
- Konačan izgled klase **Aktivnost** prikazan je u narednom primeru:

4.3 - Kreiranje aktivnosti

```
package net.myelab.android.mojaaplikacija;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class Aktivnost extends Activity {
        @Override
        public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello world");
        setContentView(tv);
```

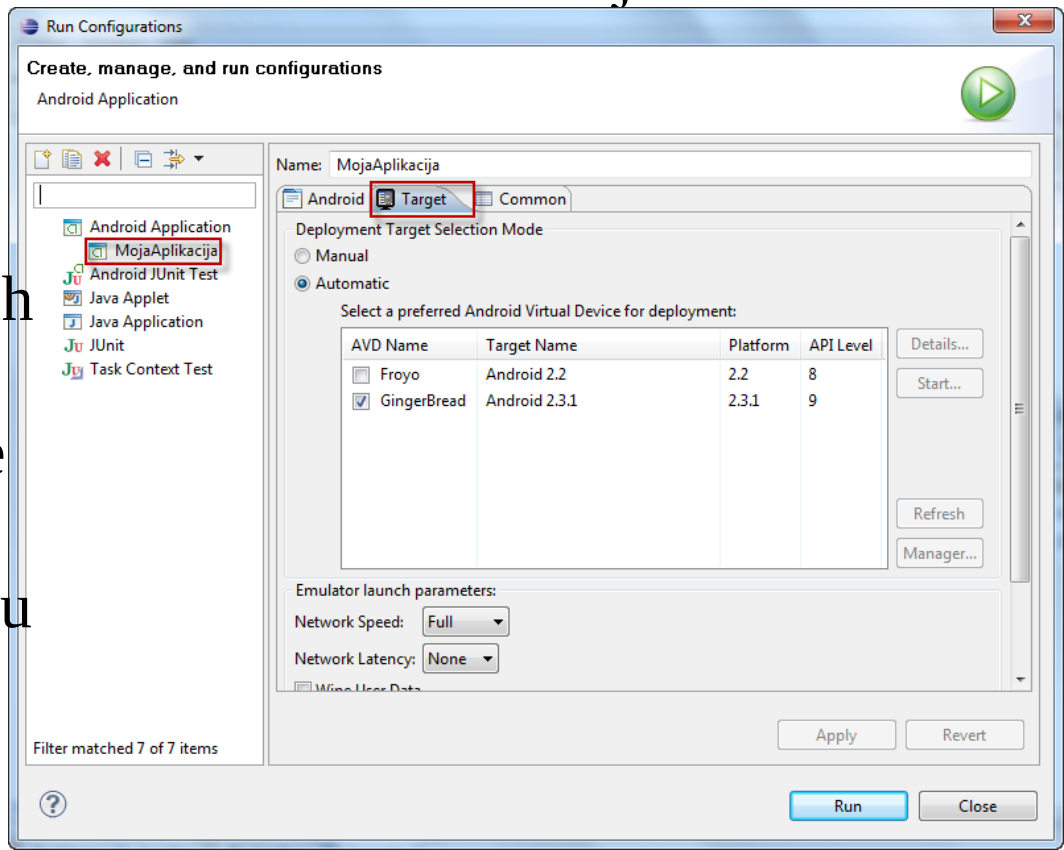
4.3 - Kreiranje aktivnosti

- Neophodno je ubaciti podatke o kreiranoj aktivnosti u fajl **AndroidManifest.xml** koji sada izgleda:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="net.myelab.android.mojaaplikacija"
android:versionCode="1"
android:versionName="1.0">
<application
 android:icon="@drawable/icon"
android:label="@string/app_name">
<activity android:name=".Aktivnost"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

4.3 – Kreiranje aktivnosti

- Tag **<activity>** omogućava aplikaciji da „zna“ da navedena klasa predstavlja aktivnost, obzirom da je moguće koristiti i pomoćne klase.
- U okviru taga **<intent-filter>** definisano je koja aktivnost predstavlja početnu aktivnost u aplikaciji.
- Izborom opcije ***Run -> Run As -> Android Application*** startuje se aplikacija na odgovarajućem Android virtuelnom uređaju.
- Startovanje virtuelnog uređaja nekada predstavlja dug proces (više od 5 min.)
- Ako ima više Android virtuelnih uređaja, željeni uređaj je moguće izabrati pomoću opcije ***Run -> Run Configurations***, a zatim je potrebno izabrati opciju ***Android Application -> Naziv Projekta***, pa karticu ***Target***



4.3 – Kreiranje aktivnosti

- Ukoliko je fizički Android uređaj povezan sa računarom preko USB porta, izborom opcije ***Run*** neće doći do startovanja virtuelnog uređaja, već će se aplikacija instalirati na fizički uređaj i automatski startovati.

