Mihael Varga
0036471690

Fakultet elektrotehnike i računarstva, Sveučilišta u Zagrebu
Zavod za elektroničke sustave i obradbu informacija
Sveučilište u Zagrebu

Bar code čitač za slijepe

- Namijenjeno slijepim i slabovidnim osobama
- Potrebno poznavanje korištenja pametnih mobilnih uređaja
- Opisana je osnovna ideja i korištene tehnologije

11. lipnja

Sažetak

Ideja ovog seminarskog rada i pripadne aplikacije je omogućiti slijepim i slabovidnim osobama jednostavnije prepoznavanje i kupovinu svakodnevnih namirnica uz pomoć pametnog telefona. Aplikacija omogućava izgovaranje imena predmeta čiji se bar kod skenira. Slijepim i slabovidnim osobama je vrlo teško izvršavati kupovinu bez pomoći drugih osoba jer se prodajna mjesta pretežito orijentiraju na komunikaciju s kupcima pomoću vizualnih alata. Aplikacija iskorištava dostupnost bar kodova na većini proizvoda pa se smanjuje potreba da slijepa i slabovidna osoba opipava i pogađa što drži u rukama. Prednosti ove aplikacije su jednostavno korištenje i unos novih predmeta u bazu podataka. Mana je ta da je potreban mobitel novije generacije Android operativnog sustava te je potreban pristup internetu kako bi se podaci mogli sinkronizirat sa serverom.

Sadržaj

1. UVOD 3
2. BAR KODOVI 4
3. INSTALACIJSKI PAKETI 5
4. KREIRANJE PROJEKTA BARCODETALKER 6
5. STRUKTURA BARCODETALKER ANDROID APLIKACIJE 8
   5.1. Android manifest datoteka (Zrinka Kovačić) 9
   5.2. SplashActivity 11
   5.3. MainActivity 12
   5.4. Izgled aplikacije (Res direktorij) 14
   5.5. Baza podataka (Mihael Varga) 15
      5.5.1. Klasa DbAdapter 15
      5.5.2. Klasa CommonDbAdapter 15
      5.5.3. Klasa BarcodeDbAdapter 16
      5.5.4. Klasa ConfigDbAdapter 16
6. KOMUNIKACIJA S POSLUŽITELJEM (DORIAN KUTLEŠA) 17
7. ZAKLJUČAK 21
8. LITERATURA 22
9. POJMOVNIK 23

Ovaj seminarski rad je izrađen u okviru predmeta „Sustavi za praćenje i vođenje procesa“ na Zavodu za elektroničke sustave i obradbu informacija, Fakulteta elektrotehnike i računarstva, Sveučilišta u Zagrebu.

1. Uvod

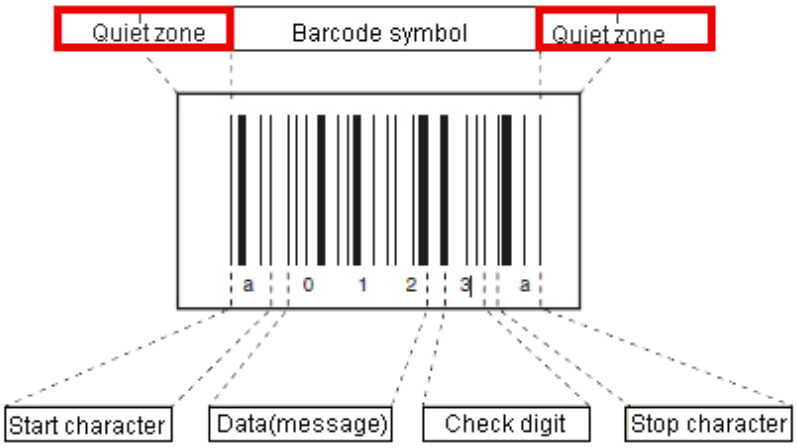
Kako je svijet većinski prilagođen osjetilu vida, slijepe i slabovidne osobe imaju velikih problema sa snalaženjem u okolini i osnovnim zadacima kao što je pronalaženje i kupovina prehrambenih, ali i drugih namirnica. Naša ideja je bila kreirati sustav koji će barem malo olakšati kupovinu slijepim i slabovidnim osobama. Sustav se sastoji od aplikacije i poslužitelja. Aplikacija omogućava skeniranje bar koda i izgovaranje imena skeniranog proizvoda. Ako skenirani proizvod ne postoji u bazi podataka, omogućen je jednostavan unos bar koda i zvučni zapis imena u bazu podataka. Upotrebom aplikacije uz TalkBack opciju dostupnu na Android operativnom sustavu omogućena je navigacija bez potrebe za vizualnom povratnom informacijom.

