ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

ANDROID APLIKACIJA ZA E MODEL IZ ITU-T G.107 PREPORUKE

–Diplomski rad–

Kandidat:

Miloš Pajić 2010/316

Mentor:

doc. dr Zoran Čiča

Beograd, Septembar 2014.

SADRŽAJ

1. UVOD

Prilikom planiranja i projektovanja paketske mreže u kojoj će se koristiti servis telefonskog razgovora, neophodno je da se napravi analitički model za procenu kvaliteta govora. Taj model treba da bude zasnovan na uticaju parametara govorne veze na subjektivni doživljaj korisnika, pri čemu fizička merenja vrednosti parametara u okviru testiranja, daju njihove objektivne vrednosti koje se koriste samom modelu [2]. Prema preporuci ITU-T (*Telecommunication Standardization Sector of the International Telecommunications Union*) G.107 definiše se E model koji se može koristiti za predikciju kvaliteta govornog signala, ukoliko su unapred poznati parametri govorne veze. Ovaj model se koristi za više vrsta kodera govornog signala, jer je zasnovan na univerzalnom principu izračunavanja kvaliteta govorne veze, dakle dovoljno je samo promeniti parametre vezane za sam koder. Parametri za neke od najčešće korišćenih kodera mogu se naći u preporuci ITU-T G.113.

Android predstavlja operativni sistem baziran na Linux jezgru, prvenstveno dizajniran za mobilne uređaje sa ekranom osetljivim na dodir. Korisnički interfejs android aplikacija je zasnovan na direktnoj manipulaciji objektima na ekranu, korišćenjem ulaza u vidu dodira koje odgovaraju pokretima u realnom svetu. Aplikacije se razvijaju u Java programskom jeziku, korišćenjem SDK (*Android Software Development Kit*). Ovaj paket sadrži niz razvojnih alata, kao što su program za pronalaženje grešaka, softverske biblioteke, emulator realnog uređaja, dokumentacija, primeri koda i priručnici. Zvanično podržano integrisano razvojno okruženje je softverski alat *Eclipse* uz korišćenje dodatka ADT (*Android Development Tools).* Android operativni sistem poseduje aktivnu zajednicu programera i entuzijasta koji koristeći njegov izvorni kod razvijaju sopstvene verzije ovog operativnog sistema. Ove verzije često donose novine na uređaje mnogo brže od proizvođača/operatera, iako bez detaljnijih testiranja i garancije kvaliteta. Najčešće dolaze prethodno „ rutovane", a sadrže izmene koje nisu podesne za korisnike koji nemaju znanja iz ovih oblasti, kao što je povećanje radnog takta ili povećanje/smanjenje radnog napona procesora. *Cyagen Mod* je najčešće korišćen firmver razvijen od strane zajednice, i predstavlja osnovu za brojne druge [3].

Glavna motivacija ove teze jeste integracija analitičkog modela za procenu kvaliteta govora, tj. E modela u android razvojno okruženje. Ovim se postiže lako dostupan program pomoću koga se mogu vršiti željena testiranja parametara paketske mreže koja podržava telefonski servis. Jedna od funkcija ovog programa postoji na sajtu ITU-T pod imenom *E model Calculaton Tool*, koja vrši izračunavanje ocene kvaliteta veze za koder G.107. Cilj aplikacije jeste da se implementira pomenuta funkcija, sa većim izborom kodera, kao i mogućnošću crtanja jednostrukih i višestrukih 2D grafika izabranih parametara. Takođe treba napomenuti da za pokretanje aplikacije nije potrebna internet konekcija, dok za aplikaciju koja se nalazi na sajtu ITU-T jeste.

Ostatak teze je organizovan na sledeći način. Drugo poglavlje se bavi samim android aplikacijama, dakle opisuje samo razvojno okruženje, pokretanje/debagovanje aplikacija, korišćene programske alate itd. U trećem poglavlju se objašnjava sama struktura koda i funkcije aplikacije, kao i pojedini interesantni delovi koda. Četvrto poglavlje predstavlja uputstvo za upotrebu same aplikacije sa primerima rada. Na kraju je u petom poglavlju dat finalni zaključak teze.

2. ANDROID APLIKACIJE

2.1. Razvojni sistem

Kao što je navedeno u prethodnom poglavlju, Android je operativni sistem koji je baziran na Linux jezgru. Svi uređaji koji podržavaju android okruženje dele određene funkcije zbog same prirode operativnog sistema. Sam android OS (*Operating system*) je podeljen na sledeće celine:

- *Bootloader* – inicira učitavanje fajla *boot image* prilikom pokretanja sistema
- *Boot image* – kernel i RAM (*Random-Access Memory*) disk
- *System image* – platforma android operativnog sistema i aplikacije
- *Data image* – korisnički podaci koji su sačuvani prilikom procedure paljenja/gašenja samog hardvera
- *Recovery image* – fajlovi koji se koriste za reinstaliranje ili nadogradnju operativnog sistema
- *Radio image* – fajlovi radio hardvera

Ove funkcije se čuvaju na fleš memoriji koja može da čuva podatke čak i kad nije napajana električnom energijom, tj. *nonvolatile memory*. Samim tim podaci su sigurni i kada se uređaj ugasi. Fleš memorija se koristi kao *read-only* memorija, tj iz nje se može samo čitati (zato je neki zovu ROM (*Read-only memory*)), ali se može i upisivati ukoliko je neophodno (na primer, prilikom nadogradnje sistema) [1].

Da bi se omogućilo programiranje android aplikacija na Windows operativnom sistemu, neophodna je dodatna instalacija potrebnih programa kao i neka podešavanja samog sistema. Sam proces instalacije i podešavanja je razložen u korake radi bolje preglednosti i razumevanja.

1) **JAVA JDK –** Pošto se android programi pišu u Javi potrebno je instalirati JDK (*Java SE Development Kit*). On sadrži biblioteke i funkcije koje koriste svi programi pisani u Javi. JDK treba razlikovati od JRE (*Java SE Runtime Enviroment*) koji je neophodan da bi se pokretale java aplikacije, ali nije dovoljan da bi se razvijale. Preuzimanje sa interneta je najbolje vršiti direktno sa http://www.oracle.com/technetwork/java/javase/downloads/index.html stranice.
2) **ADT Bundle for Windows –** Predstavlja besplatan paket sa osnovnim alatima za razvoj android aplikacija. Može se skinuti sa stranice: http://developer.android.com/sdk/index.html. Nije potrebna nikakva instalacija već se samo raspakuje .zip fajl (reporuka je da se arhiva raspakuje u folder koji ne sadrži razmake – na primer, C:\adt-bundle-windows-x86_64). Sam paket se sastoji od tri glavne komponente:

a) **Eclipse IDE (*Integrated development enviroment*)** – Predstavlja besplatno razvojno okruženje koje se prvenstveno koristi za razvoj Java aplikacija. Razvojno okruženje obuhvata editor koda, grafički editor interfejsa, kompajler za pravljenje izvršnih aplikacija, debager za debagovanje koda itd [3].

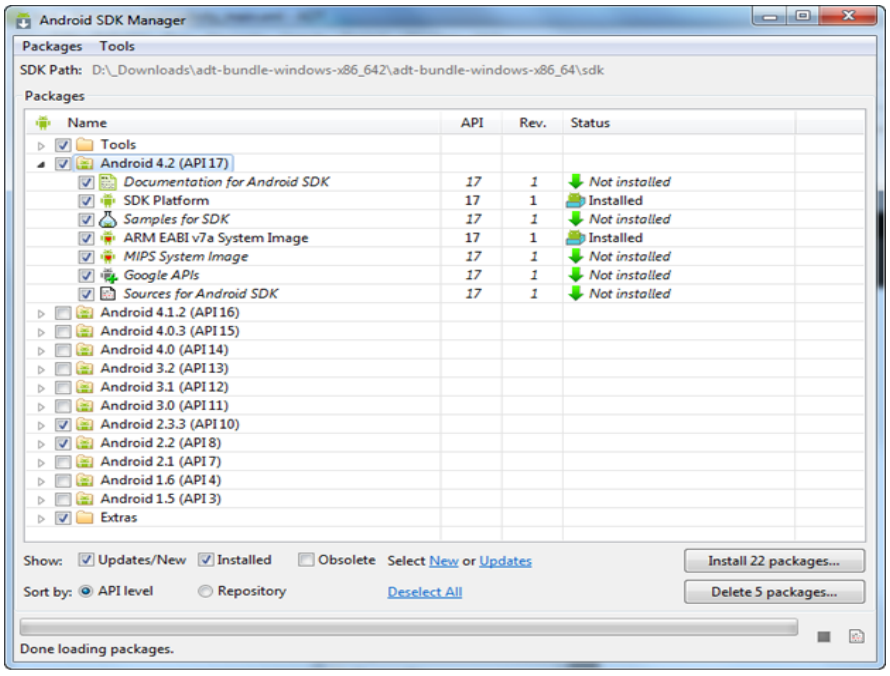
b) **ADT –** Predstavlja dodatak za *Eclipse*, koji proširuje funkcionalnost samog programa i pruža mogućnost automatskog pravljenja Android projekata sa podrazumevanom strukturom, testiranje programa u emulatorima itd [3].

c) **Android SDK Tools –** Predstavlja skup zvaničnih android alata koji podrazumevaju Android API (*Application Program Interface*), Google API, virtuelne mašine koje se koriste u emulatorima za testiranje android aplikacija [3].

Da bi pokrenuli Eclipse potrebno je startovati C:\adt-bundle-windows-x86_64\eclipse\eclipse.exe, odnosno direktorijum gde je otpakovan ADT Bundle za Windows. Pri prvom pokretanju programa, biće potrebno podešavanje lokacije radnog prostora (*workspace*), gde će se čuvati svi projekti. Podešavanje lokacije je ostavljeno korisniku da odredi gde mu najviše odgovara.

Poslednji korak jeste instaliranje potrebnih API verzija. U SDK Tools koji se nalazio u paketu nisu uključene sve potrebne verzije API interfejsa i treba ih posebno instalirati pomoću alatke SDK Manager. U programu *Eclipse*izabere se meni *Window*, zatim *Android SDK Manager*.

U njemu treba štiklirati sve stavke koje su potrebne za traženi android projekat. Savetuje se da se skinu sve moguće verzije API interfejsa, i samim tim se otklanja mogućnost nedostatka funkcija potrebnih pri kreiranju projekta. Takođe treba štiklirati i foldere Tools i Extras kao na slici 2.1.1. Android SDK Manager.