4.4 – Resursi u Android aplikacijama

- Sve android aplikacije sastoje se iz dva segmenta:
  1. **funkcionalnost** – predstavlja kod kako se aplikacija ponaša na neki zahtev i obuhvata sve algoritme koji pokreću aplikaciju
  2. **resursi ili podaci** – obuhvataju tekstove, slike, audio i video fajlove, datoteke, ikone i druge podatke koje aplikacija koristi.
- Resursi se dele na dva tipa: **resurse aplikacije** i **resurse sistema**.
- Resurse aplikacije definiše programer unutar fajlova Android projekta i tačno su određeni za neku aplikaciju.
- Resursi sistema su standardni resursi koje definiše Android platforma i dostupni su svim aplikacijama kroz Android SDK gde se oni i nalaze.
- Postoje posebne klase za svaki od većih tipova sistemskih resursa.
- Resursi su organizovani, definisani i upakovani sa paketom aplikacije.
- Resursi aplikacije se ne dele sa ostatkom Anroid sistema.
- Svi resursi aplikacije se čuvaju u strukturi direktorijuma **/res** i sastavljeni su u projekat u trenutku pravljenja aplikacije.
- Resursi aplikacije se mogu koristiti programerski preko klase **R.java** i na njih se mogu odnositi resursi drugih resursa aplikacija.

4.5 – Upravljanje i pristup resursima

- Da bi se omogućila kompatabilnost sa različitim uređajima potrebno je resurse organizovati u folderima i podfolderima kako bi se svaki deo android aplikacije mogao održavati nezavisno od ostalih delova aplikacije i kako bi se na najlakši način mogla izvršiti lokalizacija i prilagođavanje aplikacija za različite konfiguracione uređaje.
- Podrazumevani stil prikazivanja aplikacije čuva se u folderu ***res/layout/***
- Međutim moguće je u konfiguracionom fajlu podesiti da kada se aplikacija pokrene na nekom uređaju koji ima drugačiju rezoluciju ekrana, npr koristi Landscape orijentaciju, da tada poziva funkcije za prikaz is foldera ***res/layout-land/***
- Na taj način sama aplikacija, zavisno od rezolucije uređaja, može prikazati adekvatan sadržaj.

4.6 – Tipovi resursa

- Korišćenje resursa je veoma jednostavno ako se koristi Eclipse ili Android Studio za razliku od nekih drugih razvojnih okruženja
- Android aplikacije koriste veliki broj različitih tipova resursa kao što su tekst, grafika, šeme u boji za dizajn korisničkog interfejsa itd.
- Preporuka je da se u programu izvrši grupisanje resursa aplikacije i njihovo spajanje u paket aplikacije jer to ima sledeće koristi:
  - ✓ Kod je jasniji i lakši za čitanje, što dovodi do smanjivanja grešaka.
  - ✓ Resursi su organizovani po tipovima i sigurno su jedinstveni
  - ✓ Resursi su dobro smešteni za prilagođavanje telefonima
- Fajlovi resursa sačuvani su u **/res** direktorijumu android aplikacije i svi oni moraju da poštuju sledeća pravila:

1. Imena fajlova resursa moraju biti napisani malim slovima
2. Imena fajlova resursa mogu sadržati samo slova, brojeve, donju crtu, i tačke
3. Imena fajlova resursa (i XML atributi imena) moraju biti jedinstveni.

