Simbolički kalkulator

Crvenković, Ivan

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij računarstva

SIMBOLIČKI KALKULATOR

Završni rad

Ivan Crvenković

Osijek, 2021.

Sadržaj

1. UVOD ... 1
1.1. Zadatak završnoga rada ... 1
2. SLIČNE APLIKACIJE ... 2
2.1. PhotoMath ... 2
2.2. Financial Calculators ... 3
2.3. Symbolab ... 4
2.4. Windows Calculator ... 5
2.5. Casio fx-991ES PLUS ... 6
3. RESURSI I TEHNOLOGIJE KORIŠTENE ZA IZRADU APLIKACIJE ... 7
3.1. Android Studio ... 7
3.2. Biblioteka ... 9
3.3. Objektno orijentirani jezik Java ... 9
4. RAZVOJ APLIKACIJE ... 11
4.1. Fragmenti ... 12
4.2. Prvi fragment za kalkulator ... 17
4.3. Drugi fragment za formule ... 21
4.4. Treći fragment za tablice i matrice ... 25
5. ZAKLJUČAK ... 29
LITERATURA ... 31
SAŽETAK ... 32
ABSTRACT ... 33

1. UVOD

Ideja završnog rada je pomoći učenicima i studentima kod lakšeg rješavanja matematičkih zadataka i u provjeravanju točnosti istih. Početna zamisao je bila napraviti popularni kalkulator *Casio fx-115es plus* za pametne telefone, te proširiti funkcionalnosti. Današnji kalkulatori ili imaju previše nepotrebnih, nepreglednih funkcionalnosti ili jednostavno nedostaju funkcionalnosti koje bi učenici svakodnevno koristili. Primjerice, kada se radi otplatna tablica kod zajma, običnim kalkulatorom se izgubi najmanje 15 minuta na računanje zbog upisivanja podataka. Unutar završnog rada prikazano je jednostavno rješavanje problema za provjeru rezultata izvan nastave.

U drugom poglavlju završnoga rada prikazuju se slične aplikacije koje postoje za istu problematiku. Nakon toga opisane su tehnologije korištene za izradu aplikacije. Razvoj aplikacije je strukturiran na način da se u prvom potpoglavlju opisuju fragmenti, zatim u drugom slijedi opisan prvi fragment zadužen za računanje matematičkih izraza, u idućem je opisan drugi fragment zadužen za računanje matematičkih formula i na posljetku je opisan treći fragment za računanje determinante matrica i zajam.

1.1. Zadatak završnoga rada

Napraviti aplikaciju za kalkulator koji bi bio u stanju parsirati matematičke formule, računati s razlomcima i računati determinante matrica. Rezultati i međurezultati bi se prikazivali na simbolički način.

2. SLIČNE APLIKACIJE

U današnje doba teško je izumiti ili proizvesti nešto sasvim jedinstveno, tako da postoje slična rješenja za napisanu problematiku. U nastavku, bit će opisane pet aplikacija s preko par milijuna preuzimanja sa *Google Play*[1] trgovine.

2.1. PhotoMath

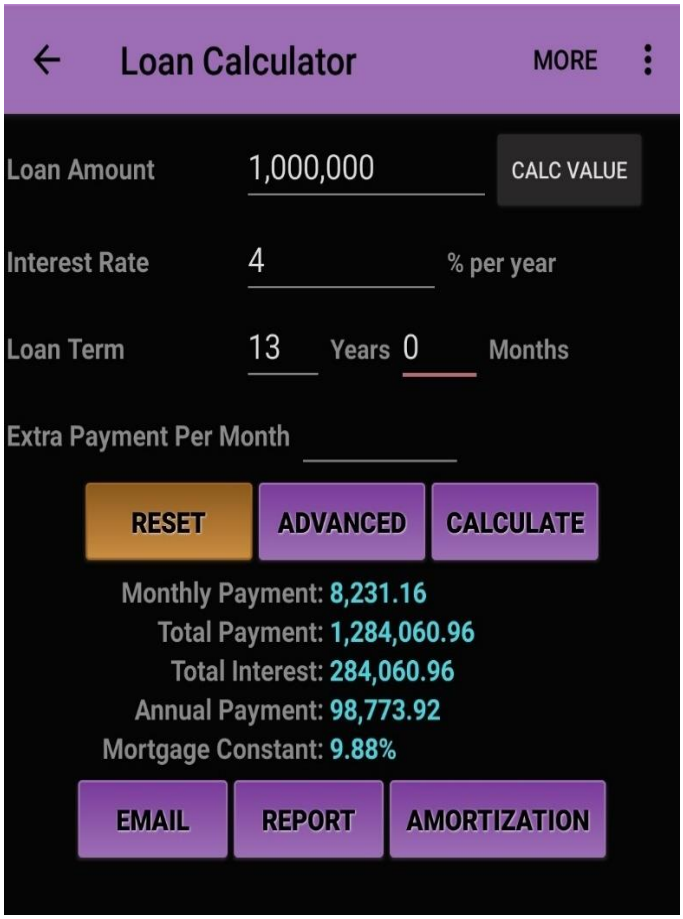
Za primjer može se uzeti aplikacija *PhotoMath* koju je razvio hrvatski startup i aplikacija se probila na svjetsko tržište. Izvrsna je za provjeru rješenja i dobivanje postupka za linearne jednadžbe za osnovnu i srednju školu jer zna biti dosta nezgodno za pronaći grešku u izračunu (slika 2.1).

Slika 2.1 [1] Snimka zaslona *PhotoMath* aplikacije – primjer rješavanja sustava jednadžbi

[1] Google play – dućan za kupovanje mobilnih aplikacija na android sustavima

2.2. Financial Calculators

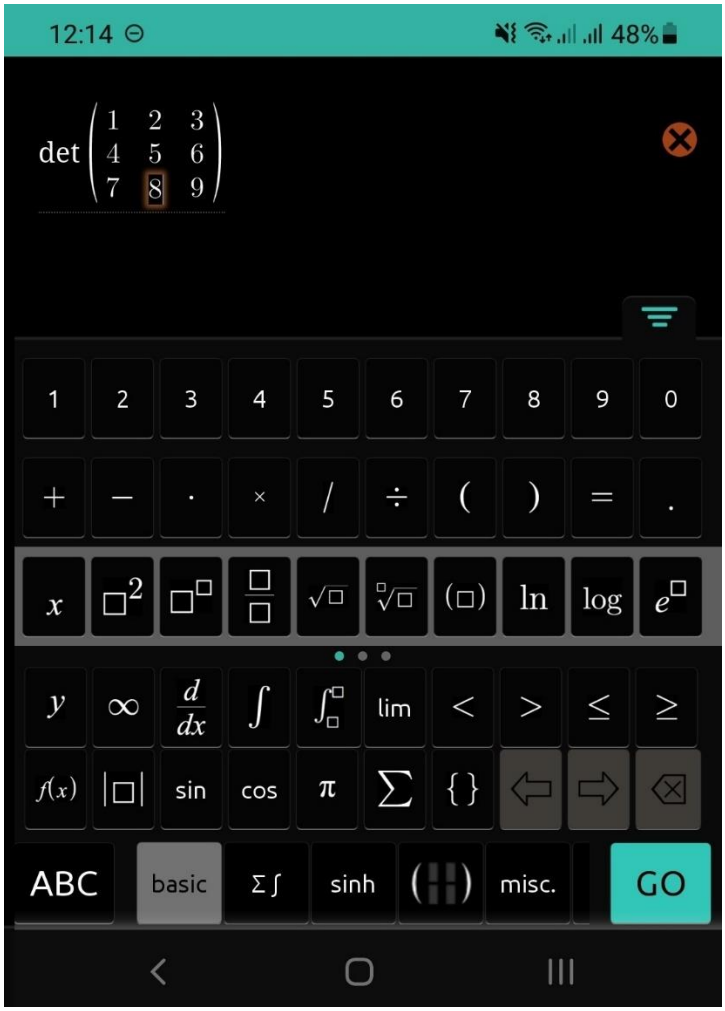
Studenti ekonomije često se susreću sa zadacima u kojima se računaju zajmovi. Poznata android aplikacija vezana za financijsku matematiku *Financial Calculators* računa kamate i daje otplatnu tablicu (slika 2.2). Jednostavnim upisom iznosa kredita, kamatne stope i broja godina otplaćivanja kredita dobiva se stvarni iznos kredita uz uračunate kamate. Pritiskom na Amortization dobiva se tablični prikaz kamata za svako razdoblje.