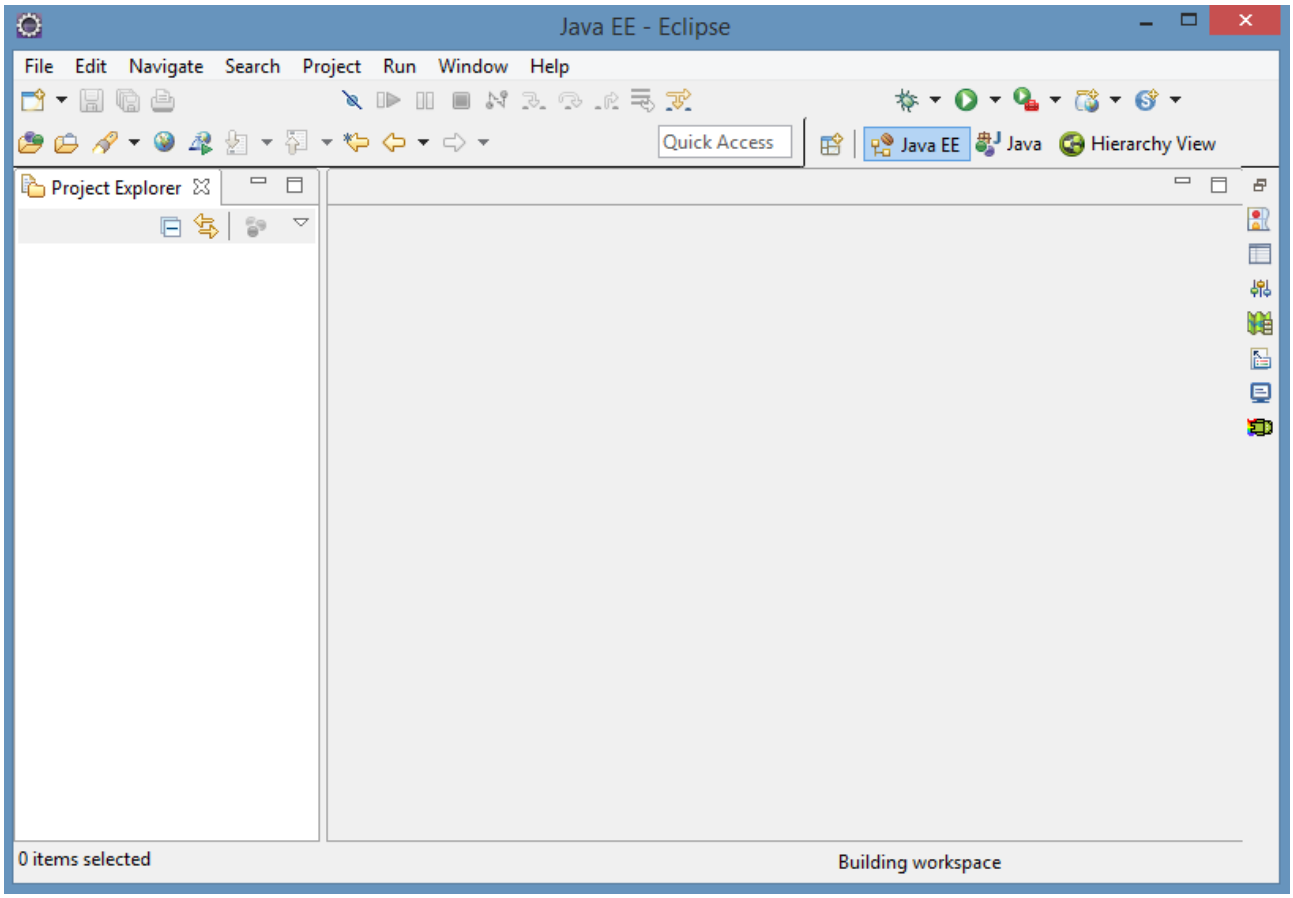


Slika 2.1.1. Android SDK Manager

Nakon toga izvrši se instalacija selektovanih fajlova klikom na dugme *Install packages*. Time je završena priprema razvojnog okruženja za programiranje android aplikacija.

2.2. Eclipse IDE

Nakon završene instalacije potrebnih programa i dodataka takođe je potrebno importovati određene pomoćne android biblioteke koje će se koristiti u samom projektu kao osnova samog koda. Pri prvom pokretanju programa *Eclipse* korisniku će biti prikazan ekran kao na slici 2.2.1. Početni ekran Eclipse IDE.



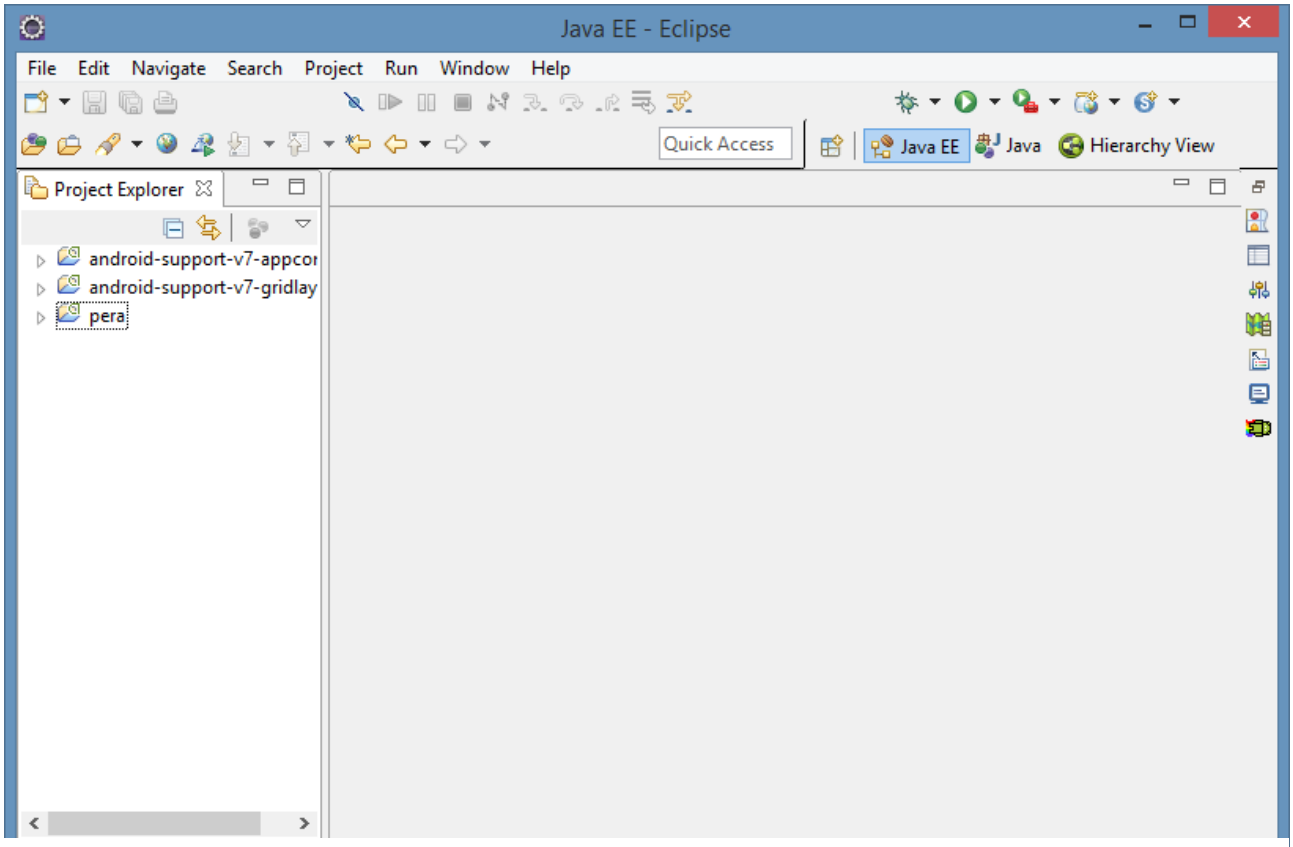
Slika 2.2.1. Početni ekran Eclipse IDE

Slika 2.2.1 Početni ekran Eclipse IDE predstavlja standardni prikaz početnog prozora, koji može da varira u zavisnosti od verzije samog programa. Nov android projekat se kreira tako što se izabere opcija *File->New->Other* a zatim se u folderu Android selektuje *Android Application Project*. U sledećem prozoru se definiše ime android projekta, kao i opseg API interfejsa (odnosno verzija android operativnog sistema). Preporučljivo je da gornji opseg prati poslednju verziju API interfejsa kako bi aplikacija radila na najnovijim uređajima sa android platformom. Za potrebe ove

teze nećemo zalaziti u pojedinosti kreiranja novog android projekta, tako da je dovoljno kliktati na dugme Next do završetka dijaloga.

Nakon uspešnog kreiranja android projekta, sam program će prijaviti par grešaka koje su posledica nedostatka osnovnih pomoćnih biblioteka koje su ranije pomenute u tekstu. Postupak dodavanja tih pomoćnih biblioteka je sledeći:

1. *File->Import->Existing Projects into Workspace*
2. Zatim se selektuje dugme *Browse* i izvrši se navigacija na sledeći folder: C:\Users\Ime_korisnika\AppData\Android\extras\android\support\v7\appcompat (folder AppData je nevidljiv tako da treba uključiti opciju *Show hidden folders*)
3. Ponoviti postupak za fajl u okviru v7 foldera pod nazivom gridlayout
4. Nakon toga biblioteke bi trebalo da se pojave kao na slici 2.2.2. Dodavanje pomoćnih biblioteka
5. Zatim se izabere*Project->Properties->Android*, pri dnu prozora klikne dugme Add i selektuje appcompat i gridlayout

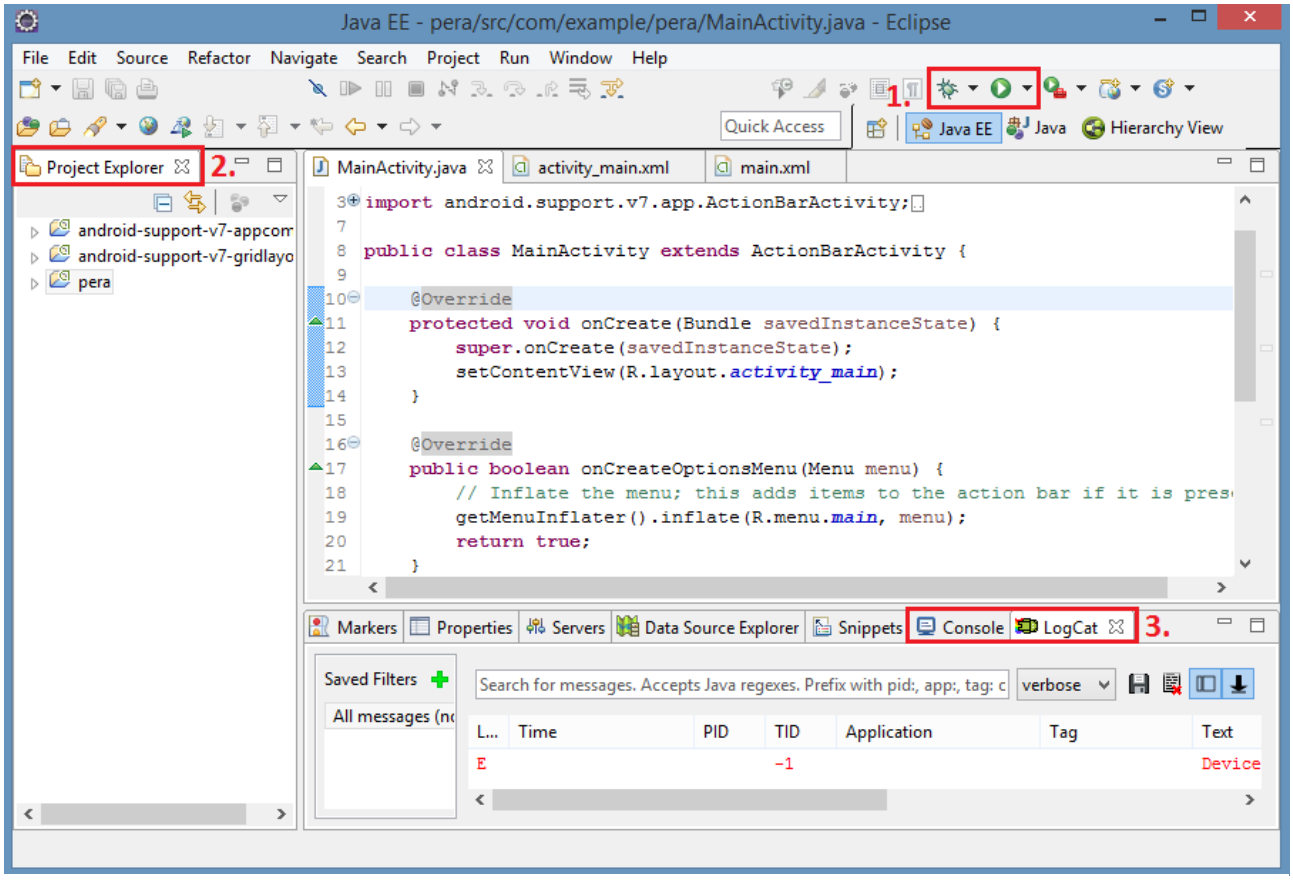


Slika 2.2.2. Dodavanje pomoćnih biblioteka

Postoji mogućnost da Java sama automatski implementira ovu biblioteku, ukoliko to nije slučaj treba primeniti postupak koji je naveden u tekstu. Android projekat je podeljen na 3 celine: **gen** folder koji sadrži java klase, **layout** folder (nalazi se u **res** folderu) koji sadrži xml fajlove vezane za grafički izgled same aplikacije, **values** folder (takođe u **res** folderu) u kome se nalaze vrednosti stringova, definišu se stilovi margine itd. Takođe treba pomenuti i AndroidManifest.xml koji svaka aplikacija mora da ima baš pod tim imenom a on služi da predstavi važne informacije

android sistemu (imena java paketa, opisiuje komponente aplikacije, privilegije, listu biblioteka itd.).