4.6 – Tipovi resursa

Tip resursa	Potreban direktorijum	Naziv	XML tag
Strings	/res/values/	strings.xml	<string>
Niz stringova	/res/values/	strings.xml	<string-array>, <item>
Boolenas	/res/values/	bools.xml	<bool>
Colors	/res/values/	Colors.xml	<color>
Dimensions	/res/values/	Dimens.xml	<dimen>
Integers	/res/values/	integers.xml	<integer>
Mešovit niz	/res/values/	Arrays.xml	<array>, <item>
Jed. elementi (Drawables)	/res/values/	drawables.xml	<drawable>
Graphics(slike)	/res/drawable/	Icon.png, logo,jpg itd	
Meni	/res/menu/	mainmenu.xml helpmenu.xml	<menu>
XML fajlovi	/res/xml/		
Izgled(Layouts)	/res/layout/	main.xml	
Stilovi i teme	/res/values/	styles.xml themes.xml	<style>
Animacija	/res/drawable/	sekvenca1.xml sekvenca2.xml	<animation-list>, <item>

4.6 – Rad sa stringovima

- String resursi su najjednostavniji tip resursa sa kojima može da se radi
- Stringove možete promeniti ili dodati editovanjem datoteke **strings.xml**.
- String se čuva u okviru taga **<string>**.
- U prilogu sledi primer kako izgleda strings.xml

```
<?xml version=”1.0” encoding=”utf-8”?>
<resources>
<string name=”naziv_aplikacije”>Moja prva aplikacija</string>
<string name=”test_string”>Test 1,2,3</string>
<string name=”test_string2”>Test 4,5,6</string>
<string
   name=”formatiran_text”><b>Bold</b>,<i>Italic</i>,<u>Line</u></string
   >
</resources>
```

- U narednoj liniji koda prikazano je kako možete određenom stringu u xml fajlu pristupiti i koristiti u aplikaciji:

```
String nazivAplikacije = getResources().getString(R.string.naziv_aplikacije);
```

4.3 – Rad sa nizom stringova

- Rad sa nizom stringova je veoma pogodan kada želimo da u nekoj padajućoj listi prikažemo više opcija.
- Niz stringova se u XML fajlu definiše na putanji **rez/values/** i čuvaju se u okviru tagova **<string-array>** i tagom za predstavljanje elemenata niza **<item>**. Na putanji **rez/values/arrays.xml** čuva niz stringova:

```
<?xml version=”1.0” encoding=”utf-8”?>
<resources>
<string-array name=”predmeti”>
   <item>Elektronsko poslovanje</item>
   <item>Internet tehnologije</item>
   <item>Mobilni operativni sistemi</item>
</string-array>
<string-array name=”Predavaci”>
   <item>MirkoKosanovic</item>
   <item>SlavimirStosovic</item>
</string-array>
</resources>
```

- U datom primeru pristup resursu **predmeti** radimo na sledeći način:

```
String[] predmeti = getResources().getStringArray(R.array.predmeti);
```

4.6-Rad sa Boolean i Integer resursima

- Boolean vrednosti se čuvaju na putanji **rez/values/bools.xml**.
- Ovako izgleda primer čuvanja Bool vrednosti:

```
<?xml version=”1.0” encoding=”utf-8”?>
<resources>
<bool name=”podrzanSrpskiJezik”>true</bool>
<bool name=”podrzanIzborJezika”>false</bool>
</resources>
```

- Resursu iz Android aplikacije pristupa se na sledeći način:

```
boolean podrzanIzborJezika = getResources().getBoolean(R.bool.podrzanIzborJezika);
```

- Slično se radi i sa Integer resursima, jedina razlika jestu tagovi koji se koriste i putanja na gde se čuvaju integer resursi **rez/values/nums.xml**

```
<?xml version=”1.0” encoding=”utf-8”?>
<resources>
<integer name=”sabirakA”>25</integer>
<integer name=”sabirakB”>3</integer>
</resources>
```

- Pristup integer resursima iz Android alikacije izgleda ovako:

```
int sabirakA = getResources().getInteger(R.integer.sabirakA);
```