2. Bar kodovi

Bar kod je optička reprezentacija podataka namijenjena strojnom čitanju pomoću optičkih čitača. On opisuje objekt (ugl. proizvod) na kojem se nalazi, a može opisivati i lokaciju ili čak osobu. Sastoji se od crnih i bijelih linija koje su grupirane u pet zona:

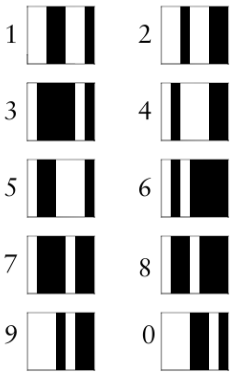
- Tiha zona
- Početni znak
- Podatkovni znakovi koji mogu uključivati opcionalni znak za provjeru
- Zaustavni znak
- Još jedna tiha zona

Navedene zone su jasno vidljive na slici:



Slika 1: Izgled i dijelovi bar koda

Sljedeća slika prikazuje zapise dekadskih brojeva pomoću bar kod zapisa:



Slika 2: Bar kod zapisi dekadskih brojeva

3. Instalacijski paketi

Za rad na projektu, bilo je potrebno instalirati slijedeće programske pakete:

- Java SE Development Kit 8
- Android Studio 2.3.2
- Xampp

Android Studio je razvojni alat koji sadrži editor i Android SDK sa svim pratećim pomoćnim alatima koji su nužni za razvoj, testiranje i objavljivanje Android aplikacija. Programski kod za Android aplikaciju pisan je u objektno-orijentiranom jeziku Java. Stoga je nužno, prije instalacije najnovije verzije Android Studija 2.3.2, imati instaliran odgovarajući Java Development Kit (JDK). Za stvaranje poslužitelja, potrebno je instalirati program Xampp (Apache HTTP).

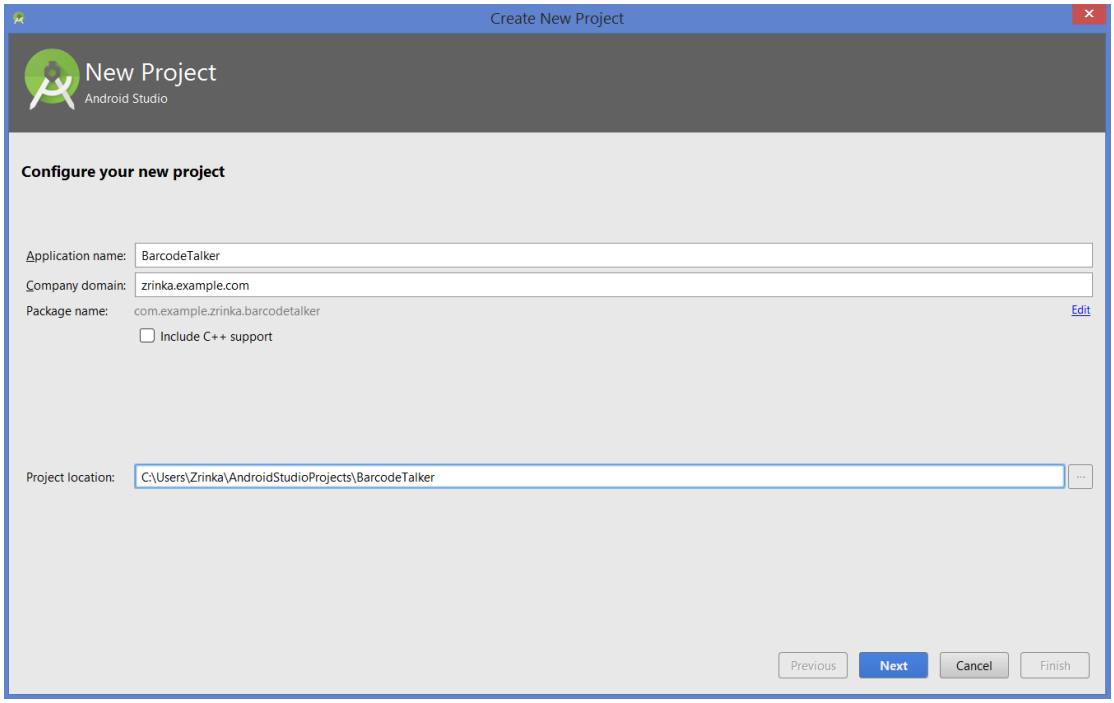
Java SE Development Kit 8 za 64-bitni Windows operacijski sustav može se preuzeti na linku:
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Android Studio 2.3.2 može se preuzeti na linku:
https://developer.android.com/studio/index.html