Potrebno predznanje za programiranje android aplikacija predstavlja osnovne koncepte razumevanja načina na koji rade funkcije, *for* i *while* petlje, uslovne naredbe *if*, tipovi podataka i kastovanje, rad sa matricama itd. U principu ukoliko se korisnik solidno snalazi u programskom jeziku *C*, odnosno *C++*, ne bi trebalo da ima ikakvih problema u Javi. Naravno, za samo upoznavanje i rad sa novim funkcijama postoje solidna objašnjenja na internetu kao i dosta primera na osnovu kojih se lako može razumeti upotreba istih. Za početak na slici 2.2.3. Glavni delovi softverskog alata Eclipse je prikazano gde se nalaze bitne stvari u softverskom alatu *Eclipse*:



Slika 2.2.3. Glavni delovi softverskog alata Eclipse

1) **Pokretanje/debagovanje programa:** Pokretanje se vrši preko ikonice Run gde se u padajućem meniju može izabrati koja se od aplikacija želi pokrenuti. Nakon toga postoje dve opcije emuliranja aplikacije: virtuelna i realna. Kod virtuelnog emuliranja može se definisati virtuelni mobilni uređaj kome se može definisati veličina ekrana, RAM memorija, memorija na SD (*Secure Digital*) kartici, tip procesora itd. Prilikom realnog emuliranja mora postojati fizički android mobilni uređaj koji se spoji preko USB *(Universal Serial Bus)* kabla i na samom uređaju se omogući opcija *USB Debugging.* Prethodno je potrebno instalirati drajvere za sam mobilni uređaj kako bi ga Windows operativni sistem prepoznao. To su tzv. OEM (*Original Equipment Manufacturer*) USB android drajveri koji se mogu naći na http://developer.android.com/tools/extras/oem-

usb.html. Zanimljiva je činjenica da je mnogo brže pokretanje programa realnim emuliranjem nego virtuelnim, a razlog za to je što se samo emuliranje prebacuje na procesor android uređaja, za razliku kod virtuelnog gde procesor na računaru postaje previše opterećen što rezultuje lošim odzivom na komande i sporim pokretanjem emulatora. Debagovanje u sofverskom alatu *Eclipse* se pokreće tako što se u padajućem meniju *Window* izabere nova kartica *Open Perspective* i tu izabere opcija *Debugging*. Ova opcija otvara nove prozore na osnovnom korisničkom interfejsu koji imaju sledeću ulogu: **Debug** - predstavlja prethodno i trenutno debagovane Android aplikacije, **Variables** - kada su postavljene, tačke prekida izvršenja programa (*breakpoints*) prikazuje vrednosti promenljivih prilikom izvršenja koda, **Breakpoints** - prikazuje listu postavljenih tačaka prekida programa u samoj aplikaciji [3].

2) **Project Explorer:** Pikazuje sve otvorene Android projekte, tj. foldere u kojima se nalaze java klase, xml fajlovi, manifest itd.
3) **Console/LogCat:** Obe opcije se nalaze pod softverskim alatom DDMS (*Dalvik Debug Monitor Server*) koji omogućuje uvid u procesorke aktivnosti prilikom pokretanja aplikacije u realnom vremenu. Konkretno, dve bitne opcije za ovu tezu su bile *Console* i *LogCat*. Kod opcije *Console* prilikom pokretanja aplikacije mogu se videti bitne informacije kao što su stanja povezivanja uređaja sa virtuelnom ili realnom mašinom, proces instalacije same aplikacije itd. *LogCa*t opcija omogućuje uvid u sistemske poruke u realnom vremenu prilikom pokretanja aplikacije.Takođe je korisna činjenica da se pomoću komande *System.out.println(data)* koja se piše u java klasama može ispisati sadržaj promenljive (u ovom slučaju *data*) u *LogCat* prozor [3].

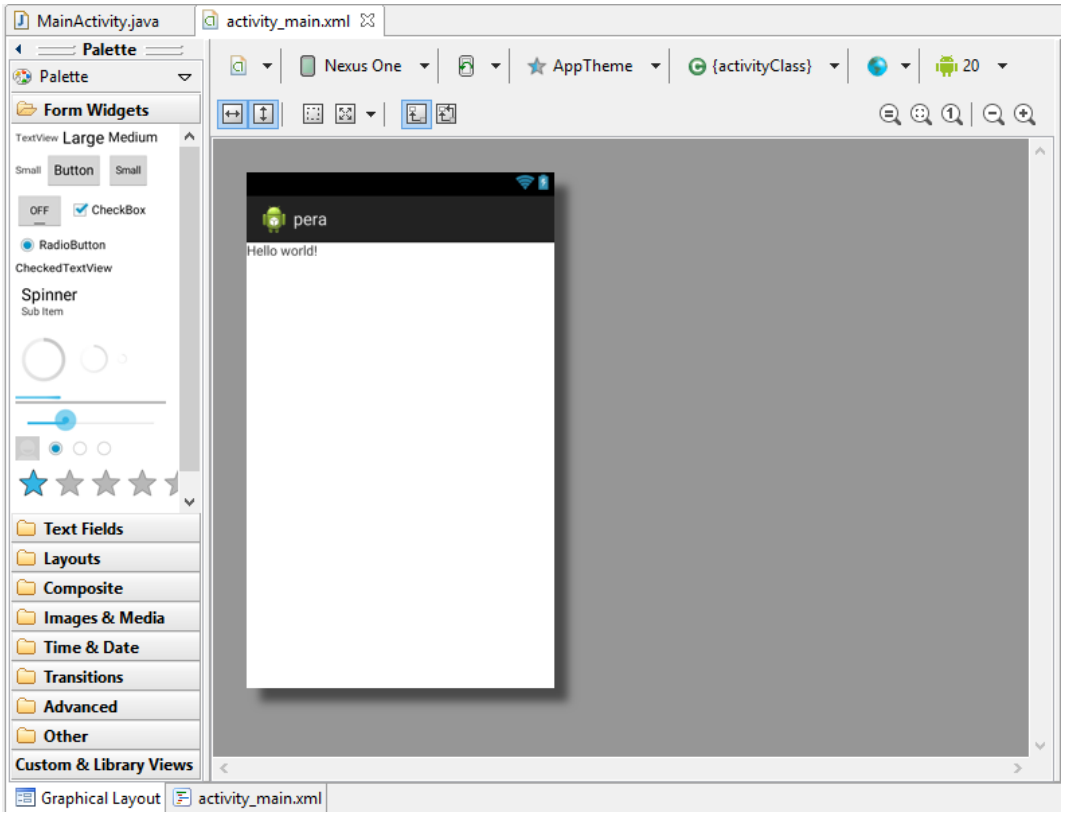
Ono što se razlikuje prilikom programiranja u android okruženju u odnosu na okruženja poput softverskog alata *C*, tj. *C++* jeste raspodela koda na java klase i xml fajlove. U java klasama se programira ono što aplikacija radi na trenutnom prozoru odnosno u androidu se to zove *Activity*. Tu se definišu šta tačno koji od objekata koje smo definisali u xml fajlu radi u zavisnosti koja je interakcija sa njim (dodir, višestruki dodir, prevlačenje itd.). Dakle, java klasa strogo definiše šta se sve dešava na ekranu prilikom interakcije. Takođe treba napomenuti da svaki novi ekran u aplikaciji predstavlja novu java klasu, odnosno novi *Activity* sa odgovarajućim xml fajlom. Xml fajlovi predstavljaju grafički izgled svakog događaja i bazirani su na HTML (*HyperText Markup Language*) sintaksi uz određena prilagođavanja Javi. Svaki xml fajl ima grafički prozor koji u zavisnosti od izabranog modela prikaza, *Eclipse IDE* pokušava da emulira kako će događaj izgledati na ekranu korisnika. Naravno, u velikom broju slučajeva će biti manjeg odstupanja, negde i većeg od realne situacije, ali i dalje programer dobija uvid kako će to otprilike izgledati. Na grafičkom delu xml fajla raspored i oblikovanje objekata se vrši sistemom *Drag and Drop*, tj. objekti se biraju iz padajućeg menija i prevlače na grafički interfejs na kome se može vršiti modifikacija istih. Svaka radnja sa grafičkog dela xml fajla se koduje u programski deo, tj. svaka radnja na grafičkom delu je pretvorena u kod koji posle kompajler čita i prikazuje. Bitno je napomenuti da se u programskom delu xml fajla dodeljuju imena objektima pomoću kojih im se pristupa iz java klasa.

U nastavku teze biće objašnjeni delovi početnog ili tzv. *Hello* programa kako bi čitalac mogao lakše da razume koncept programiranja sa događajima. Sledeći kod predstavlja java klasu početnog programa i koja se nalazi u *src->com.example.imeprograma* folderu:

```
package com.example.pera;
/*importovanje funkcija neophodnih za rad aplikacije */
import android.app.Activity;
import android.os.Bundle;
/*definisanje klase MainActivity koja je proširenje klase Activity. To znači da
prilikom povezivanja više događaja, klase koje se nalaze u njima moraju da budu
proširenje glavne klase Activity.
 */
public class MainActivity extends Activity {
/* Override predstavlja Java Annotation koji govori kompajleru da metoda ima za
cilj da premosti metodu iz superklase.To nije striktno neophodno ali pomaže da
se nađe greska prilikom kompajliranja tako što proverava da li se data metoda
uspešno kompajlirala, ako nije šalje gresku. U principu predstavlja zaštitni
mehanizam za lakše pronalaženje grešaka. */
      @Override
/*Ukoliko se stanje aplikacije čuva u klasi Bundle , može se proslediti funkciji
OnCreate,ukoliko postoji potreba da se rekreira stanje aplikacije tako da se ne
izgube početni podaci. Pri prvom pokretanju savedInstanceState je null a nakon
rekreiranja je nenulta vrednost. Tipičan primer je promena orijerntacije ekrana.
 */
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            /*Povezuje klasu MainActivity sa xml fajlom activity_main*/
            setContentView(R.layout.activity_main);
      }
}
```

Xml fajl main_activity koji se nalazi u folderu *res->layout*:

Grafički prikaz *Hello* programa:



Slika 2.2.4. Grafički prikaz Hello programa

Programski kod:

```
<RelativeLayoutxmlns: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"
tools:context="${relativePackage}.${activityClass}">

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

</RelativeLayout>
```

Funkcije xmlns:android, tools i context su samogenerišuće funkcije koje povezuju sadržaj, alate i izgled sa pomoćnim bibliotekama android projekta. Ostale funkcije su krajnje logične: *heigh t*predstavlja visinu objekta, *width* širinu itd. U ovom primeru predstavljen je relativni raspored koji pozicionira elemente jedne u odnosu na druge, a postoje takođe i linearni, mrežni itd. Linearni raspored se najčešće koristi kad se pravi struktura gde se svaki objekat nalazi u novom redu i postoji samo jedan objekat po redu. Relativni se koristi za malo komplikovanije raspodele objekata i nepravilnim rasporedom može se lako uzrokovati pucanje aplikacije. Takođe treba napomenuti da se prilikom dodele teksta tekstualnom polju u funkciji android: text, može raditi na dva načina. Prvi način je da se u xml fajlu strings.xml može napraviti string koji se kasnije u activity_main.xml u funkciji android:text može zvati kao "@string/ime_stringa". Drugi način predstavlja direktno upisivanje teksta u samu funkciju npr. android:text="Hello World", to se zove *hardcoded string*, odnosno direktno kodovan tekst. Java preporučuje prvu varijantu, ali ne pravi problem ukoliko se koristi druga koja je znatno brža.