4.6 – Rad sa bojama

- Android aplikacija skladišti vrednosti RGB boje, što automatski obezbeđuje da se te boje primene i na druge elemente ekrana
- Vrednosti koje se čuvaju, mogu se iskoristiti za podešavanje boje teksta ili pozadine ekrana apliakcije.
- Vrednosti za boje čuvaju se u fajlu **rez/values/colors.xml**.
- Android podržavaja rad sa sledećim formatima boja:
  1. **#RGB** (primer, #F00 , 12-bit boja red)
  2. **#ARGB** (primer, #8F00 , 12-bit boja, crvena transparetnost do 50%)
  3. **#RRGGBB** (primer, #FF00FF, 24-bit boja, purpurcrvena)
  4. **#AARRGGBB** (primer, #80FF00FF , 24-bit boja, purpurcrvena)
- Primer skladištenja boja:

```
<?xml version=”1.0” encoding=”utf-8”?>
<resources>
<color name=”background_color”>#006400</color>
<color name=”text_color”>#FFE4C4</color>
</resources>
```

- Sledeća linija koda vraća vrednost određene boje:

```
int mojaBoja = getResources().getColor(R.color.mojaBoja);
```

4.6 – Rad sa resursima koji se crtaju

- Resursi koji se crtaju (animacija), kao što su slike, moraju se sačuvati u **/res/drawable** direktorijumu projekta.
- Ovi tipovi resursa se onda sastavljaju u paket aplikacije i dostupne su aplikaciji.
- Najčešći resursi koji se crtaju su bitmap slike, kao što su PNG i JPG
- Ovi fajlovi se često koriste kao ikonice aplikacije i slike dugmića ali se mogu koristiti i za veliki broj komponenti korisničkog interfejsa.
- Resursi slika su u klasi ***BitmapDrawable***.
- Kako bi se pristupilo grafičkom resursu koji se zove **/res/drawable/logo.png**, treba se koristiti komanda ***getDrawable()***, po sledećem redosledu:

BitmapDrawable logoBitmap = (BitmapDrawable)getResources().getDrawable(R.drawable.logo);

- Većinu vremena, međutim, grafika ne mora da se otvara direktno.
- Umesto toga, može se koristiti identifikator resursa kao atribut za kontrolu kao što je kontrola **ImageView**.

4.6 – Rad sa resursima koji se crtaju

```
<ImageView
    android:id="@+id/LogoImageView "
    android:src="@drawable/logo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
```

- Sledeći kod, na primer, postavlja i otvara **logo.png** sliku u kontroli **ImageView** koja se zove **LogoImageView**, koja mora biti definisana unutar fajla za layout (main.xml):

```
ImageView logoView = (ImageView)findViewById(R.id.LogoImageView);
logoView.setImageResource(R.drawable.logo);
```

- Takođe se može kreirati specijalni XML fajl kako bi se opisale druge **Drawable** podklase, kao što je **ShapeDrawable**.
- Može se koristiti klasa **ShapeDrawable** da bi se definisali razni oblici, kao što je pravougaonik i krug.

4.6 – Rad sa menijima

- Kod razvoja aplikacije, stalno se javlja potreba za kreiranjem menija
- Andird podržava rad sa menijima, meni se kao i ostali resursi skladište u direktorijumu **rez/menu/**.
- Svaki resurs menija, koji predstavlja skup stavki menija, se pamti kao posebno formatirana XML datoteka i kompajliraju se u vreme izvršavanja aplikacije.
- Na slici je dat jednostavni meni koji je skladišten u **rez/menu/speed.xml**

```
<menu xmlns:android
=”http://schemas.android.com/apk/res/android
”>
<item
android:id=”@+id/start”
android:title=”Startuj aplikaciju!”
android:orderInCategory=”1”></item>
<item
android:id=”@+id/stop”
android:title=”Stopiraj aplikaciju!”
android:orderInCategory=”4”></item>
<item
android:id=”@+id/ubrzaj”
android:title=”Ubrzaj aplikaciju!”
android:orderInCategory=”2”></item>
<item
android:id=”@+id/uspori”
android:title=”Uspori aplikaciju!”
android:orderInCategory=”3”></item>
</menu>
```

4.6 – Rad sa menijima

- Meni se može kreirati i korišćenjem Eclipse dodatka koji može pristupati konfiguracionim atributima za svaku stavku menija.
- U predhodnom primeru smo direktno u xml fajl postavili naziv stavke menija, inače ti stringovi se takođe mogu izdvojiti iz tog fajla i kasnije lokalizovati na više jezika.