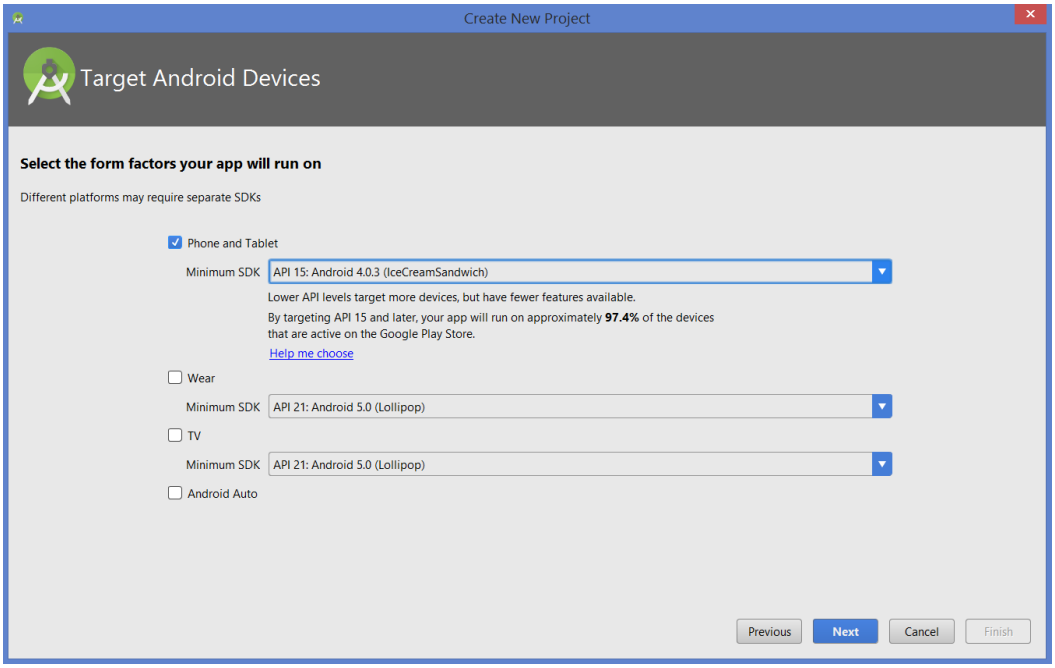
Xampp za bilo koji operacijski sustav može se preuzeti na linku: https://www.apachefriends.org/download.html