Na kraju treba napomenuti da sem softverskog alata *Eclipse IDE* postoji još jedan besplatan softver za razvoj android aplikacija koji se naziva Android Studio. Mana je što je još uvek u razvoju i što je napisan mali broj uputstava za njega, ali kad bude gotov biće glavno sredstvo za razvoj Android aplikacija. Neke od pogodnosti koje nudi ovaj softverski alat:

- Fleksibilan *Gradle* baziran sistem
- Podrška za *Google Cloud Platform*
- *ProGuard* i opcije za logovanje na aplikaciju
- *Lint tools* za rad sa preformansama, kompatibilnost sa verzijama itd.
- Bogat *Layout editor* i mogućnost podešavanja tema

Android Studio se može preuzeti sa https://developer.android.com/sdk/installing/studio.html [3].

3. OPIS IMPLEMENTACIJE

U ovom poglavlju će biti definisana struktura koda, kao i šta sam kod radi. Zbog same količine koda (a i činjenice da značajan deo koda predstavlja male varijacije jednog dela koda) neće biti prezentovan sav kod već samo oni delovi kojipredstavljaju svaku oblast. Kompletan kod će biti priložen u elektronskom formatu. Celokupna aplikacija je podeljena na tri dela: proračun R ocene po E modelu ITU-T G.107 preporuke, crtanje dvostrukog i višestrukog grafika sa R parametrom na y osi i izabranim parametrom na x osi, crtanje dvostrukog i višestrukog grafika sa Ppl parametrom na y osi i izabranim parametrom na x osi.

3.1. Proračun R ocene

Kao što je rečeno u uvodu same teze, prvi deo aplikacije se bavi proračunom R ocene kvaliteta veze po E modelu G.107 preporuke. Sama preporuka ima matematički algoritam koji na osnovnu unetih subjektivnih i fizičkih parametara vrši proračun. Celokupan algoritam kao i objašnenje se mogu nađu na ITU-T sajtu na linku: https://www.itu.int/rec/T-REC-G.107. Da bi se shvatila sama struktura koda za početak će biti objašnjen kod sa početnog ekrana, odnosno događaja gde korisnik bira koju će od 3 celine koda izabrati. U nastavku će biti prezentovana klasa Izbor.java i activity_izbor.xml koji predstavlja grafički izgled početnog događaja.

Izbor.java:

```
package com.example.r_calculator;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.widget.Button;
import android.view.View;
public class Izbor extends Activity {
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_izbor);
/* Definišemo 4 promenljive tipa button koje pomoću funkcije findViewbyId
povezujemo sa dugmićima na osnovnom ekranu, odnosno grafičkom prikazu. */
            Button b1=(Button)findViewById(R.id.button1);
            Button b2=(Button)findViewById(R.id.button2);
            Button b3=(Button)findViewById(R.id.button3);
            Button b4=(Button)findViewById(R.id.button4);
/* pomoću funkcije setOnClickListener definišemmo event handler tako što
napravimo novi objekat View.OnClickListener sa kojim ćemo povezati dugme.*/
            b1.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View v) {
/*Ovde definišemo šta dugme radi kada se na njega klikne. U našem slučaju dugme
treba samo da nas preumseri na drugi događaj odnosno na proacun_parametri.class,
i u tom slučaju koristimo tip objekta intent. U njemu definišemo početni događaj
i događaj u koji želimo da idemo. Nakon toga startujemo događaj sa StartActivity
od tog objekta kog smo definisali. Na taj način klikom na dugme prelazimo na
sledeći ekran. */
```

```
Intent myintent2 = new Intent(Izbor.this,proracun_parametri.class);
startActivity(myintent2);}
            });
/*Analogno važi za ostale dugmiće */
b2.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
Intent myintent2 = new Intent(Izbor.this,Grafik_R.class);
                        startActivity(myintent2);
                    }
            });
b3.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
Intent myintent2 = new Intent(Izbor.this,Grafik_2.class);
                        startActivity(myintent2);
                    }
            });
b4.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
Intent myintent2 = new Intent(Izbor.this,O_programu.class);
                        startActivity(myintent2);

                    }
            });
      }
}
```

activity_izbor.xml:

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/linearLayout1"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="@drawable/repeat"
android:orientation="vertical">
<TextView
android:id="@+id/textView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_above="@+id/textView2"
android:layout_centerHorizontal="true"
android:text="Student:"
android:textAppearance="?android:attr/textAppearanceSmall"/>
<TextView
android:id="@+id/textView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android:layout_centerHorizontal="true"
android:layout_marginBottom="1dp"
android:text="Miloš Pajić 316/2010"
android:textAppearance="?android:attr/textAppearanceSmall"/>
<ImageView
android:id="@+id/imageView1"
android:layout_width="75dp"
android:layout_height="75dp"
android:layout_above="@+id/textView1"
android:layout_centerHorizontal="true"
android:src="@drawable/etf"/>
<Button
```

```
android:id="@+id/button3"
android:layout_width="250dp"
android:layout_height="wrap_content"
android:layout_below="@+id/button2"
android:layout_centerHorizontal="true"
android:layout_marginTop="2dp"
android:text="Grafički prikaz sa Ppl parametrom"/>
<TextView
android:id="@+id/textView3"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignBottom="@+id/imageView2"
android:layout_alignLeft="@+id/imageView1"
android:layout_alignRight="@+id/button1"
android:gravity="center"
android:text="Calculator"
android:textAppearance="?android:attr/textAppearanceLarge"
android:textSize="32sp"/>
<ImageView
android:id="@+id/imageView2"
android:layout_width="60dp"
android:layout_height="60dp"
android:layout_alignParentTop="true"
android:layout_marginTop="10dp"
android:layout_toLeftOf="@+id/textView3"
android:src="@drawable/logo1"/>
<Button
android:id="@+id/button4"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/button3"
android:layout_alignRight="@+id/button3"
android:layout_below="@+id/button3"
android:layout_marginTop="2dp"
android:text="O programu"/>
<Button
android:id="@+id/button2"
android:layout_width="250dp"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/button3"
android:layout_below="@+id/button1"
android:layout_marginTop="2dp"
android:text="Grafički prikaz sa R parametrom"/>
<Button
android:id="@+id/button1"
android:layout_width="250dp"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/button2"
android:layout_below="@+id/textView3"
android:layout_marginTop="30dp"
android:text="Proračun R ocene"/>
</RelativeLayout>
```

Xml fajlovi i njihovi kodovi su obično dosta dugački tako da će se u nastavku teze prikazivati samo zanimljivi nestandardni delovi xml fajlova. Konkretno u ovom fajlu možemo da povežemo funkciju android:id koja daje ime samom objektu i sa tim imenom možemo mu pristupati u java klasama tog događaja. Ovde smo sem objekta dugme, koristili i objekat slika i objekat tekst. Raspodela je rađena u relativnom rasporedu.

Izgled izbor.java u grafičkom prikazu:

Slika 3.1.1. Grafički prikaz izbor.java klase

Nakon odabira dugmeta pod nazivom Proračun R ocene, prelazimo na sledeći događaj proracun_parametri.java. U ovoj java klasi smeštamo podatke koje ćemo kasnije prosleđivati algoritmu za izračunavanje R ocene.

proracun_parametri.java:

```
package com.example.r_calculator;
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.WindowManager;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
public class proracun_parametri extends Activity {
//definišemo privatne objekte tipa dugme i spinner(padajući meni)
private Button b1;
private Spinner spinner1;
@Override
protected void onCreate(Bundle savedInstanceState) {
```

```
/*Ova funkcija je ubačena zbog estetskih razloga, naime kada se prelazi iz
događaja u događaj,prvo polje koje se može menjati postaje zumirano i može
smetati korisniku.Zato je uključena funkcija soft_input_state koja je uvek
sakrivena tako da ne dolazi do automatskog zumiranja promenljivih polja */
getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_
HIDDEN);
super.onCreate(savedInstanceState);
setContentView(R.layout.proracun_parametri);
addListenerOnButton();
addItemsOnSpinner1();
      }
//Pravimo funkciju koja će dodati elemente u padajući meni, i to radimo preko
lsite
private void addItemsOnSpinner1() {
//povežemo promenljivu spinner1 sa padajućim menijem sa grafičkog prikaza
spinner1 = (Spinner) findViewById(R.id.spinner1);
//zatim napravimo listu elemenata koju ćemo dodeliti padajućem meniju
            List<String>list = new ArrayList<String>();
            list.add("G.107");
            list.add("G.711");
            list.add("G.726");
            list.add("G.728");
            list.add("G.729");
            list.add("G.723.1");
            list.add("GSM 06.10");
            list.add("GSM 06.20");
            list.add("GSM 06.30");
            list.add("Novi koder");
//zatim pravimo adapter na padajući meni koji će se ponašati kao spona između
//liste i padajućeg menija
            ArrayAdapter<String>dataAdapter = new ArrayAdapter<String>(this,
                        android.R.layout.simple_spinner_item, list);
dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_ite
m);
//na kraju dodelimo elemente liste padajućem meniju preko adaptera
spinner1.setAdapter(dataAdapter);
//u ovom delu definišemo šta ce se desiti kada selektujemo jedan od elemenata
padajućeg menija
spinner1.setOnItemSelectedListener(new OnItemSelectedListener() {
@Override
//AdapterView odnosno onaj isti adapter koji je pravio sponu između liste i
padajućeg menija
//svaki element liste ima 4 argumenta po kojima se može definisati
public void onItemSelected(AdapterView<?>arg0, View arg1,intarg2, longarg3) {
/*pristupamo promenljivim tekstualnim poljima kako bi smo upisali vrednosti
kodera u zavisnosti koji koder je izabran.*/
EditText text1 = (EditText) findViewById(R.id.editText20);
EditText text2 = (EditText) findViewById(R.id.editText21);
EditText text3 = (EditText) findViewById(R.id.editText22);
/*pravimo switch funkciju u zavisnosti koji je parametar izabran (prvi je 0,
drugi 1 itd)*/
switch(arg2) {
case 0 :
//ako je izabran prvi parametar, odnosno iz liste to je parametar G.107
//Upiši vrednosti 0,1,0 u polja editText20,21,22 respektivno
text1.setText("0");
text2.setText("1");
text3.setText("0");
//Postavljamo da polja više nisu promenljiva.
```