Slika 2.2 [2] Snimka zaslona *Financial Calculators* aplikacije – primjer računanja kamata

2.3. Symbolab

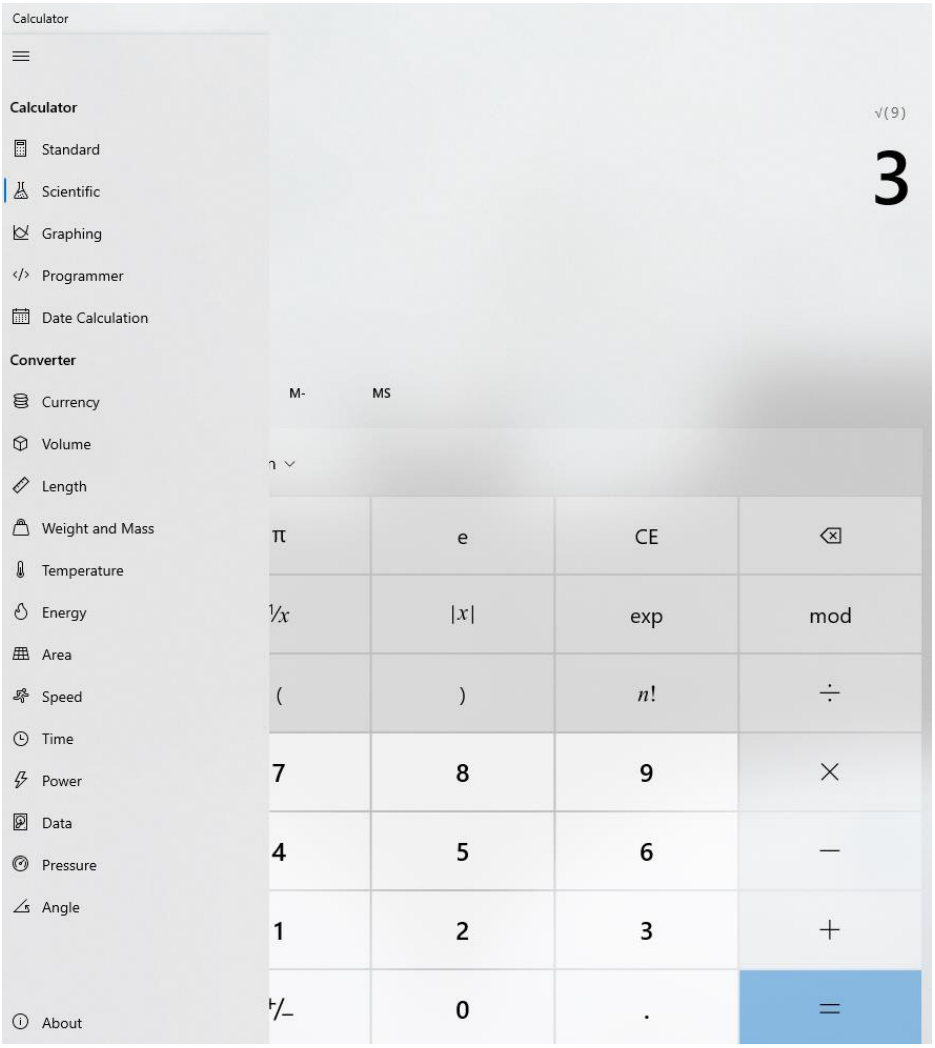
Poznata aplikacija za android uređaje *Symbolab* ima mnogo funkcionalnosti, među njima računa determinantu matrice. Na slici 2.3 primjeti se nedostatak kod upisa članova u predviđena polja. Često je nezgodno odabrati točno polje i prebacivati se s jednog na drugo.



Slika 2.3 [3] Snimka zaslona *Symbolab* aplikacije – primjer upisa determinante matrice

2.4. Windows Calculator

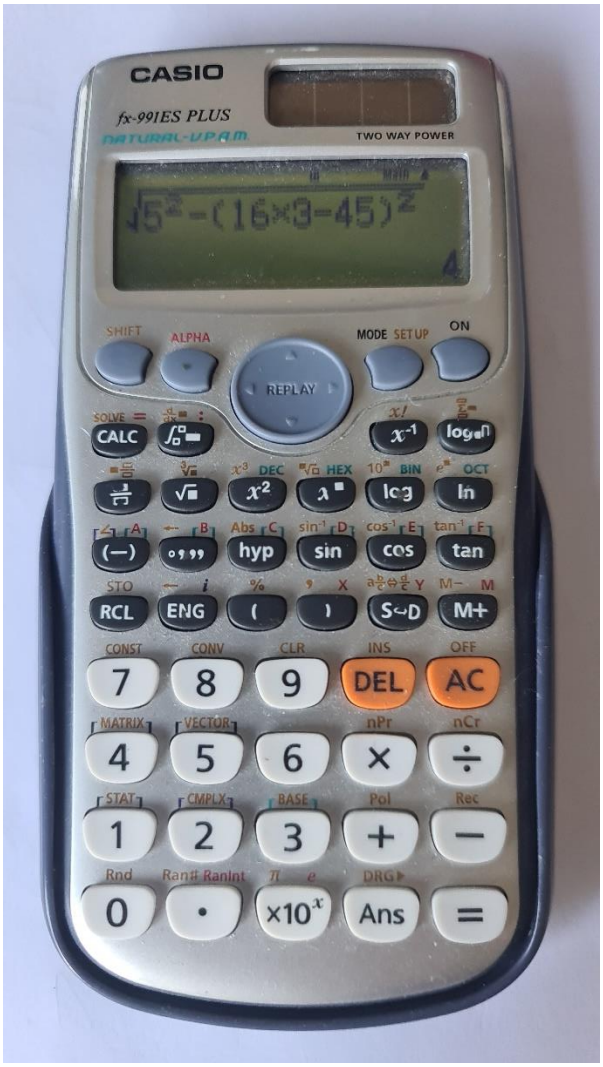
Kalkulator koji je instaliran na Windows računalima i tabletima vrlo često se koristi i u većini slučajeva zadovoljava sve zahtjeve korisnika. Specifično kod Windows kalkulatora je da se prvo upisuju brojevi i onda pritiskom na funkciju ili operaciju dobije se rezultat. Zbog toga je idealan u računavodstvu i financijama. Prikaz rada kalkulatora vidljiv je na slici 2.4.



Slika 2.4 [4] Snimka zaslona *Windows Calculator* – primjer korjenovanja

2.5. Casio fx-991ES PLUS

Najpoznatiji kalkulator kod studenata i učenika je *Casio fx-991ES PLUS* (slika 2.5). Dopušteno je korištenje na maturi te se učenici početkom srednje škole uče koristiti njime. Jednostavan je za korištenje, ima širok raspon funkcija i matematičkih operacija.



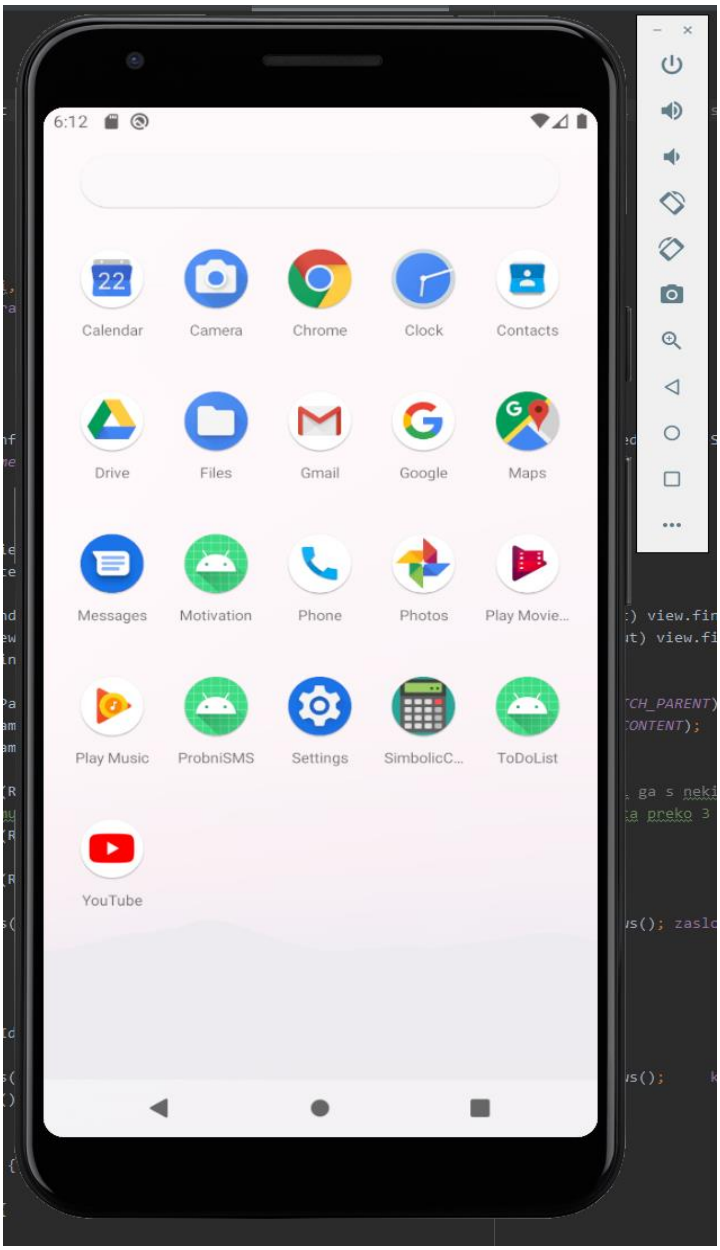
Slika 2.5 [5] *Casio fx-991ES PLUS* – primjer korjenovanja kompleksnog izraza

3. RESURSI I TEHNOLOGIJE KORIŠTENE ZA IZRADU APLIKACIJE

U ovom poglavlju navedeni su alati korišteni za izradu projekta i opisane su tehnologije potrebne za izradu istoga.