4. Kreiranje projekta BarcodeTalker

Prvi korak u stvaranju Android mobilne aplikacije, nakon instalacije svih potrebnih programskih paketa, je stvaranje novog Android projekta. Odabrani naziv projekta i same aplikacije je „Barcode Talker" (Slika 3). Isti naziv odgovara nazivu direktorija u koji se spremaju programski kod, biblioteke i resursi. Također, potrebno je odabrati platformu na kojoj će se vrtiti razvijena aplikacija (Slika 4). Odabirom verzije API 15: Android 4.0.3 (IceCream Sandwich), zadovoljen je minimalni uvjet za univerzalno pokretanje aplikacije na većini uređaja s Android operativnim sustavom. Aplikacija će se moći pokrenuti i na kasnijim inačicama sustava Androida: Android 4.4 (KitKat), Android 5.0/5.1 (Lollipop), Android 6.0.x (Marshmallow) i najnovijoj Android 7.0 (Nougat).

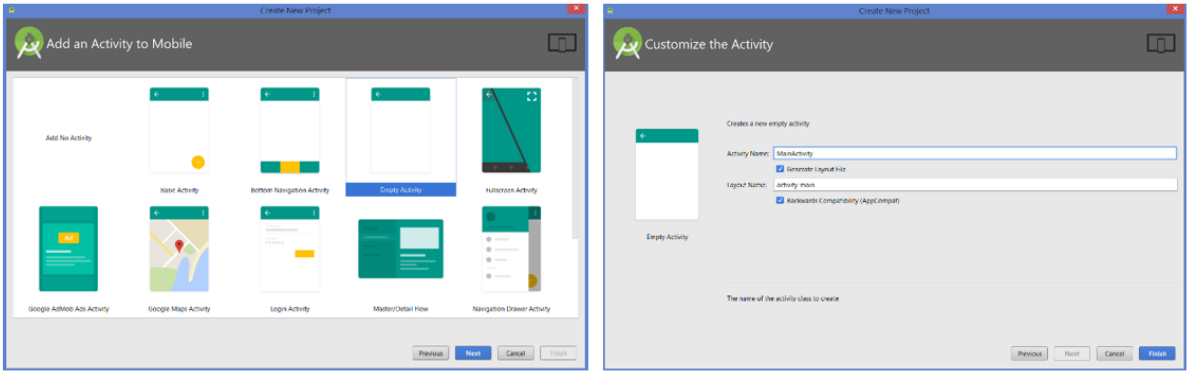


Slika 3: Odabir naziva projekta i aplikacije



Slika 4: Odabir inačice sustava Androida

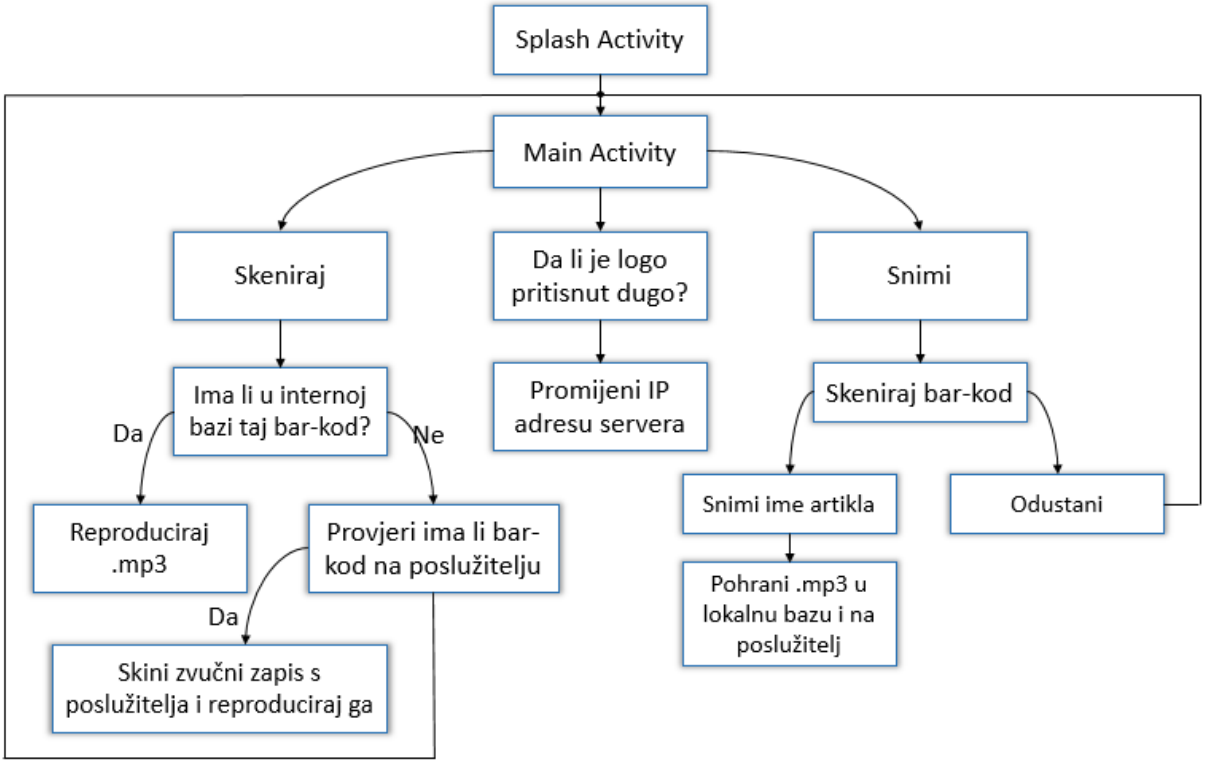
Pri kreiranju novog projekta, dodaje se i početna aktivnost koja predstavlja prvo grafičko sučelje koje vidi korisnik pri pokretanju aplikacije. Potrebno je odabrati praznu aktivnost (*Empty Activity*) i dodijeliti joj naziv *MainActivity*.