Primer: prikaz menija gde su definisani samo nazivi resursa, gde se vrednost resursa uzima kasnije iz konkretnog xml fajla:

```
<menu xmlns:android=
“http://schemas.android.com/apk/res/android”>
<item
android:id=”@+id/start”
android:title=”@string/start”
android:orderInCategory=”1”></item>
<item
android:id=”@+id/stop”
android:title=”@string/stop”
android:orderInCategory=”2”></item>
</menu>
```

4.6 – Rad sa menijima

- Da bi se koristio predhodno napisan meni potrebno je u aplkaciji pozvati resurs koji je skladišten na putanji **rez/menu/speed.xml** tako što se overajduje metoda nadređene klase **onCreateOptionsMenu()** u android aplikaciji.

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

- Sada, ako se pokrene aplikacija i pritisne taster meni dobiće se meni koji je gore definisan.
- Postoji veliki broj atributa koji se može dodeliti nekoj stavki menija, spisak svih stavki možete pogledati u zvaničnoj dokumentaciji za Andoird SDK

http://d.android.com/guide/topics/resources/menu-resource.html

4.6 – Rad sa fajlovima

- Kao dodatak stringu, grafici i layout resursima, Android projekti mogu sadržati fajlove kao resurse.
- Ovi fajlovi mogu biti u bilo kom formatu.
- XML fajl format je dobro podržan na Android platformi.
- Proizvoljni XML fajlovi mogu biti uključeni kao resursi.
- Ovi XML fajlovi se čuvaju u **/res/xml direktorijumu**.
- XML fajlovi su preferirani format za bilo koje strukturisane podatke koje aplikacija traži.
- Kako se formatira XML fajl zavisi od programera.
- Različiti XML alati su dostupni za Android platformu.
- Primer xml fajla:

```
<?xml version=”1.0” encoding=”utf-8”?>
<predmeti>
<predmet name=”Mobilno poslovanje” type=”izborni” />
<predmet name=”Elektronsko poslovanje” type=”obavezan” />
<predmet name=”ITEH” type=”izborni” />
</predmeti>
```

4.6 – Rad sa fajlovima

- XML resursu iz aplikacije može se pristupiti na sledeći način:

```
XmlResourceParser mojiPredmeti = getResources().getXml(R.xml.mojiPredmeti);
```

- Aplikacija može sadržati **.raw** fajlove kao resurse.
- Raw fajlovi koje aplikacija koristiti sadrži audio, video i druge fajlove.
- Svi raw fajlovi resursa bi trebali biti sačuvani u **/res/raw** direktorijum.
- Nema pravila ili ograničenja pri kreiranju raw fajlova (osim pravila o nazivu fajla koji je ranije pomenut).
- Ako treba ubaciti multimedia fajlove, treba proveriti dokumentaciju Android platforme kako bi se odredili koji su formati i kodeci podržani
- Ako format fajla koji treba da se koristi nije podržan u Android sistemu, aplikacija će možda zatražiti da sama odradi proveru fajlova.
- Da bi se pristupilo raw fajlu programerski, jednostavno treba koristiti komandu **openRawResource()**.
- Na primer, sledeći kod bi napravio InputStream objekat da bi pristupio resurs fajlu /res/raw/file1.txt:

```
InputStream iFile = getResources().openRawResource(R.raw.file1);
```

4.6 - Rad sa rasporedom(Layout)

- Većina korisničkih interfejsa Android aplikacija se definišu koristeći specijalni XML fajl koji se zove **layout**.
- Resursi rasporeda su u **/res/layout** direktorijumu.
- **Layout** fajlovi se sastavljaju u aplikaciju kao i svaki drugi resurs.
- Oni često definišu ceo ekran ili deo ekrana
- Mogu da budu povezani sa određenom aktivnošću, ali i ne moraju
- Resursi rasporeda takođe mogu biti deo nekog drugog rasporeda.

```
<?xml version=”1.0” encoding=”utf-8”?>
<LinearLayout
xmlns:android=”http://schemas.android.com/apk/res/android”
android:orientation=”vertical”
android:layout_width=”fill_parent”
android:layout_height=”fill_parent”>
<TextView
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”@string/hello” />
</LinearLayout>
```

4.6 – Rad sa rasporedom(Layout)