3.1. Android Studio

Za razvoj aplikacije korišteno je razvojno okruženje Android Studio 3.4.2. Android Studio[6] je striktno namijenjen izradi Android aplikacija na Android uređajima (pametnim telefonima, tabletima, satovima i drugo). Ima ugrađen uređivač za pisanje koda u Javi ili Kotlinu. Tijekom pisanja koda uređivač nudi prijedloge za postojeće varijable ili imena metoda koje je moguće upotrijebiti što uvelike olakšava pisanje koda programeru. Također, uređivač raznim bojama označava dijelove koda radi lakšeg snalaženja (narančastom bojom označuje ključne riječi, ljubičastom bojom označuje globalne varijable, i drugo). Boje je moguće prilagoditi kako programeru odgovara koristeći postavke unutar uređivača. Također, Android Studio omogućuje testiranje aplikacije na stvarnom android uređaju ili u nedostatku istoga, omogućuje testiranje na virtualnom Android uređaju (slika 3.1).



Slika 3.1 Android emulator – virtualni android pametni telefon

3.2. Biblioteka

Kako bi se moglo koristiti navedeno razvojno okruženje, potrebno je dodatno instalirati SDK pakete koji pružaju alate za izradu aplikacije. Uz naveden skup alata, korištena je dodatna biblioteka *mXparser* za parsiranje[2] matematičkih izraza. Biblioteka ima široku podršku za razne sustave i jezike te se unutar biblioteke nalaze podrške za funkcionalnosti (za parsiranje matematičkih izraza, za rješavanje sustava jednadžbi, za rješavanje linearnih jednadžbi, za deriviranje i integriranje funkcija, i drugo). Za izradu projekta korištena je grana biblioteke za parsiranje matematičkih izraza i dobivanje rezultata toga izraza. Biblioteka *mXparser* [7] ima pojednostavljenu BSD licencu koja dozvoljava korištenje bibilioteke u edukativne svrhe uz davanje kredita autoru biblioteke Mariusz Gromada.

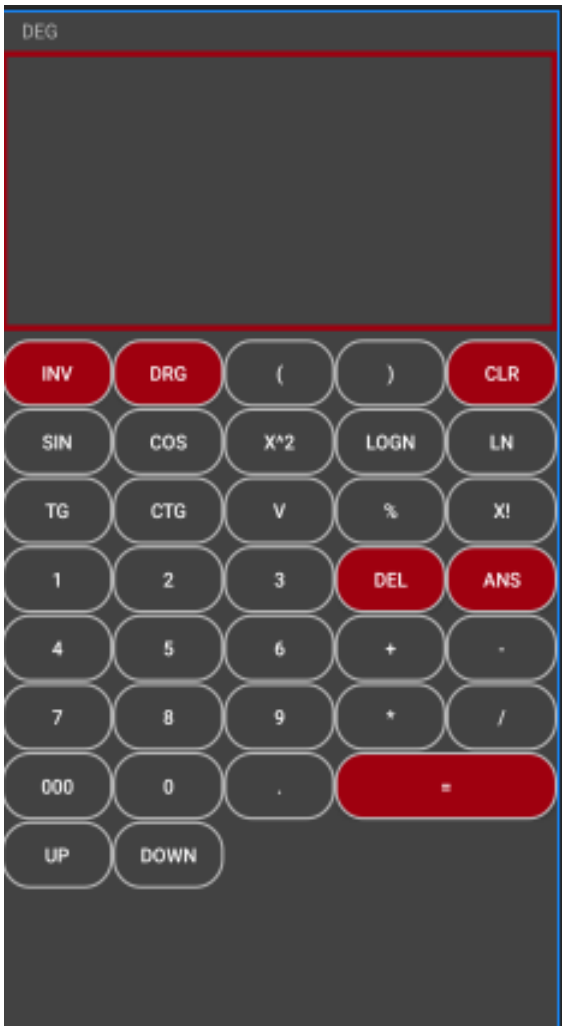
3.3. Objektno orijentirani jezik Java

Za pisanje logike rada aplikacije korišten je objektno orijentirani programski jezik *Java*. Objektno orijentirani jezik razlikuje se naspram običnog programskog jezika što dodatno ima objekt klase kao tip podatka. Klase, osim mogućnost postavljanja atributa, imaju dodatnu mogućnost pisanja metoda i postavljanje početnih parametara pri kreiranju objekata. Za vrijeme izrade završnoga rada postojala je mogućnost pisanja koda u *Kotlinu*, novom programskom jeziku kojemu je glavna funkcionalnost smanjiti kod i olakšava pisanje samom programeru. Svejedno je odabran programski jezik *Java* zbog duljeg postojanja i samim time veće podrške.

Za prezentiranje podataka, odnosno prikaz sučelja aplikacije, korišten je XML, jezik za označavanje podataka. Jezik je nastao u svrhu razdvajanja pisanja logike aplikacije od prezentiranja podataka. XML se sastoji od *widgeta*[3] koji komunicira s objektima tako da se na *widget* postavi referenca i unutar objekta poziva se na odabranu referencu. Najčešće korišteni widgeti su *LinearLayout* (kontejner za ostale widgete), *Button* (widget koji se ponaša kao tipka),

[2] Parsiranje – proces kod prevođenja izvornog programa, kada se prepoznavaju osnovne strukture programskog jezika
[3] Widget – objekt pogleda, komponenta

TextView (widget za prikaz teksta) i *EditText* (widget za upis teksta). Widgeti *Button* i *TextView* prikazani su na slici 3.2.



Slika 3.2 Primjer widgeta Button (tipke) i TextView (ekran kalkulatora)

4. RAZVOJ APLIKACIJE

Za početak treba pokrenuti Android Studio i kreira se novi projekt s praznom aktivnošću. Grane projekta koje se najčešće koriste su *manifest*, *java* i *res* te *gradle*.

U datoteci *AndroidManifest.xml* nalaze se bitne informacije o projektu (ime projekta, teme i drugo) i postavljaju se razne dozvole za korištenje dijelova mobitela (kamera, mikrofon, lokacija i drugo). U datoteci *Gradle Scripts* nalaze se Java klase u obliku paketa i koriste se za dodavanje raznih klasa (*ConstraintLayout*, *RecycleView* i drugo). U dijelu *build.gradle (module:app)* dodaje se biblioteka za korištenje tablayouta i viewpagera (Slika 4.1.).