```
text1.setFocusable(false);
text2.setFocusable(false);
text3.setFocusable(false);
break;
//Linearno izvršavamo za ostala polja
case 1 :
text1.setText("0");
text2.setText("4.3");
text3.setText("0.25");
text1.setFocusable(false);
text2.setFocusable(false);
text3.setFocusable(false);
break;

...

case 9 :
//Koder koji može sam korisnik da definiše
text1.setText("");
text2.setText("");
text3.setText("");
//Polja se u ovom slučaju mogu editovati
text1.setFocusableInTouchMode(true);
text2.setFocusableInTouchMode(true);
text3.setFocusableInTouchMode(true);
}
}
//Ukoliko ništa nismo selektovali iz padajućeg menija, ne treba raditi ništa
@Override
public void onNothingSelected(AdapterView<?>arg0) {}
});
}
//U ovom delu funkcije postavljamo listener na dugme , odnosno definišemo sta
//će se desiti kada se klikne na dugme.
private void addListenerOnButton() {
//povezemo padajući meni i dugme sa promenljivima , spinner1 i b1 respektivno.
spinner1 = (Spinner) findViewById(R.id.spinner1);
b1 = (Button) findViewById(R.id.button1);
b1.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
/*Prilikom klika na dugme trebamo da sačuvamo sve parametre koje smo definisali
iz trenutnog dogadjaja u promenljive tipa string.Selektujemo promenljivo polje
gde se nalazio i-ti parametar. */
EditText editText1 = (EditText) findViewById(R.id.editText1);
//Upišemo ga u i-tu promenljivu tipa string pomoću funkcije getText().toString()
String message1 = editText1.getText().toString();
. . .
EditText editText22 = (EditText) findViewById(R.id.editText22);
String message22 = editText22.getText().toString();
//zatim opet preko intenta prelazimo iz jednog događaja u drugi
Intent intent = new Intent(proracun_parametri.this, Proracun_ocena.class);
/*s tim što ovde imamo i informacije koje želimo da prenesemo u taj drugi
događaj. Te promenljive stavimo u niz stringova myStringArray.*/
String[] myStringArray = new String[]{message1, message2, message3, message4,
message5, message6, message7, message8,message9, message10, message11,essage12,
message13, message14, message15, message16,message20, message21, message22};
//i pomoću funkcije put.Extra nazovemo string proizvoljnim imenom pod kojim ćemo
mu pristupati iz drugogdogađaja.Ovde se takođe može slati više nizova zato je
bitno da svaki ima svoje ime.
```

```
intent.putExtra("strings", myStringArray);
//na kraju startujemo drugi događaj
startActivity(intent);  }
            });
}
}
```

Kod grafičkog prikaza imamo dve novine: *EditText*, *ScrollView* i *Spinner* pa ćemo u fajlu proracun_parametar.xml prikazati samo ove funkcije:

```
<ScrollViewxmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/scrollView1"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="@drawable/repeat"
android:fillViewport="true">
<TableLayout
android:layout_width="match_parent"
android:layout_height="match_parent">
<TableRow
android:id="@+id/tableRow1"
android:layout_width="wrap_content"
android:layout_height="wrap_content">
<EditText
android:id="@+id/editText1"
android:layout_width="0dp"
android:layout_weight="0.2"
android:layout_height="wrap_content"
android:ems="10"
android:inputType="numberSigned|numberDecimal"
android:digits="0123456789-."
android:text="-70"
android:gravity="right">
</EditText>
</TableRow>
<TableRow
android:id="@+id/tableRow19"
android:layout_width="wrap_content"
android:layout_height="wrap_content">
<TextView
android:id="@+id/textView19"
android:layout_width="0dp"
android:layout_weight="0.55"
android:layout_height="wrap_content"
android:text=" Izaberite vaš koder:"
android:textAppearance="?android:attr/textAppearanceMedium"/>
<Spinner
android:id="@+id/spinner1"
android:layout_width="0px"
android:layout_weight="0.45"
android:layout_height="50dp"/>
</TableRow>
</TableLayout>
</ScrollView>
```

Ovde je bitno napomenuti jednu vrlo korisinu funkciju prilikom korišćenja rasporeda u vidu tabele, a to je *android:layout_weight*. Ukoliko u jednom redu postoje dve kolone, ovom funkcijom je moguće odrediti težinski faktor koja kolona zauzima koliko mesta u tom redu. Bitno je samo da zbir

vrednosti ovih funkcija u obe kolone bude 1 zbog preklapanja. *Spinner* predstavlja padajući meni, *EditText* promenljivo tekstualno polje, a *ScrollView* je vrsta rasporeda kada količina objekata ne može da stane na ekran pa je potrebno ubaciti klizač ekrana kako bi se svi objekti videli.

Nakon ovog događaja pritiskom na dugme **Izračunaj** prelazi se na sledeći događaj *Proracun_ocena.java.* Njegov grafički deo ne sadrži novine u odnosu na prethodne, pa se taj deo koda neće prikazati u okviru ovog poglavlja. U ovom delu programa se vrši sama kalkulacija ocene na osnovu zadatih parametara, koje smo preneli preko funkcije *intent* iz prethodnog događaja. Sada te parametre moramo otpakovati i ubaciti u matematički algoritam i ispisati rezultat.

Proracun_ocena.java:

```
package com.example.r_calculator;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.TextView;
public class Proracun_ocena extends Activity {
@Override
public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.proracun_ocena);
/*povezujemo promenljive tipa TextView sa objektima iz grafičkog xml fajla
 proracun_ocena.xml */
            TextView newtext = (TextView) findViewById(R.id.editText1);
            TextView newtext1 = (TextView) findViewById(R.id.editText2);
            TextView poruka = (TextView) findViewById(R.id.textView3);
/*Pravimo objekat tipa intent kome dodeljujemo vrednosti funkcije intent
 prosleđene iz prethodnog događaja */
            Intent intent = getIntent();
/*Pomoću funkcije getStringArrayExtra pristupamo preko objekta tipa intent onom
nizu koji smo prosleđivali iz prošlog događaja. Sada definišemo nove promenljive
kojima ćemo dodavati vrednosti ovog niza. */
            String[] myStrings = intent.getStringArrayExtra("strings");
            double Nc = Double.parseDouble(myStrings[0]);
            double Nfor = Double.parseDouble(myStrings[1]);
            double Ps = Double.parseDouble(myStrings[2]);
            double Pr = Double.parseDouble(myStrings[3]);
            double SLR = Double.parseDouble(myStrings[4]);
            double RLR = Double.parseDouble(myStrings[5]);
            double STMR = Double.parseDouble(myStrings[6]);
            double Dr = Double.parseDouble(myStrings[7]);
            double Ds = Double.parseDouble(myStrings[8]);
            double T1 = Double.parseDouble(myStrings[9]);
            double TELR = Double.parseDouble(myStrings[10]);
            double WEPL = Double.parseDouble(myStrings[11]);
            double qdu = Double.parseDouble(myStrings[12]);
            double Ppl = Double.parseDouble(myStrings[13]);
            double BURST = Double.parseDouble(myStrings[14]);
            double A = Double.parseDouble(myStrings[15]);
            double Ie = Double.parseDouble(myStrings[16]);
            double Bpl = Double.parseDouble(myStrings[17]);
            double kasnjenje = Double.parseDouble(myStrings[18]);
/* Sledi matematički proračun sa definisanim parametrima. Funkcije korenovanja,
 stepenovanja se pozivaju preko Math biblioteke.*/
            double T=T1+kasnjenje;
```

```
            double LSTR=STMR+Dr;

            double Tr=2*T;
            double OLR=SLR+RLR;
            double Nfo=Nfor+RLR;
            double Pre=Pr+10*Math.log10(1+Math.pow(10,((10-LSTR)/10)));
            double Nor=RLR-121+Pre+0.008*Math.pow((Pre-35),2);
            double Nos=Ps-SLR-Ds-100+0.004*Math.pow((Ps-OLR-Ds-14),2);
            double
N0=10*Math.log10(Math.pow(10,(Nc/10))+Math.pow(10,(Nos/10))+Math.pow(10,(Nor/10)
)+Math.pow(10,(Nfo/10)));
            double R0=15-1.5*(SLR+N0);

            double Q=37-15*Math.log10(qdu);
            double G=1.07+0.258*Q+0.0602*Math.pow(Q,2);
/*Treba obratiti paznju na kastovanje pošto su nama svi brojevi tipa double,
ukoliko bi smo podelili 46/30 dobili bi smo broj tipa unsigned. Zato se vrši
kastovanje. */
            double Z=(double)46/30-G/40;
            double Y=(R0-100)/15+(double)46/8.4-G/9;
            double Iq=15*Math.log10(1+Math.pow(10,Y)+Math.pow(10,Z));

            double STMRo=-10*Math.log10(Math.pow(10,(-STMR/10))+Math.exp(-
T/4)*Math.pow(10,(-TELR/10)));
            double Ist=12*Math.pow((1+Math.pow(((STMRo-
13)/6),8)),((double)1/8))-
28*Math.pow((1+Math.pow(((STMRo+1)/19.4),35)),((double)1/35))-
13*Math.pow((1+Math.pow(((STMRo-3)/33),13)),((double)1/13))+29;
            double Xolr=OLR+0.2*(64+N0-RLR);
            double Iolr=20*(Math.pow((1+Math.pow((Xolr/8),8)),((double)1/8))-
Xolr/8);

            double Is=Iolr+Ist+Iq;
            double X,Idd;
            if (T>100){
            X=Math.log10(T/100)/Math.log10(2);
Idd=25*(Math.pow((1+Math.pow(X,6)),((double)1/6))3*Math.pow(1+Math.pow((X/3),6),
((double)1/6))+2);
            }
            else  {Idd=0;}
            double Rle=10.5*(WEPL+7)*Math.pow((Tr+1),(-0.25));
            double Idle=(R0-Rle)/2+Math.sqrt(Math.pow((R0-Rle),2)/4+169);

            double TERV=TELR-40*Math.log10((1+T/10)/(1+T/150))+6*Math.exp(-
0.3*Math.pow(T,2));

            double Re=80+2.5*(TERV-14);
            double Roe=-1.5*(N0-RLR);

            double Idte=(((Roe-Re)/2)+Math.sqrt((Math.pow((Roe-Re),2))/4+100)-
1)*(1-Math.exp(-T));
            double Id=Idte+Idle+Idd;
            double Ieeff=Ie+(95-Ie)*Ppl/(Ppl/BURST+Bpl);
            double R=R0-Is-Id-Ieeff+A;
/*Defnisanje Mos ocene koja ide uz R ocenu a definise subjektivni osećaj
korisnika i kreće se od 1 do 4.5 */
            double Mos;
            if (R<0){Mos=1;}
            if (R>0 &&R<100){Mos=1+0.035*R+R*(R-60)*(100-R)*7*Math.pow(10,-6);}
```