- Ovo je običan raspored, koji se zove **main.xml**, koji se pravi sa svakom Android aplikacijom.
- Ovaj layout file opisuje korisnički interfejs jedine aktivnosti u aplikaciji
- Sadrži kontrolu **LinearLayout** koja se koristi kao skup za sve druge kontrole korisničkog interfejsa.
- U ovom slučaju to je kontrola **TextView.**
- **Main.xml layout** fajl sadrži i druge resurse: string resurs @string/hello, koji je definisan u strings.xml resurs fajlu koji sadrži tekst ***Moja prva Android aplikacija!***
- Nakon pokretanja aplikacije, izgled ekrana je sledeći:

- Postoje dva načina da se formiraju resursi rasporeda.
- Najjednostavniji način je da se koristi **Layout Resource Editor** u Eclipsu da bi se dizajnirali layout fajlovi.
- Takođe se mogu direktno menjati **XML layout** fajlovi.

4.6 – Rad sa stilovima

- Korisnički interfejs Andoird aplikacije, dizajneri mogu uređivati korišćenjem stilova.
- Kontrola rasporeda elemenata izvedena je iz **View** klase, koja ima veliki broj korisnih osobina.
- Stilovi se označavaju tagom **<style>** koji su smešteni u direktorijum **rez/values/** .
- Stilovi se takođe definišu u **XML fajlu** a potom se kompajliraju u trenutku izvršavanja aplikacije.
- Stilovi se ne mogu gledati korišćenjem Eclipse Resource designer-a ali se ispravno prikazuju na emulatoru kao i na samom uređaju.
- U nastavku prikazan je jednostavan primer kako se koriste stilovi koji su smešteni u fajlu **rez/values/styles.xml**.
- Definisana su dva stila, prvi stil je za obavezna polja u formularu, dok drugi je za neobavezna polja.
- Obavezna polja biće predstavljena crvenom bojom i veličinom slova 14p koja će biti boldovana, dok opciona plja biće predstavljena belom bojom, iskošena i veličina 12p.

4.6 – Rad sa stilovima

```
<?xml version=”1.0” encoding=”utf-8”?>
<resources>
<style name=”obavezna_polja_style”>
<item name=”android:textColor”>#e4101a</item>
<item name=”android:textSize”>14pt</item>
<item name=”android:textStyle”>bold</item>
</style>
<style name=”opciona_polja_style”>
<item name=”android:textColor”>#FFFFFF</item>
<item name=”android:textSize”>12pt</item>
<item name=”android:textStyle”>italic</item>
</style>
</resources>
```

4.6 – Rad sa temama

- Tema predstavlja skup stilova koji se primenjuju na sve elemente neke aktivnosti.
- Teme se definišu na isti način kao i stilovi, koristi se tag **<style>** koji se čuva u **rez/values/** direktorijumu.
- Jedina razlika što temu od stilova izdvaja jeste to što tema mora de se definiše kao atribut aktivnosti u AndoirdManifest.xml fajlu.

```
<application android:theme="@style/CustomTheme">
<activity android:theme="@style/CustomTheme">
```

4.6 – Referenciranje sistemskih resursa

- Sistemski resursi se koriste na isti način kao i resursi same aplikacije.
- Android paket sadrži sve vrste resursa, koje možete pretraživati u **android.R** podklasu.
- U ovoj podklasi mogu se naći sledeći sistemski resursi za:
  - ✓ Animiranje sekvence ka unutra ili ka spolja
  - ✓ Liste imejlova, telefona itd.
  - ✓ Standardni sistem boja,
  - ✓ Dimenzije i primenu slika i ikona,
  - ✓ Sistem stilova i tema,
  - ✓ Poruke o greškama,
  - ✓ Kreiranje rasporeda elemenata kao i crtanje jednostavnih elemenata.
- Refernciranje sistemskih resursa je na isti način kao i kada se koriste spostveni resursi.
- Sistemskim resursima se pristupa na isti način kako se pristupa resursima aplikacije: **android.R.string.ok.**

Primer: ako želimo da podesimo da boja teksta bude svetlo siva potrebno je podesiti atribut **@android:color/darker_gray**.

Hvala na pažnji !!!

Pitanja

? ? ?