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'com.android.support:design:28.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
}
```

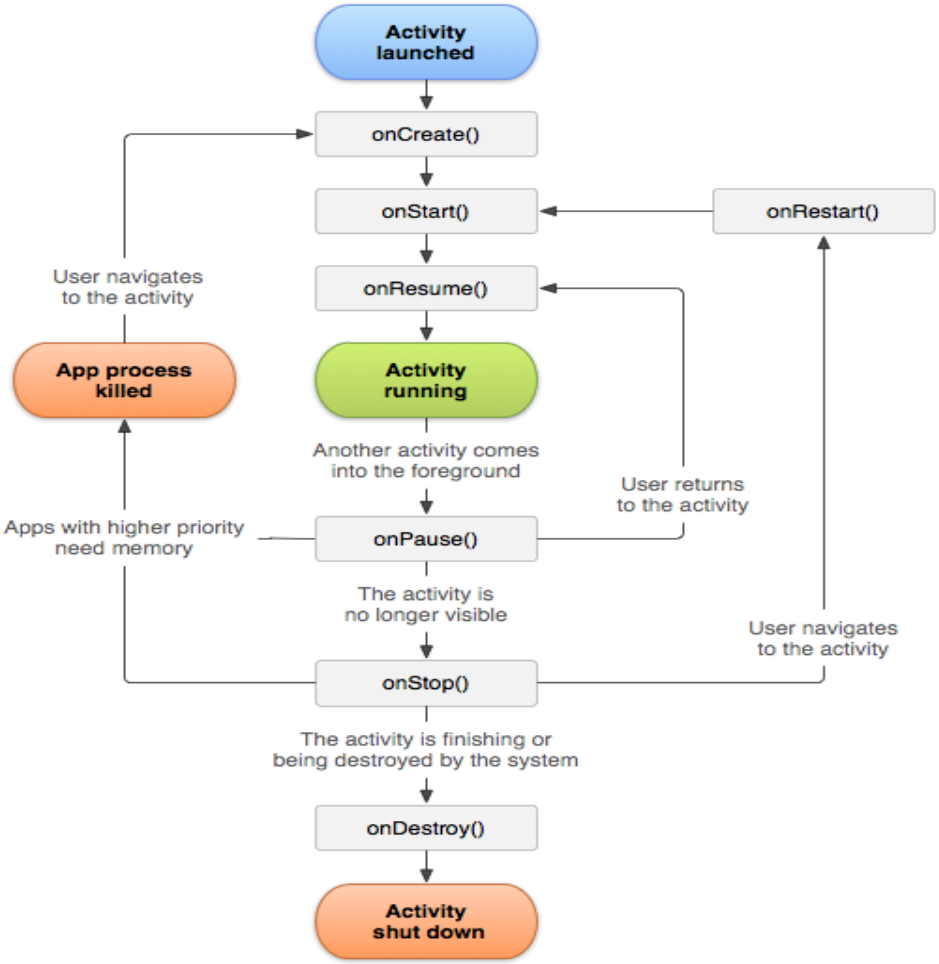
Slika 4.1 Build.gradle – datoteka s bitnim informacijama o aplikaciji i dozvolama za korištenje

U repozitoriju *res* nalaze se resursi kao što su boje, stilovi, slike, rasporedi sučelja i drugo. Najviše se koriste rasporedi sučelja koji su pisani u XML-u i na njih se stavljaju razni *widgeti* (*TextView*, *EditText*, *Button*, *LinearLayout* i drugo) koji se prikazuju na zaslonu uređaja. Kako bi bilo preglednije, repozitorij *res* je podijeljen u dijelove *drawable* (ovdje se nalaze slike i izgledi kontejnera), *layout* (ovdje stavljamo rasporede sučelja), *mipmap*, *values* (ovdje se nalaze boje, stilovi, fontovi), *xml* (kreiranje custom tipkovnice).

U mapi *java* nalaze se sve klase koje su napisane za ovaj projekt. Osim klasa koje su izvorno napisane za ovaj projekt, nalazi se i skup klasa u obliku biblioteke pod nazivom *mxparser* koja se u projektu koristi za parsiranje matematičkih izraza.

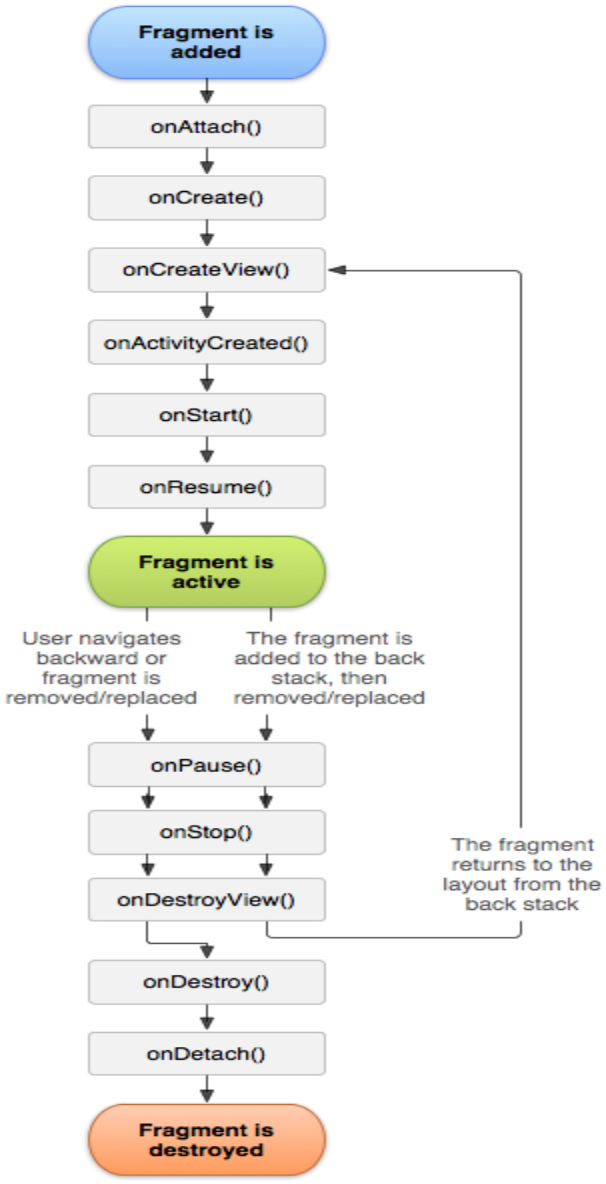
4.1. Fragmenti

Kreiranjem projekta kreira se klasa *MainActivity* i u mapi *res/layout* kreira se datoteka *activity_main.xml*. *MainActivity* klasa i *XML* datoteka *activity_main.xml* zajedno predstavljaju početni zaslon jedne aplikacije. Android aplikacija nema glavnu metodu nego početnu *onCreate()* metodu koja se pokreće pri kreiranju glavne aktivnosti *MainActivity*. Metoda *onCreate()* je jedna od *callback* metoda koja se poziva kada se određeni događaj dogodi u aplikaciji (stvaranje aplikacije, gašenje aplikacije, pauziranje, i drugo). Metode koje se pozivaju su *onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart() i onDestroy().* Životni ciklus *Activity* [8] prikazan je na slici 4.2.



Slika 4.2 Životni ciklus Activity – preuzeto sa https://developer.android.com/guide/components/activities/activity-lifecycle

Fragment je komponenta Android aplikacije, a realizira se klasom *Fragment*. Koristi se kada se želi jedna aktivnost razdijeliti na više sekcija. Najčešće se uz fragmente koristi raspored *Tablayout* (aplikacije koje tu metodu koriste su *Facebook Messenger*, *Whatsapp*, i drugo). Također, fragment [9] ima svoj životni ciklus koji se razlikuje naspram aktivnosti, ali je vezan uz životni ciklus aktivnosti u kojoj se nalazi. Životni ciklus fragmenta prikazan je na slici 4.3.



Slika 4.3 Životni ciklus Fragmenta – preuzeto sa https://www.programmersought.com/article/5948767158/

Za realiziranje ideje kalkulatora, stvaraju se 3 fragmenta koja se prikazuju pomoću *ViewPager* i *TabLayout*. Korištenjem *TabLayout* rasporeda omogućuje pomicanje lijevo/desno kako bi se odabrao željeni fragment, a *ViewPager* (prikazan kod na slici 4.4) će se povezati na fragment koji je povezan s odabranim tabom.