```
            else {Mos=4.5;}



/*Formatiramo ispsis da bude zaokružen na dve decimale za R ocenu i jednu
decimalu za Mos ocenu . */
            String str1=String.format("%.2f%n",Mos);
            String str=String.format("%.1f%n", R);
//Tako definisane vrednosti upisujemo u tekstualana polja
            newtext.setText(str);
            newtext1.setText(str1);
//Bojimo polje za ocenu radi lakšeg vizuelnog shvatanja.
            if(R>100){
                  newtext.setBackgroundColor(Color.GREEN);
                  poruka.setText("Kvalitet veze je: ODLIČAN");
                  ;}
            if(R>90 &&R<100){
                  newtext.setBackgroundColor(Color.GREEN);
                  poruka.setText("Kvalitet veze je: ODLIČAN");
                  ;}
            if(R>80 &&R<90){
                  newtext.setBackgroundColor(Color.GREEN);
                  poruka.setText("Kvalitet veze je: VISOK");
                  ;}
            if(R>70 &&R<80){
                  newtext.setBackgroundColor(Color.YELLOW);
                  poruka.setText("Kvalitet veze je: OSREDNJI");
                  ;}
            if(R>60 &&R<70){
                  newtext.setBackgroundColor(Color.RED);
                  poruka.setText("Kvalitet veze je: NIZAK");
                  ;}
            if(R>50 &&R<60){
                  newtext.setBackgroundColor(Color.RED);
                  poruka.setText("Kvalitet veze : NIJE ZADOVOLJAVAJUĆI");
                  ;}

            if(50>R){
                  newtext.setBackgroundColor(Color.RED);
                  poruka.setText("Kvalitet veze : NIJE PREPORUCLJIV");
                  ;}
      }


}
```

Ovim je završen prvi deo deo aplikacije, odnosno proračun R ocene po E modelu ITU-T G.107 preporuke koja definiše matematički model za proračun. Takođe treba napomenuti da je prilikom unosa parametara ostavljena opcija da korisnik može sam da definiše svoj koder ili da izabere jedan iz padajućeg menija u kome su predefinisani koderi koji su najčešće u upotrebi (G.107, G.711, G.726, G.728, G.729, G.723.1, GSM 06.10. GSM 06.20, GSM 06.30).

3.2. Crtanje jednostrukog i višestrukog grafika sa R parametrom na y osi

U ovom delu aplikacije vrši se crtanje jednostrukog i višestrukog grafika po E modelu, s tim što je R ocena uvek na y osi, a izabran parametar na x osi. Kod jednostrukog grafika pri izboru parametara postoji još jedan padajući meni u kome se bira koji od parametara će se crtati po x osi. Izborom nekog od parametara, taj parametar se automatski briše iz liste ostalih parametara koji su skalari, jer ovom parametru treba odrediti opseg crtanja. Za skoro sve parametre postoji predefinisan opseg za koji važi matematički proračun. Naravno, korisnik uvek može sam definisati opseg. Grafički prikaz i java klase za prosleđivanje parametara su veoma slične tako da neće biti prikazane u okviru poglavlja. Stvar koju treba napomenuti jeste da se uvodi nova biblioteka za crtanje *GraphView*, koja se dodaje tako što se .jar fajl same biblioteke ubaci u libs folder samog projekta i onda implementira pomoću opcije *BuildPath*. Da bi sama biblioteka radila mora da se napravi posebna klasa koja implementira *GraphViewDataInterface* zato što funkcija za crtanje *GraphViewSeries* zahteva niz objekata tipa *GraphViewDataInterface*, tj. objekata bilo koje klase koja implementira objekte ovog tipa [5]. Mi smo našu klasu nazvali *GraphViewData* i ona izgleda ovako:

```
package com.example.r_calculator;
import com.jjoe64.graphview.GraphViewDataInterface;
public class GraphViewData implements GraphViewDataInterface {
private doublex,y;
public GraphViewData(doublex, doubley) {
this.x = x;
this.y = y;
  }
@Override
public double getX() {
return this.x;
  }
@Override
public double getY() {
return this.y;
  }
      }
```

Nakon toga biće objašnjen sam algoritam za crtanje jednostrukog grafika, koji je napravljen tako da su svi parametri koji se mogu izabrati za crtanje inicijalizovani kao nizovi čija je vrednost elemenata jednaka, i to jednaka onoj vrednosti koju korisnik definiše kao parametar. Kada korisnik izabere parametar koji će se nalaziti na x osi, u taj parametar (koji je već niz) se umesto onih inicijalnih vrednosti, sada prepisuje 101 tačka opsega (izabrano je 101 radi optimalnog crtanja i rezolucije grafika). To znači da ukoliko je opseg bio npr. od 0 do 500 tačke koje će ići u niz su 0,5,10,15 itd. sve do 500, i ukupno će ih biti 101. Dakle, kao što smo napomenuli proces definisanja vrednosti parametra je vrlo sličan prvom delu uz dodatak još jednog padajućeg menija za izbor parametra po x osi.

Grafik_r_jednostruki.java :

```
package com.example.r_calculator;
import java.util.Arrays;
import java.util.HashMap;
```

```
import java.util.Map;
import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.GraphViewSeries;
import com.jjoe64.graphview.LineGraphView;
import com.jjoe64.graphview.GraphViewSeries.GraphViewSeriesStyle;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.LinearLayout;
publicclass Grafik_r_jednostruki_grafik extends Activity {
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_grafik_r_jednostruki_grafik);
//Uzimanje vrednosti iz intent objekta
		Intent intent = getIntent();
		String[] myStrings = intent.getStringArrayExtra("strings");
		String[] myStrings1 = intent.getStringArrayExtra("strings1");
//Smeštanje parametara u promenljive i definisanje promenljivih (nizova)

		double pocetna=Double.parseDouble(myStrings1[0]);
		double krajnja=Double.parseDouble(myStrings1[1]);
//Niz koji će se crtati po y osi
		double[] Ry=new double[101];
//rezolucija tačaka
		double acc=(krajnja-pocetna)/100;

		String parametar=myStrings[0];
		double[] Nc = new double[101];
	...
		double[] kasnjenje = new double[101];
//Incijalizovanje nizova vrednostima parametra
		for(inti = 0; i<= 100; i++){
			Nc[i] = Double.parseDouble(myStrings[1]);
		...
			kasnjenje[i] = Double.parseDouble(myStrings[19]);
		}
/*Mapiranje parametra tako da bi smo znali koji je parametar izabran. Pošto ne
možemo da poredimo string sa imenom niza, uvodi se mapiranje gde se parametar
koji je izabran mapira sa ključem 1 i pri kraju koda se na x osu crta onaj
parametar (niz) koji je pod ključem broj 1. Nakon toga se mapa briše za sledeći
ciklus.*/
		Map<Integer, double []>mapa = new HashMap<Integer, double []>();
		if(parametar.equals("T")){
			mapa.put(1,T);
			for (intb=0; b<=100; b++){
				T[b]=pocetna +b * acc;
			}
		}
		...
		if(parametar.equals("kasnjenje")){
			mapa.put(1,kasnjenje);
			for (intb=0; b<=100; b++){
				kasnjenje[b]=pocetna +b * acc;
			}
		}
/*za svaki član niza vrsi se proracun R ocene, tj svi parametri će biti
konstantni, osim onog parametra koji je izabran da bude na x osi. Nakon što se
```

```
cela funkcija provrti 100 puta kroz for petlju, imaćemo niz R koji će ići na y
osu. */
for(intb = 0;b<= 100; b++){
double LSTR=STMR[b]+Dr[b];
T[b]=T[b]+kasnjenje[b];
double Tr=2*T[b];
double OLR=SLR[b]+RLR[b];
double Nfo=Nfor[b]+RLR[b];
double Pre=Pr[b]+10*Math.log10(1+Math.pow(10,((10-LSTR)/10)));
double Nor=RLR[b]-121+Pre+0.008*Math.pow((Pre-35),2);
double Nos=Ps[b]-SLR[b]-Ds[b]-100+0.004*Math.pow((Ps[b]-OLR-Ds[b]-14),2);
double
N0=10*Math.log10(Math.pow(10,(Nc[b]/10))+Math.pow(10,(Nos/10))+Math.pow(10,(Nor/
10))+Math.pow(10,(Nfo/10)));

double R0=15-1.5*(SLR[b]+N0);
double Q=37-15*Math.log10(qdu[b]);
double G=1.07+0.258*Q+0.0602*Math.pow(Q,2);
double Z=(double)46/30-G/40;
double Y=(R0-100)/15+(double)46/8.4-G/9;
double Iq=15*Math.log10(1+Math.pow(10,Y)+Math.pow(10,Z));

double STMRo=-10*Math.log10(Math.pow(10,(-
STMR[b]/10))+Math.exp(T[b]/4)*Math.pow(10,(-TELR[b]/10)));

double Ist=12*Math.pow((1+Math.pow(((STMRo-13)/6),8)),((double)1/8))-
28*Math.pow((1+Math.pow(((STMRo+1)/19.4),35)),((double)1/35))-
13*Math.pow((1+Math.pow(((STMRo-3)/33),13)),((double)1/13))+29;

double Xolr=OLR+0.2*(64+N0-RLR[b]);
double Iolr=20*(Math.pow((1+Math.pow((Xolr/8),8)),((double)1/8))-Xolr/8);

double Is=Iolr+Ist+Iq;
double X,Idd;

if (T[b]>100){
X=Math.log10(T[b]/100)/Math.log10(2);
Idd=25*(Math.pow((1+Math.pow(X,6)),((double)1/6))3*Math.pow(1+Math.pow((X/3),6),
((double)1/6))+2);}

else  {Idd=0;}
double Rle=10.5*(WEPL[b]+7)*Math.pow((Tr+1),(-0.25));
double Idle=(R0-Rle)/2+Math.sqrt(Math.pow((R0-Rle),2)/4+169);

double TERV=TELR[b]-40*Math.log10((1+T[b]/10)/(1+T[b]/150))+6*Math.exp(-
0.3*Math.pow(T[b],2));

double Re=80+2.5*(TERV-14);
double Roe=-1.5*(N0-RLR[b]);

double Idte=(((Roe-Re)/2)+Math.sqrt((Math.pow((Roe-Re),2))/4+100)-1)*(1-
Math.exp(-T[b]));

double Id=Idte+Idle+Idd;

double Ieeff=Ie[b]+(95-Ie[b])*(Ppl[b]/(Ppl[b]/BURST[b]+Bpl[b]));

double R1=R0-Is-Id-Ieeff+A[b];
```

```
Ry[b]=R1;
            }


int num = 101;
double [] x=new double [101];

//definišemo novi objekat GraphViewData koji ce imati 100 tačaka crtanja.