```
package com.example.simboliccalculator;

import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;
import java.util.Locale;

public class ScreenSlidePagerAdapter extends FragmentStatePagerAdapter {

    private static final int NUM_PAGES = 5;
    private static final String BASE_NAME = "Tab #%d";

    public ScreenSlidePagerAdapter(FragmentManager fm) { super(fm);}

    @Override
    public Fragment getItem(int position) {
        switch (position) {
            case 0:
                return new FragmentCalculator();
            case 1:
                return new FragmentFormulas();
            default:
                return new FragmentTables();
        }
    }

    @Nullable
    @Override
    public CharSequence getPageTitle(int position) {
        return String.format(Locale.getDefault(), BASE_NAME, position + 1);
    }

    @Override
    public int getCount() {
        return NUM_PAGES;
    }
}
```

Slika 4.4 ScreenSlidePagerAdapter – klasa za odabir fragmenta

Za upotrebu *ViewPager* i *TabLayou*, trebaju se napisati kodovi u rasporedu sučelja u glavnoj aktivnosti (slika 4.5) i postaviti ID-ove kako bi se moglo spojiti na njih u *Java* klasama.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <android.support.design.widget.TabLayout
        android:id="@+id/tab"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"/>
    <android.support.v4.view.ViewPager
        android:id="@+id/viewPager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>
```

Slika 4.5 activity_main – sučelje početnog zaslona

Unutar klase *MainActivity* (slika 4.6) inicijaliziran je *ViewPager* i time je strukturiran cijeli dizajn aplikacije. Sada je potrebno napisati logiku za svaki fragment zasebno jer svaki fragment ima svoju zadaću.

```java
package com.example.simboliccalculator;

import android.support.design.widget.TabLayout;
import android.support.v4.view.PagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    private ViewPager mViewPager;
    private TabLayout mTabLayout;

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

    private void setUpPager() {

        mViewPager = findViewById(R.id.viewPager);
        mTabLayout = findViewById(R.id.tab);
        PagerAdapter pagerAdapter = new ScreenSlidePagerAdapter(getSupportFragmentManager());
        mViewPager.setAdapter(pagerAdapter);
        mTabLayout.setupWithViewPager(mViewPager);

    }
}
```

Slika 4.6 MainActivity – glavna klasa aplikacije koja inicijalizira *ViewPager*

4.2. Prvi fragment za kalkulator

Na prvom fragmentu se nalazi kalkulator koji je namijenjen za rješavanje jednostavnih matematičkih izraza. U kreiranju izgleda prvoga fragmenta (nalazi se u prilogu na CD-u pod nazivom *fragment_calculator_view.xml*) koriste se *LinearLayout*, *TextView*, *EditText* i *TableLayout* te *Button* widgeti. *LinearLayout* i *TableLayout* su zapravo kontejneri u koje se ubacuju ostali widgeti po rasporedu koji je ugodan korisniku za korištenje kalkulatora. *TextView* prikazuje tekst, a u widget *EditText* upisuju se matematički simboli. Na posljetku, postoje *Button* widgeti koji izvodi zadane funkcionalnosti i pokreću metodu *EventHandler*, koji su zapravo klikovi na zaslon ekrana mobitela ili tableta.

Svi *Button* widgeti reagiraju na klik, osim widgeta *DEL*, koji dodatno reagira na dugi pritisak klika. Cilj je da korisnik ne pritišće samo widget *DEL* kako bi obrisao simbole, nego ima i mogućnost bržeg brisanja simbola kada radi dugi pritisak widgeta, što je sasvim prirodno za implementirati. Kod za obavljanje dugog pritiska na widget prikazan je na slici 4.7.

```java
package com.example.simboliccalculator;

import android.os.Handler;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnTouchListener;

public class RepeatListener implements OnTouchListener {

    private Handler handler = new Handler();

    private int initialInterval;
    private final int normalInterval;
    private final OnClickListener clickListener;
    private View touchedView;

    private Runnable handlerRunnable = new Runnable() {
        @Override
        public void run() {
            if(touchedView.isEnabled()) {
                handler.postDelayed(this, normalInterval);
                clickListener.onClick(touchedView);
            } else {
                handler.removeCallbacks(handlerRunnable);
                touchedView.setPressed(false);
                touchedView = null;
            }
        }
    };

    public RepeatListener(int initialInterval, int normalInterval,
                          OnClickListener clickListener) {
        if (clickListener == null)
            throw new IllegalArgumentException("null runnable");
        if (initialInterval < 0 || normalInterval < 0)
            throw new IllegalArgumentException("negative interval");

        this.initialInterval = initialInterval;
        this.normalInterval = normalInterval;
        this.clickListener = clickListener;
    }
    public boolean onTouch(View view, MotionEvent motionEvent) {
        switch (motionEvent.getAction()) {
            case MotionEvent.ACTION_DOWN:
                handler.removeCallbacks(handlerRunnable);
                handler.postDelayed(handlerRunnable, initialInterval);
                touchedView = view;
                touchedView.setPressed(true);
                clickListener.onClick(view);
                return true;
            case MotionEvent.ACTION_UP:
            case MotionEvent.ACTION_CANCEL:
                handler.removeCallbacks(handlerRunnable);
                touchedView.setPressed(false);
                touchedView = null;
                return true;
        }
        return false;
    }
}
```

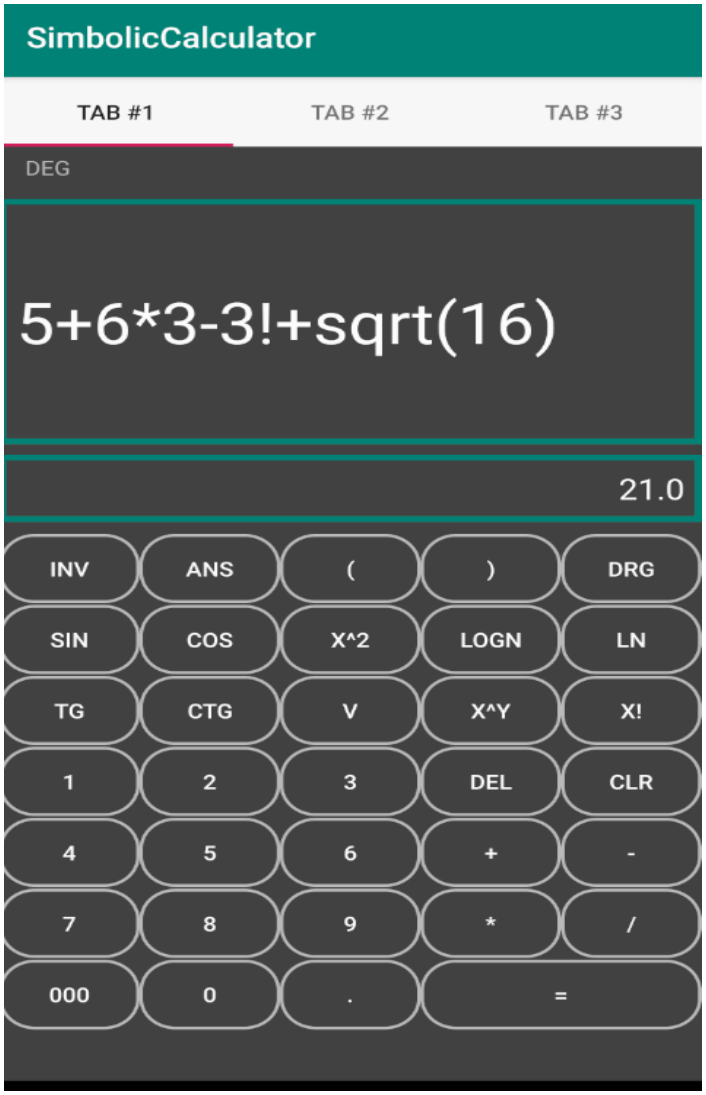
Slika 4.7 RepeatListener – klasa napisana za dugi pritisak *widgeta*

Pritiskom na *Button* widgete prikazuju se simboli na zaslon, koji su zapravo matematički izrazi, te priskom na widget = dobije se rezultat navedenog matematičkog izraza. Iza jednostavnog upisa i rješenja na zaslonu aplikacije stoji kompleksan proces rješavanja istoga. Pritiskom na widget =, aplikacija parsira navedene simbole u matematičke izraze, izračunava izraz, te prevodi simbole u znakove razumljive korisniku. Zbog toga se koristi biblioteka *mxparser* koja sve to omogućava. Glavni kod kojim je realiziran fragment kalkulator nalazi se na slici 4.8.

```
import org.mariuszgromada.math.mxparser.Expression;···

public class FragmentCalculator extends Fragment {

    Button ANSBtn, result, sin, cos, tg, ctg, ln, log, x_fact, btnUp, btnDown;
    TextView display, shiftView, resultView, drgView, tvHistory;
    String ANS = "0", racunajT = "0";
    int pomakniDRG = 0, maxPromjena = 0, selectedText = 0;
    boolean shiftPressed = false, rezPressed = false;
    String[] arrayRacunajT, arrayDisplay;
    StringBuilder strbuildHistory;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_calculator_view, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        arrayRacunajT = new String[20];
        arrayDisplay = new String[20];
        strbuildHistory = new StringBuilder();
        tvHistory = (TextView) view.findViewById(R.id.tv_history);
        display = view.findViewById(R.id.displayText);
        display.setText("0");
        resultView = view.findViewById(R.id.rezultatDisplay);
        shiftView = view.findViewById(R.id.shiftOznaka);
        brojevi_dodaci(view);   osnovne_funkcije(view);
        trig_funkcije(view);    specijalne_tipke(view);
    }

    private void setappendResult(String funDis, String funRacu)···
    private void appendResult(String fun)···

    private void specijalne_tipke(View view) {···

    private void trig_funkcije(View view) {···

    private void osnovne_funkcije(View view) {···

    private void brojevi_dodaci(View view) {···
}
```

Slika 4.8 FragmentCalculator – klasa prvog fragmenta za računanje matematičkih izraza

Kako bi se moglo u kodu kontrolirati što se prikazuje na zaslonu i primati upute od korisnika kada primjerice stisne widget, potrebno je spojiti na zadane widgete s metodom *findViewById*. Naredbom povezujemo java kod i *XML* sučelje te se izvršavaju zadane operacije. Primjer korištenja kalkulatora prikazano je na slici 4.9.



Slika 4.9 Primjer korištenja prvog fragmenta za računanje matematičkih izraza

4.3. Drugi fragment za formule

Drugi fragment je namijenjen učenicima srednjih škola jer se tu koriste dugačke formule. Većinom zadaci koriste iste formule, ali se traži drugi parametar. Često učenik nije siguran u rezultat i korisno bi mu bilo provjeriti rješenje uz pomoć kalkulatora ili aplikacije. Na nekim kalkulatorima zna biti zahtjevno za napisati formulu i za to postoji jednostavno rješenje.

Na slici 4.10 prikazano je sučelje drugog fragmenta za računanje parametara iz formula. Zanimljivo je za primijetiti kako su određeni kontejneri *LinearLayout* prazni. Prazni su na početku ali kasnije kada se dogodi određeni događaj (u ovom konkretnom slučaju pritisak widgeta) dinamički će se mijenjati widgeti na rasporedu odnosno mijenjat će se slike i izmjenjivati će se razni widgeti (*EditText*, *TextView*, *Button*, i drugo).

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#424242">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="140dp"
        android:background="@drawable/edit_text_background"
        android:id="@+id/ll_zaslon"
        android:layout_marginTop="15dp"
        android:orientation="horizontal">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:textColor="@android:color/white"
            android:paddingTop="5dp"
            android:paddingLeft="10dp"
            android:id="@+id/tv_zaslon"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_below="@+id/ll_zaslon"
        android:layout_marginTop="10dp"
        android:id="@+id/ll_opcije"
        android:orientation="horizontal">
        <EditText
            android:hint="opc"
            android:layout_width="80dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="40dp"
            android:background="@drawable/edit_text_background"
            android:textColor="@android:color/white"
            android:paddingLeft="15dp"
            android:id="@+id/et_opcija"/>
        <Button
            android:layout_width="100dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="40dp"
            android:id="@+id/btn_izaberi_opciju"
            android:background="@drawable/ripple_effect"
            android:text="IZABERI"/>
        <Button
            android:layout_width="100dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="20dp"
            android:id="@+id/btn_vratiHome"
            android:text="Izbornik"
            android:background="@drawable/probni"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_marginTop="10dp"
        android:layout_below="@+id/ll_opcije"
        android:id="@+id/ll_parametri_1"
        android:orientation="horizontal">
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_marginTop="10dp"
        android:layout_below="@+id/ll_parametri_1"
        android:id="@+id/ll_parametri_2"
        android:orientation="horizontal">
    </LinearLayout>
    <LinearLayout
        android:layout_width="120dp"
        android:layout_height="60dp"
        android:layout_marginTop="10dp"
        android:layout_below="@id/ll_parametri_2"
        android:id="@+id/ll_izracunajGumb"
        android:orientation="horizontal">
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_marginTop="10dp"
        android:layout_below="@id/ll_parametri_2"
        android:layout_toRightOf="@+id/ll_izracunajGumb"
        android:id="@+id/ll_izlazni_parametri"
        android:orientation="horizontal"
        android:layout_centerHorizontal="true">
    </LinearLayout>
</RelativeLayout>
```

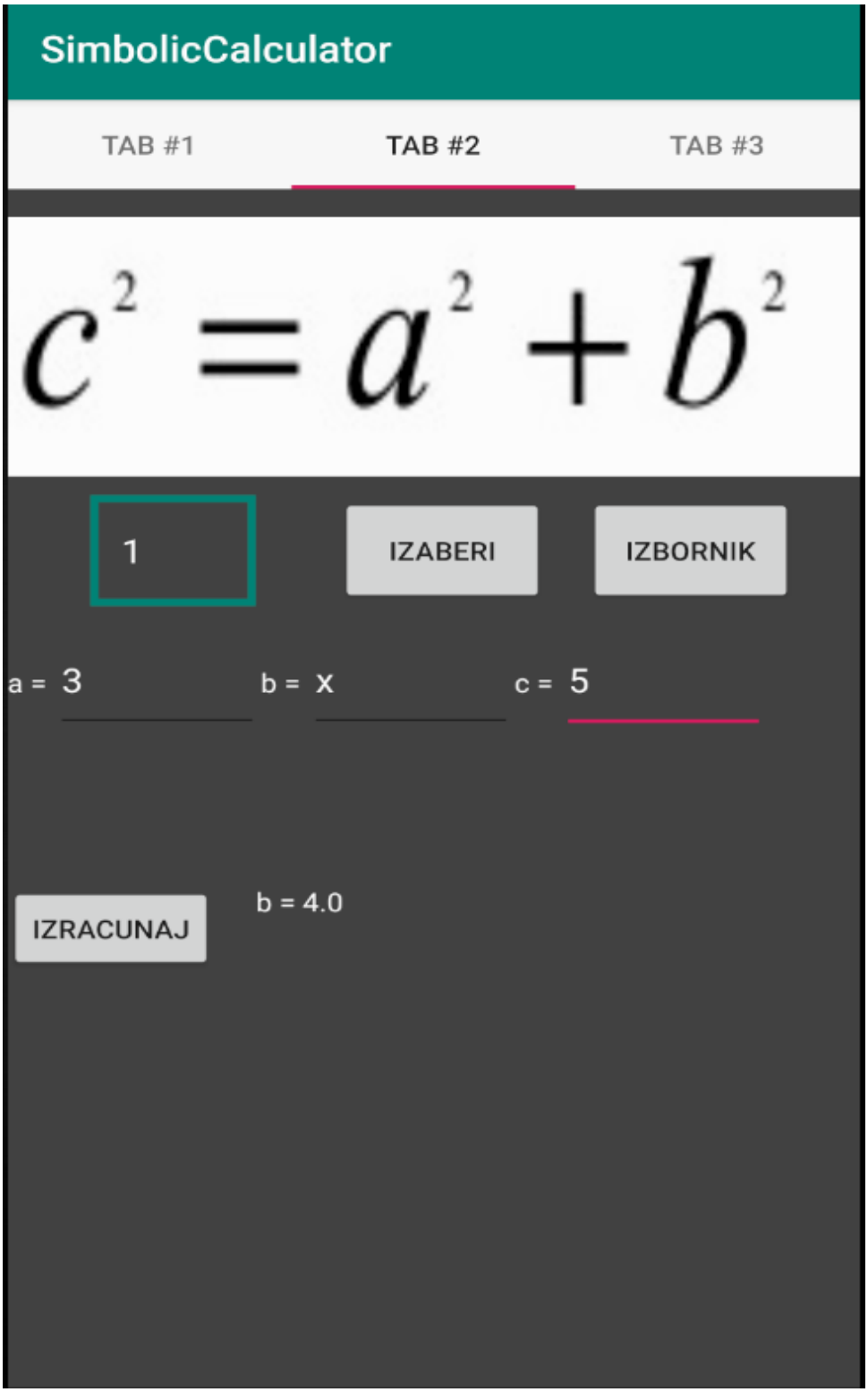
Slika 4.10 fragment_formulas_view – sučelje drugog fragmenta za računanje formula

Unutar klase *FragmentFormulas* (slika 4.11) nalaze se metode za dobivanje nepoznatog parametra iz odabrane formule. Napisana je logika kako dobiti svaki parametar iz formule (može se vidjeti u prilogu na CD-u).