GraphViewData[] data = new GraphViewData[num];
//u s niz upisujemo onaj parametar koji je u mapi imao ključ sa rednim brojem 1,
// tj onaj parametar koji je izabran.
double [] s = mapa.get(1);
for (inti=0; i<num; i++) {
//izolovan slučaj, ukoliko je parametar bio T, ne crta se po zakašnjenoj osi
//zbog uticaja algoritamskog kašnjenja, već se crta od početka.
//to važi samo za ovaj parametar.
if (Arrays.equals(T, mapa.get(1))){

x[i]=s[i]-kasnjenje[i];
data[i] = new GraphViewData(x[i], Ry[i]);}

//Ukoliko nije parametar T, crta se sa s nizom na x osi i R nizom na y osi

else  data[i] = new GraphViewData(s[i], Ry[i]);
}
//brisanje mape.
mapa.clear();
//definisanje imena grafika i boje kao i podataka za crtanje (graph)
GraphViewSeries graph = new GraphViewSeries("Grafik", new
GraphViewSeriesStyle(Color.RED, 3), data);
//definisanje linijskog grafika objektom graphView
GraphView graphView = new LineGraphView(this, "Grafik");
//dodavanje podataka na grafik
graphView.addSeries(graph);
//definisanje početnog prozora gledanja grafika
graphView.setViewPort(pocetna, krajnja-pocetna);
//definisanje mogućnosti zumiranja metodom multi-touch
graphView.setScalable(true);
//definisanje fonta i izgleda margina za grafik. Ove vrednosti se nalaze u
folderu res->values->dimens.xml
graphView.getGraphViewStyle().setTextSize(getResources().getDimension(R.dimen.pi
e_segment_label_font_size));
//brojhorizontalnih i vertikalnih podeoka.
graphView.getGraphViewStyle().setNumHorizontalLabels(10);
graphView.getGraphViewStyle().setNumVerticalLabels(10);
//boja mreže na grafiku (grid)
graphView.getGraphViewStyle().setGridColor(Color.WHITE);
//povezivanje linearnog rasporeda grafika sa grafičkim prikazom xml fajla koji
je prethodno napravljen.
LinearLayout layout =(LinearLayout) findViewById (R.id.subLayout);
//dodavanje crteža na linearni raspored
layout.addView(graphView);

      }

}
```

Kod višestrukog crtanja postupak je isti samo se uvodi i treći parametar po kome se crta, odnosno dobijaju višestruki grafici. Taj parametar nema opseg već konkretne vrednosti - maksimalno 5 vrednosti je moguće definisati preko odgovarajućih polja. Logika koja se ovde koristi jeste da se svaki parametar incijalizuje kao matrica čiji će broj kolona biti 101 (i to predstavlja broj tačaka koje idu na x osu), a broj vrsta će biti jednak broju izabranih vrednosti trećeg parametra (dakle maksimalno 5). Tako da ukoliko je jedan parametar izabran za x osu, on će u svim vrstama imati vrednosti izabranog opsega, a parametar koji je izabran kao treći parametar će u svakoj vrsti imati određene vrednosti koje su mu dodeljene (npr. ako su vrednosti bile 1, 2 i 3 prva vrsta će biti sve jednice, druga sve dvojke itd.).
Sledi deo koda koji se razlikuje u odnosu na jednostruki grafik:

Grafik_r_visestruki_grafik.java:

```
            Intent intent = getIntent();
            String[] myStrings = intent.getStringArrayExtra("strings");
            String[] myStrings1 = intent.getStringArrayExtra("strings1");
            double pocetna=Double.parseDouble(myStrings1[0]);
            double krajnja=Double.parseDouble(myStrings1[1]);
//treći parametar
            String multiparametar=myStrings1[2];
//broj vrednosti trećeg parametra
            int br=Integer.parseInt(myStrings1[3]);
//vrednosti trećeg parametra
            double vr1=Double.parseDouble(myStrings1[4]);
            double vr2=Double.parseDouble(myStrings1[5]);
            double vr3=Double.parseDouble(myStrings1[6]);
            double vr4=Double.parseDouble(myStrings1[7]);
            double vr5=Double.parseDouble(myStrings1[8]);
//niz u kome se čuvaju vrednosti
            double[] vrednostibr ={vr1,vr2,vr3,vr4,vr5};
            double[][] Ry=new double[br][101];
            double acc=(krajnja-pocetna)/100;
//parametar na x osi
            String parametarx=myStrings[0];
            double[][] Nc = new double[br][101];
            double[][] Nfor = new double[br][101];
            double[][] Ps = new double[br][101];
            double[][] Pr = new double[br][101];
            double[][] SLR = new double[br][101];
            double[][] RLR = new double[br][101];
            double[][] STMR = new double[br][101];
            double[][] Dr = new double[br][101];
            double[][] Ds = new double[br][101];
            double[][] T = new double[br][101];
            double[][] TELR = new double[br][101];
            double[][] WEPL = new double[br][101];
            double[][] qdu = new double[br][101];
            double[][] Ppl = new double[br][101];
            double[][] BURST = new double[br][101];
            double[][] A = new double[br][101];
            double[][] Ie = new double[br][101];
            double[][] Bpl = new double[br][101];
            double[][] kasnjenje = new double[br][101];
//inicijalizacija promenljivih vrednostima parametra
```

```
            for (inti = 0; i<br; i++){
      for (intj = 0; j<= 100; j++){
            Nc[i][j] = Double.parseDouble(myStrings[1]);

      ...
            kasnjenje[i][j] = Double.parseDouble(myStrings[19]);
      }
      }
//pravljenje mape za povezivanje imena parametra sa matricom parametra
      String[]                              Parametri                              =
{"Nc","Nfor","Ps","Pr","SLR","RLR","STMR","Dr","Ds","T","Telr","Wepl","qdu","Ppl
","BurstR","A","Ie","Bpl","kasnjenje"};
      Map<String, double [][]>mapa = new HashMap<String, double [][]>();
      mapa.put("Nc",Nc);
   ...
      mapa.put("kasnjenje",kasnjenje);
//u s smestamo parametar na x osi, u w treći parametar
      double [][] s= new double[br][101];
      double [][] w= new double[br][101];

/*pošto nam se u nizu Parametri nalaze imena svih parametara upoređujemo svaki
element sa onim sto je korisnik izabrao. Kada dodje do podudaranja pomoću imena
parametra u mapi se traži matrica koja je povezana na to ime.Zatim se vrši
popunjavanje tih matrica.*/
      for(inti=0; i<=18; i++){
            if(parametarx.equals(Parametri[i])){
                  s = mapa.get(Parametri[i]);
                  for (intb=0; b<br; b++){
                        for (intc=0; c<=100; c++){
                              s[b][c]=pocetna +c*acc;
                              }
                  }
            }
      }
            for(intj=0; j<=18; j++){
                  if(multiparametar.equals(Parametri[j])){
                        w = mapa.get(Parametri[j]);
                        for (intb=0; b<br; b++){
                              for (intc=0; c<=100; c++){
                              w[b][c]=vrednostibr[b];
                              }
                        }
                  }
            }
```

Ostatak koda definiše matematički algoritam za proračun R ocene koji se razlikuje u tome što parametri više nisu nizovi već matrice. Deo za crtanje ostaje isti s tim što je napravljeno pet slučajeva u zavisnosti koliko je od 5 mogućih vrednosti parametara definisano.

3.3. Crtanje jednostrukog i višestrukog grafika sa Ppl parametrom na y osi

Ovaj deo aplikacije je skoro potpuno identičan sa drugim delom, jedina razlika je što se na y osi nalazi Ppl parametar umesto R parametra. Takođe, zbog neretkih dešavanja da parametar Ppl često iskače iz predefinsanog opsega (od 0 do 20) uvedena je granica, tako da kad god Ppl parametar na grafiku treba da pređe granicu preko 20 ili ispod 0 biće mu dodeljene granične vrednosti. Samim tim što se na y osi nalazi Ppl parametar umesto R jedina promena će biti u matematičkom algoritmu izračunavanja, tj. sad se preko R i ostalih parametara izražava Ppl parametar. U nastavku je prikazan deo koda matematičkog algoritma koji se razlikuje od drugog dela.

```
double Idte=(((Roe-Re)/2)+Math.sqrt((Math.pow((Roe-Re),2))/4+100)-1)*(1-
Math.exp(-T[b]));

double Id=Idte+Idle+Idd;

double Ieff=R0-Is-Id-R1[b]+A[b];

Ppl[b] = (((Ieff-Ie[b])/(95-Ie[b]))*Bpl[b])/(1-((Ieff-Ie[b])/(95-
Ie[b])*(1/BURST[b])));


//uslov za probijanje granice Ppl (od 0 do 20)
GraphViewData[] data = new GraphViewData[num];
double [] s = mapa.get(1);
            for (inti=0; i<num; i++) {
                  if(Ppl[i]>20){Ppl[i]=20;}
                  if(Ppl[i]<0){Ppl[i]=0;}
                  data[i] = new GraphViewData(s[i], Ppl[i]);