```
package com.example.simboliccalculator;

import android.app.DatePickerDialog;…

public class FragmentFormulas extends Fragment {

    TextView[] parametarTV;
    EditText[] parametarET;
    TextView zaslon;
    EditText opcija;
    ImageView zaslonImage;
    Button opcijaGumb, calculate, izbornik;
    LinearLayout zaslonLayout, kontejnerUlazni, kontejnerUlazni_2, kontejnerIzlazni, kontejnerGumb;
    LinearLayout.LayoutParams paramZaslon, paraText, paraEdit;
    String date1, date2;


    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {…

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {…

    private void postaviFormuluNaZaslon(int option) {…

    private void createParametars(TextView[] nameTV, EditText[] nameET, int parametarCount) {…

    private void setParametarsOnLayout(TextView[] nameTV, EditText[] nameET, int parametarCount) {…

    private void setParametarsOnLayoutTwo(TextView[] nameTV, EditText[] nameET, int parametarsCount) {…

    private void createRezView(TextView textTV) {…

    private void kreirajGumbZaRacun() {…

    private void pitagorinPoucak() {…

    private void potrosackiKredit() {…

    private void povrsinaTrokuta() {…

    private void kosinusovPoucak() { // omg daje dva rjesenjaaaaaaaaaa, namjesti formulu za dva jaoooo…

    private void getBrojDanaIzmeduDatuma() {…

    private void showDatePickerDialog2() {…

    private void showDatePickerDialog1() {…
}
```

Slika 4.11 FragmentFormulas – klasa drugog fragmenta za računanje formula

U glavnom izborniku, odabere se redni broj formule koje korisnik želi upotrijebiti, prikaže se formula na zaslonu, upišu se traženi parametri koje ima i za nepoznati parametar upiše se simbol x. Pritiskom widgeta *IZRACUNAJ*, aplikacija prikazuje jedinstveno rješenje kao što je prikazano na slici 4.12.



Slika 4.12 Primjer korištenja drugog fragmenta za računanje formula

4.4. Treći fragment za tablice i matrice

Posljedni fragment je namijenjen studentima ekonomije, jer se oni najčešće susreću sa zajmovima i matricama. Često kod vježbanja zadataka studenti doma žele provjeriti svoj račun s matricama i vidjeti jesu li ustanovili gradivo. Studenti koji se susreću s financijama nailaze na zajmove i moraju raditi otplatne tablice koje također žele provjeriti.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#424242">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:background="@drawable/edit_text_background"
        android:id="@+id/tv_zadatak"
        android:paddingTop="5dp"
        android:paddingLeft="10dp"
        android:layout_marginTop="15dp"
        android:textColor="@android:color/white"/>
    <EditText
        android:layout_width="80dp"
        android:layout_height="30dp"
        android:id="@+id/et_opcijaizaberi"
        android:layout_below="@+id/tv_zadatak"
        android:background="@drawable/edit_text_background"
        android:layout_marginTop="15dp"
        android:layout_marginLeft="60dp"
        android:paddingLeft="15dp"
        android:textColor="@android:color/white"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Izaberi"
        android:layout_toRightOf="@+id/et_opcijaizaberi"
        android:layout_below="@+id/tv_zadatak"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="40dp"
        android:id="@+id/btn_odaberi"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="45dp"
        android:orientation="horizontal"
        android:id="@+id/ll_kontejner"
        android:layout_below="@+id/btn_odaberi"
        android:layout_marginTop="10dp">
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ll_zaracunati"
        android:layout_below="@+id/ll_kontejner"
        android:layout_marginTop="10dp"
        android:orientation="horizontal">
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ll_provjera"
        android:layout_below="@+id/ll_zaracunati"
        android:layout_marginTop="10dp"
        android:orientation="horizontal">
    </LinearLayout>
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/ll_provjera"
        android:layout_marginTop="15dp"
        android:id="@+id/scrollKontejner">

    </ScrollView>
</RelativeLayout>
```

Slika 4.13 fragment_tables_view – sučelje trećeg fragmenta za računanje matrica i tablica

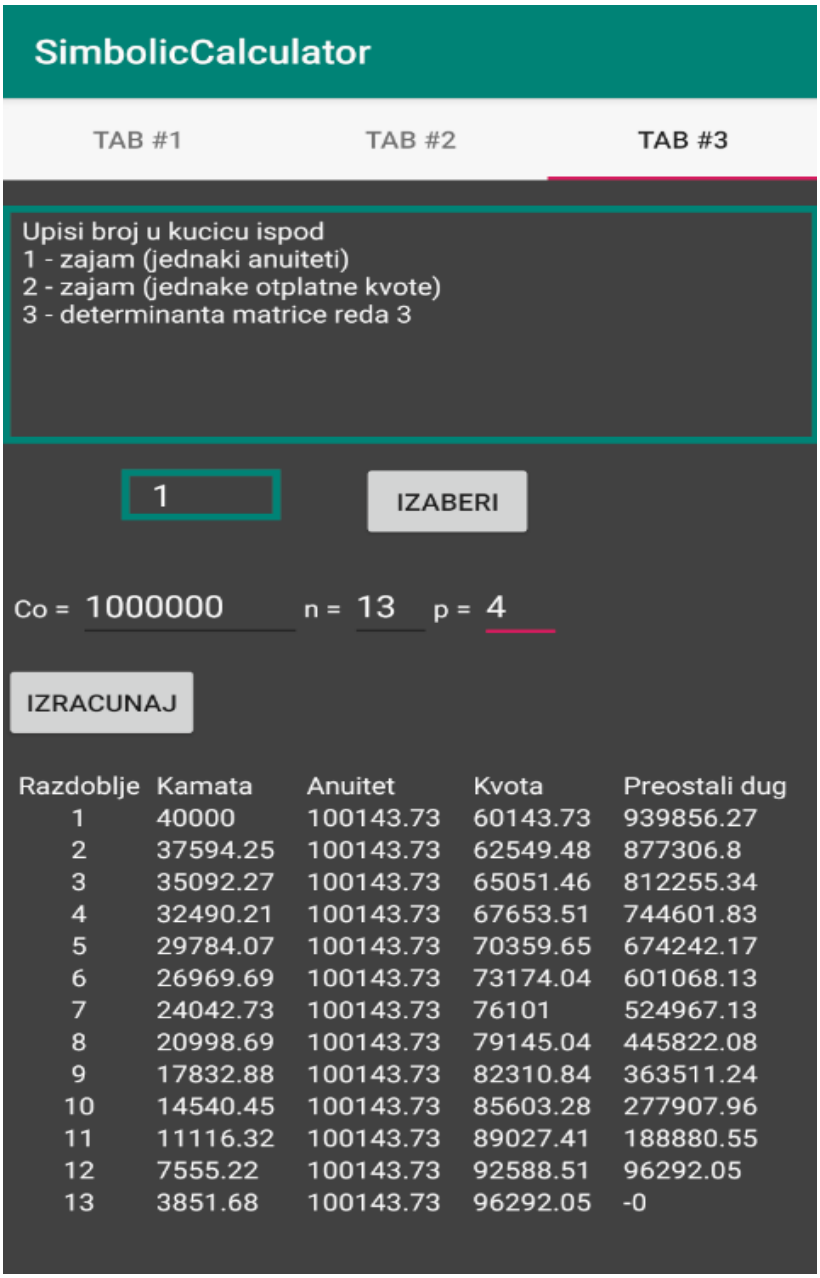
Na slici 4.13 prikazan je izgled sučelja za fragment za tablice i matrice. Koncept je sličan kao na drugom fragmentu za formule, samo je ispis rezultata izmijenjen kako bi bio prikazan u obliku tablica. Na slici 4.14 prikazana je logika fragmenta za računanje matrica i tablica.

```
package com.example.simboliccalculator;

import android.content.Intent;…


public class FragmentTables extends Fragment {

    EditText[] parametarET;
    TextView zadatak, tv;
    EditText opcija;
    TableLayout tableResult;
    Button odaberi, calc;
    ScrollView scrollResult;
    LinearLayout kontejner, kontejner_2, kontejnerProvjera;
    LinearLayout.LayoutParams paraText;
    DecimalFormat df2 = new DecimalFormat("#.##");
    DecimalFormat df4 = new DecimalFormat("#.####");

    int count = 0;

    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {…

    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {…



    private void createParametersZajam(String znak) {…

    private void racunaj_determinantu() {…

    private void zajam_dogovoren_anuitet() {
…
    public void zajam_jednaki_anuiteti() {
…
    public void zajam_jednake_kvote() {
…


    private void createMembersForRow(TextView[] nameTV, int brojStupaca) {…



    private void putTVInRowWithinTable(TableRow rowNumber_members, TextView[] memberTV, int brojStupaca)
    {

        for (int i = 0; i<brojStupaca; i++)
        {
            rowNumber_members.addView(memberTV[i]);
        }
        tableResult.addView(rowNumber_members);
    }



}
```

Slika 4.14 FragmentTables – klasa trećeg fragmenta za računanje matrica i tablica

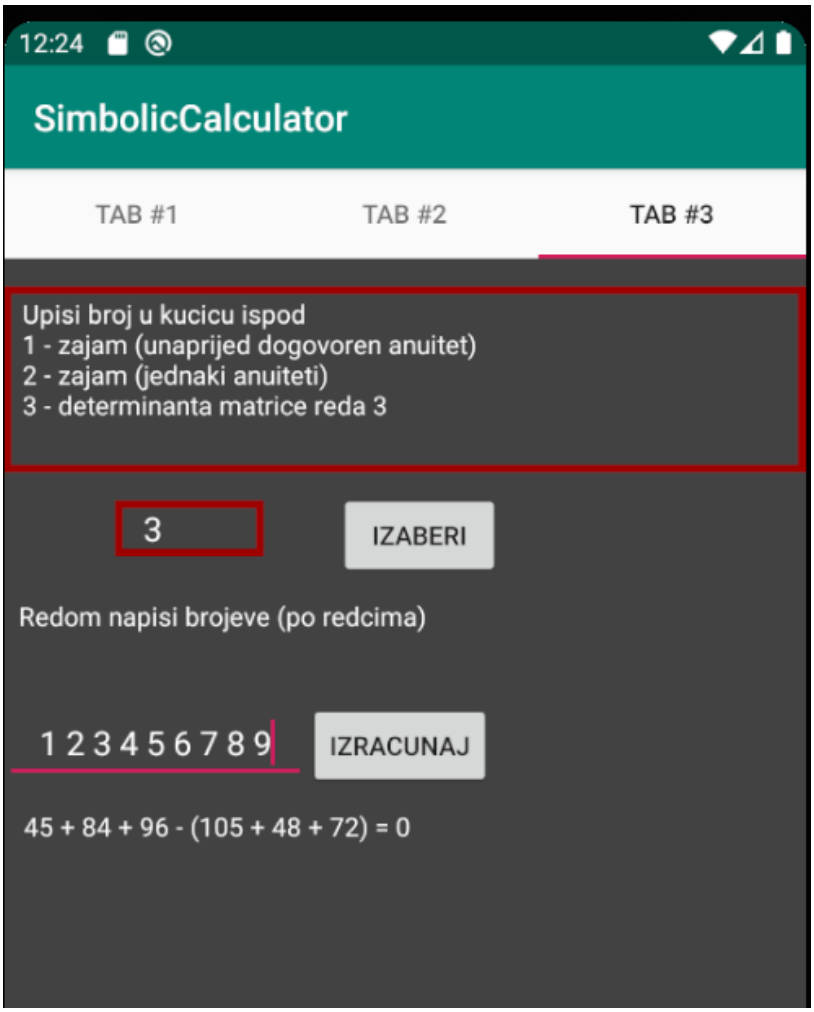
Fragment se koristi tako da se odabere broj ispred opcije. Nakon toga pojave se polja koja se popune brojevima, te pritiskom na widget *IZRACUNAJ* dobije se potpuna matrica ili tablica, ovisno o izboru opcije. Koncept je bio sličan kao kod drugog fragmenta, ali kod zajma dobije se

polje brojeva kao rješenje te je prikladno prikazati rješenje u obliku tablica. Dodatno u rasporedu widgeta koristio se widget *ScrollView* koji omogućava upotrebu klizača za tablicu kada ona izađe s vidnog polja ekrana. Primjer dobivanja otplatne tablice nalazi se na slici 4.15.



Slika 4.15 Primjer korištenja trećeg fragmenta za računanje matrica i tablica

Primjer računanja determinante matrice reda 3 prikazano je na slici 4.16.



Slika 4.16 Primjer korištenja trećeg fragmenta za računanje matrica i tablica

5. ZAKLJUČAK

Cilj aplikacije je stvoriti temelje za kalkulator koji je jednostavan za korištenje, pregledan, lako razumljiv svakome i da provjeri računske operacije u što kraćem vremenu. Po dogovoru, raspored tipki nije napravljen za završi rad s obzirom na kompleksnost projekta i manjak vremena za izradu istoga. Potrebna je daljnja nadogradnja kako bi u potpunosti mogao pomoći učenicima i studentima kroz cijelo obrazovanje te kako bi mogao konkurirati drugim aplikacijama istoga tipa. Temelji aplikacije su napravljeni te se lako proširi funkcionalnost dodavanjem fragmenata. Sve je veća potražnja za mobilnim aplikacijama i veća je upotreba tableta u školama tako da će se u budućnosti kalkulatori na tabletima više koristiti i vjerojatno će se moći koristiti na ispitima, pogotovo na sveučilištima.

PhotoMath ima i postupak za deriviranje i integriranje što je prijekopotrebno za studente, ali Simbolički kalkulator ima najveću primjenu u uštedi vremena. Za učenike srednjih škola, postoji navedena formula, upišu se podaci koje znamo i za parametar koji je nepoznat upiše se simbol x te se dobije rezultat (slika 4.12). U aplikaciji *PhotoMath* trebao bi učenik sam napisati matematički izraz za dobivanje rješenja što za posljedicu troši vrijeme učenika.

Razlika između *Financial Calculators* i Simboličkog kalkulatora jest što se ukupne kamate i cijela otplatna tablica kod Simboličkog kalkulatora nalazi na jednom mjestu. Kod Simboličkog kalkulatora, upisom parametara u predviđena polja, kao što je prikazano na trećem fragmentu za računanje matrica i tablica (slika 4.15), dobivamo cijelu otplatnu tabelu što im uvelike olakšava provjeru rezultata.

Opće je poznato da se determinanta računa kod kvadratnih matrica (matrice s ukupno 4, 9, 16, 25 članova) tako da se može iskoristiti ta informacija. U *Simbolic calculator* napravljeno je da se upišu svi članovi redom i ako je 4 člana upisano, onda se računa determinanta matrice reda 2, ako je upisano 9 članova, onda se računa determinanta matrice reda 3 kao što je prikazano na slici 4.16. Praktičnije je za upisati podatke nego u aplikaciji *Symbolab*.

Učenici i studenti ne koriste *Windows Calculator* zbog njegovog načina rada. Spomenuto je da se prvo upiše broj i onda računska operacija ili funkcija (korjenovanje, sinus funkcija i drugo) kako bi se dobio rezultat. Na nastavi nailaze na kompliciranе matematičke izraze i korištenjem

windows kalkulatora mora se računati u dijelovima kako bi se dobio rezultat. U *Simbolic calculator* jednostavno se upiše cijeli matematički izraz i dobije se krajnji rezultat.

Nedostatak kod Casio kalkulatora je što nema zaslon na dodir. Kod upisivanja dugačkog matematičkog izraza, može doći do pogreške upisivanja jednog parametra i pritiskom na tipke lijevo ili desno mora se doći do pozicije i ispraviti parametar. Pomicanje lijevo ili desno zna biti naporno i oduzima dosta vremena, što na aplikaciji *Simbolic calculator* nije slučaj jer je napravljen za pametne telefone.

LITERATURA

[1] PhotoMath, dostupno na:
https://photomath.com/en/
(19. srpanj 2021.)

[2] Financial Calculators, dostupno na:
https://play.google.com/store/apps/details?id=com.financial.calculator&hl=en&gl=US
(19. srpanj 2021.)

[3] Symbolab, dostupno na:
https://www.symbolab.com/
(19. srpanj 2021.)

[4] Windows Calculator, dostupno na:
https://www.microsoft.com/en-us/p/windows-calculator/9wzdncrfhvn5?activetab=pivot:overviewtab
(19. srpanj 2021.)

[5] Casio fx-991ES PLUS, dostupno na:
https://www.casio-intl.com/asia/en/calc/products/fx-991ESPLUS/
(19. srpanj 2021.)

[6] Android Studio, dostupno na:
https://developer.android.com/studio
(19. srpanj 2021.)

[7] Biblioteka mXparser, dostupno na:
http://mathparser.org/
(19. srpanj 2021.)

[8] Android Lifecycle, dostupno na:
https://developer.android.com/guide/components/activities/activity-lifecycle
(19. srpanj 2021.)

[9] Fragments, dostupno na:
https://developer.android.com/guide/components/fragments
(19. srpanj 2021.)

SAŽETAK

Za završni rad razvijena je *Android* mobilna aplikacija nazvana *Symbolic Calculator* koja je zapravo kalkulator za pametne telefone. Namijenjena je učenicima i studentima u provjeravanju zadataka. Aplikacija je kodirana u razvojnom okruženju *Android Studio* i pisana u objektno orijentiranom programskom jeziku Javi. Uz navedeni kod, prikazan je i primjer rada aplikacije koji se može vidjeti na slikama.

Ključne riječi: Android, Java, simbolički kalkulator

ABSTRACT

SYMBOLIC CALCULATOR

For this bachelor's thesis author developed an Android mobile application called Symbolic Calculator that is actually a calculator for smartphones. It is intended for students to solve their mathematical problems. Application was developed in an environment called Android Studio and was written in Java, an object oriented programming language. In addition to the code, an example usage of the application is shown, which can be seen on the images.

Keywords: Android, Java, symbolic calculator