            }
```

Ovim je završeno poglavlje o samoj aplikaciji i strukturi koda. U sledećem poglavlju će biti pokazani konkretni primeri rada aplikacije. Čitav kod projekta je elektronski priložen uz tekst teze.

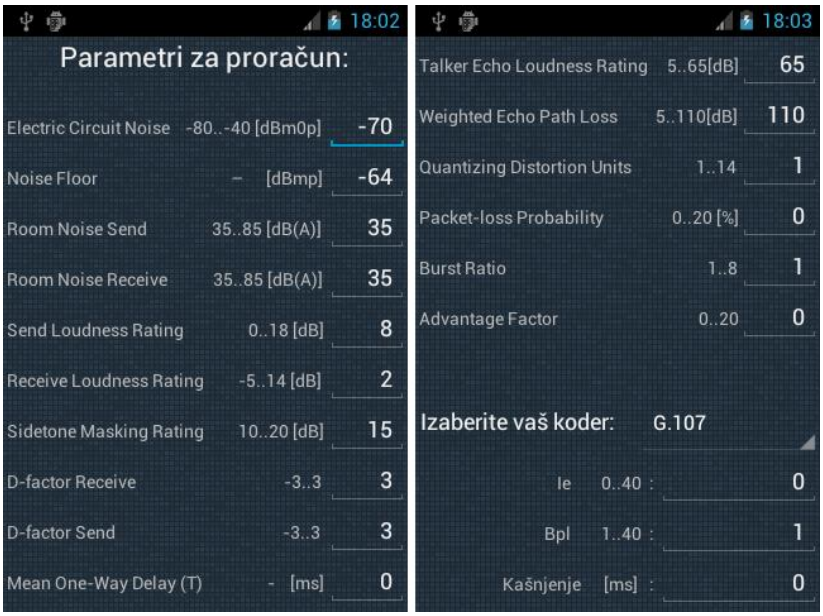
4. UPUTSTVO ZA KORIŠĆENJE APLIKACIJE

U ovom delu teze će biti dati konkretni primeri i objašnjenja za svaki od delova aplikacije. Najpre će biti objašnjeno kako učitati sam projekat koji je elektronski priložen uz tekst teze. Prvo treba raspakovati zip fajl u kome se projekat nalazi. Zatim treba učitati pomoćne biblioteke, a sam proces je detaljno objašnjen na strani 7 drugog poglavlja. Nakon toga se aplikacija može emulirati virtuelnim ili realnim android uređajem što je takođe objašnjeno u drugom poglavlju (desni klik na projekat, zatim treba izabrati opciju *Run As->Application Project* i onda izabrati virtuelno ili realno emuliranje).

Početni ekran je prikazan na slici 4.1. Početni ekran android projekta se sastoji od četiri dugmeta. Prva tri dugmeta vode ka istoimenim delovima ovog projekta, dok četvrto dugme vodi ka uputstvu za korisnike.

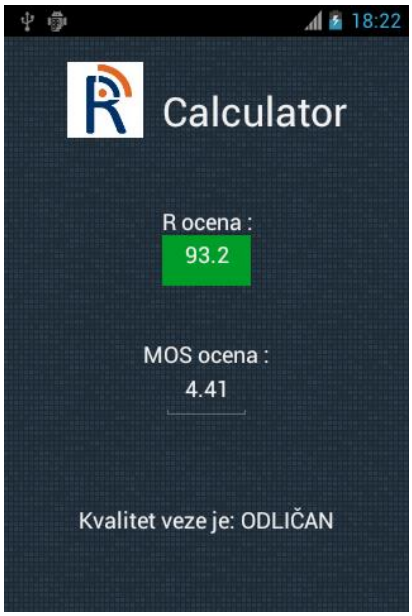
Slika 4.1. Početni ekran Android projekta

Klikom na dugme *Proračun R ocene* dobijemo prikaz na ekranu kao na slici 4.2. Parametri proračuna R ocene. Vrednosti parametara prvog primera su difolt vrednosti sa oficijalnog sajta za proračun R ocene na linku: https://www.itu.int/ITU-T/studygroups/com12/emodelv1/calcul.php [4].



Slika 4.2. Parametri proračuna R ocene

Svaki od parametara je definisan punim imenom, opsegom i jedinicama. Pritiskom na promenljivo polje sa desne strane može se izmeniti trenutna vrednost. Pri dnu ekrana se vrši odabir kodera klikom na padajući meni ili se može definisati novi koder odabirom opcije *Novi koder*. Nakon završetka unosa svih parametara klikom na dugme *Izračunaj* prelazi se na ekran sa R i Mos ocenom prikazan na slici 4.3. Proračun R i Mos ocene.

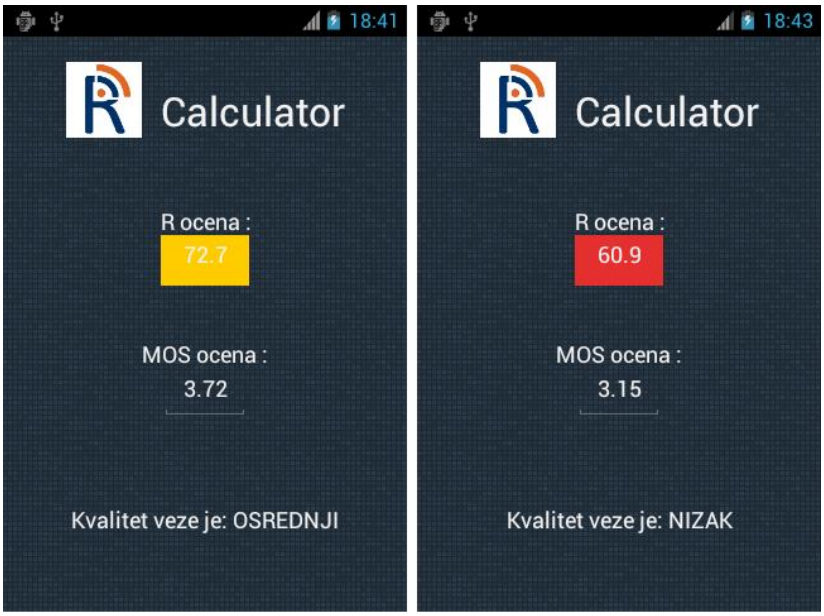


Slika 4.3. Proračun R i Mos ocene

Prikazan rezultat je identičan rezultatu sa oficijalnog sajta aplikacije, čime je verifikovana tačnost algoritma (naravno, ispravan rad aplikacije je verifikovan proračunima i za druge vrednosti parametara). Primer proračuna za druge vrednosti parametara:

1) T=50ms, Ppl=2%, koder G.729 sa parametrima Ie=10, Bpl=18, kašnjenje=25
2) T=250ms, A=10, koder GSM 06.20 sa parametrima Ie=23, Bpl =15, kašnjenje=40

Rezultati su prikazani na slici 4.4. Rezultati proračuna za druge parametre (sa leve strane je prvi slučaj, a sa desne drugi):



Slika 4.4. Rezultati proračuna za druge parametre

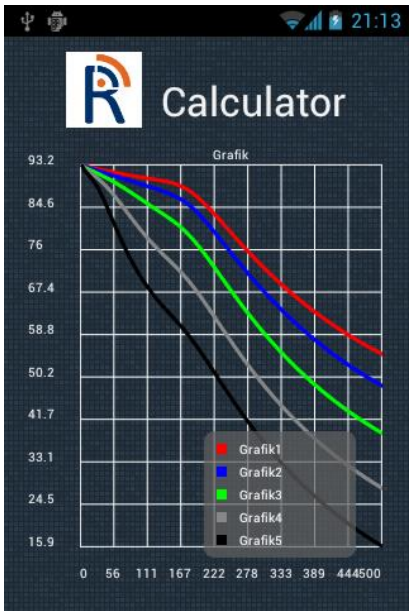
Analogno prethodnim primerima, slede primeri za višestruki grafik sa R ocenom na y osi. Klikom na dugme *Grafički prikaz sa R parametrom* prelazi se na prozor gde se vrši izbor vrste crtanja grafika prikazan na slici 4.5. Izbor vrste grafika.

Slika 4.5. Izbor vrste grafika

Primeri za jednostruke grafike nisu neophodni, jer se dati grafici sadrže u višestrukim, tako da se iz ovih primera može izvući isti zaključak. Definisanje vrednosti parametara je isto kao i na slici 4.2. Parametri proračuna R ocene uz dodatak padajućeg menija za izbor parametra koji se iscrtava na x osi. Klikom na dugme *Dalje* prelazi se na ekran prikazan na slici 4.6. Definisanje parametara višestrukog grafika.

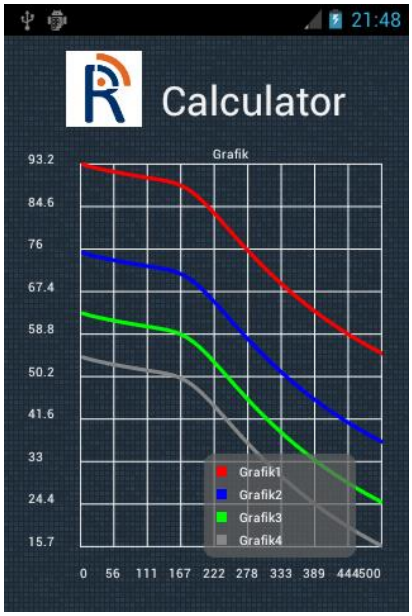
Slika 4.6. Definisanje parametara višestrukog grafika

U polju *Izaberite parametar* se definiše treći parametar za crtanje višestrukog grafika. U polju *Broj vrednosti parametra* definišemo broj vrednosti parametra, zatim se na levoj strani ispod polja *Vrednosti* otvaraju polja za popunjavanje u zavisnosti koliko je vrednosti izabrano. Ispod polja *X osa* se definiše opseg crtanja za parametar na x osi. Klikom na dugme *Nacrtaj* prelazi se na ekran prikazan na slici 4.7. Višestruki grafik sa R parametrom. Višestruki grafik sadrži vreme T kao parametar na x osi koji se kreće u opsegu od 0 do 500ms, a treći parametar je TELR koji ima pet vrednosti: 65, 60, 55, 50, 45[dB]. Ostali parametri su postavljeni na difolt vrednosti iz prvog primera prikazanog na slici 4.2. Parametri proračuna R ocene.



Slika 4.7. Višestruki grafik sa R parametrom

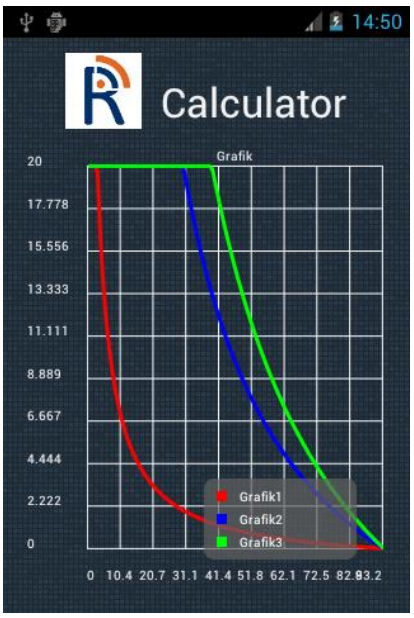
U sledećem primeru prikazanom na slici 4.8. Primer za višestruki grafik sa R parametrom za parametar na x osi uzeto je opet vreme T u opsegu od 0 do 500ms, a treći parametar je Ppl koji ima 4 vrednosti: 0, 1, 2, 3%. Koder je G.711 sa parametrima: Ie=0, Bpl=4.3, kašnjenje=0.25. I ovde su vrednost ostalih parametra postavljene na difolt vrednosti.



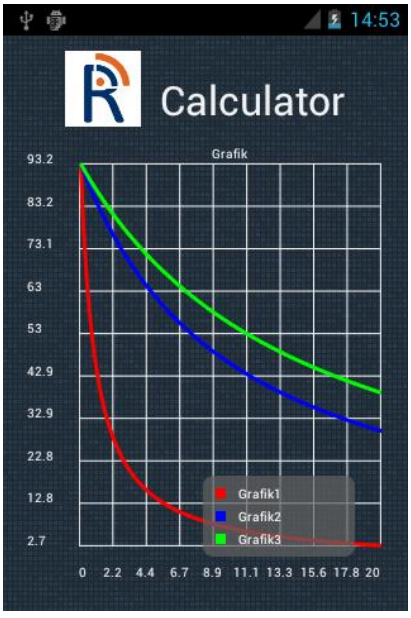
Slika 4.8. Primer za višestruki grafik sa R parametrom

Primeri su poređeni sa graficima iz skripte za predmet Komutacioni sistemi čiji je autor prof. Zoran Čiča na stranama 61 i 62 poglavlja 8 [2].

Za primer kada je parametar Ppl na y osi (klikom na dugme *Grafički prikaz sa Ppl parametrom*, ostatak je analogan definisanju parametara sa R parametrom na y osi) prikazanom na slici 4.9. Pimer za višestruki grafik sa Ppl parametrom uzet je parametar R na x osi u opsegu od 0 do 93.2, a treći parametar Bpl je definisan za 3 vrednosti: 1, 10, 15. Ostali parametri imaju difolt vrednosti sa slike 4.2. Parametri proračuna R ocene. Kao proveru ispravnosti grafika uradili smo inverzan grafik na slici 4.10. Provera višestrukog grafika sa Ppl parametrom u drugom delu aplikacije gde je R na y osi a Ppl na x osi u opsegu od 0 do 20. Vrednosti trećeg parametra Bpl su iste.



Slika 4.9. Pimer za višestruki grafik sa Ppl parametrom



Slika 4.10. Provera višestrukog grafika sa Ppl parametrom

5. ZAKLJUČAK

Sama aplikacija objedinjuje funkcionalnosti proračuna R ocene po ITU-T G.107 preporuci sa mogućstvom crtanja jednostrukih i višestrukih grafika sa R i Ppl parametrom na y osi. Pomoću grafika mogu se utvrditi međusobne zavisnosti nekih od parametara u proračunu ocene kvaliteta telefonskog servisa, na osnovu čega se potencijalno može utvrditi način ublažavanja uticaja negativnih efekata koji dovode do niže ocene kvaliteta.

Prednost realizovane aplikacije je takođe što ne zahteva upotrebu internet konekcije za rad, vrlo je jednostavna za korišćenje, ne zauzima puno prostora u memoriji, i pre svega dostupna je na svim uređajima koji podržavaju Android operativni sistem.

Ono što predstavlja manu ove aplikacije i što bi trebalo doraditi jeste pre svega sistem za crtanje grafika, jer je sistem relativno prost, tj. ne ume da prepozna kada neki parametar ode u nedozvoljene vrednosti i da izvrši zabranu istih. Takođe bi trebalo da se uvede ispitivanje da li je popunjeno svako polje pre prosleđivanja parametara, jer ukoliko nije dolazi do pucanja aplikacije.

Što se tiče grafičkog dizajna aplikacije, postoje određeni opšti problemi vezani za android aplikacije, jer u zavisnosti od verzije Android operativnog sistema, određeni elementi će se prikazivati drugačije, može doći do preklapanja na manjim ekranima itd.

Ono što predstavlja veliku prednost svih android aplikacija jeste distributivnost preko aplikacije Play Store, pomoću koje su aplikacije dostupne svim korisnicima android uređaja.

LITERATURA

[1] James Steel Nelson “The Android Developer’s Cookbook” , 2010

[2] Zoran Čiča “Komutacioni Sistemi”, 2013

[3] http://developer.android.com/index.html

[4] https://www.itu.int/ITU-T/studygroups/com12/emodelv1/index.htm

[5] http://android-graphview.org/