Slika 5: Empty Activity i Main Activity

5. Struktura *BarcodeTalker* Android aplikacije

Kako su Android aplikacije pisane u Java programskom jeziku, programski kod se kompajlira u izvršni kod. Izvršni kod je datoteka s ekstenzijom .apk koja se može instalirati na mobilni uređaj s operacijskim sustavom Android. U nastavku će biti opisane glavne komponente BarcodeTalker aplikacije.



Slika 6: Blok dijagram Android aplikacije

5.1. Android manifest datoteka (Zrinka Kovačić)

Manifest je strukturirana XML datoteka naziva *AndroidManifest.xml* u kojoj se uređuje aplikacijski pristup sustavu uređaja. U njemu se nalaze informacije o komponenatama aplikacije potrebne Android operacijskom sustavu prije pokretanja same aplikacije. To je potrebno kako bi mobitel znao što treba napraviti s aplikacijom te koje je resurse mobitela potrebno dopustit da bi aplikacija radila.

Prvi redak je obavezan u svakom XML dokumentu i označava njegov početak. Definira se verzija XML-a i način zapisa (UTF-8). Unutar *<manifest>* i *</manifest>* unosi se cijeli kod s postavkama aplikacije.

Prvo se deklariraju dozvole koje se moraju tražiti od korisnika kako bi se mogle koristiti zaštićene mogućnosti mobilnog uređaja. Traži se dopuštenje korisnika za korištenje kamere, interneta, snimanje zvuka. Također, potrebno je dobiti dopuštenje za pristup medijima i datotekama na mobilnom uređaju korisnika, za spremanje i čitanje mp3 zapisa.

Preko oznaka *<uses-feature android: name="..."/>* navode se potrebne sklopovske i programske značajke koje mobilni uređaj mora sadržavati da bi mogao vrtiti aplikaciju Barcode Talker. To su kamera i autofokus. Bez mogućnosti autofokusiranja, ne bi bilo moguće uslikati barkod.

```
      <?xml                version="1.0"                encoding="utf-8"?>
<manifest     xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="hr.spvp.barcodetalker"                                      >
    <!--android:roundIcon="@mipmap/ic_launcher_round"-->

    <!--You          need          camera          for          scanner-->
    <uses-permission     android:name="android.permission.CAMERA"     />
    <!--You    need    Internet    for    communication    with    server-->
    <uses-permission    android:name="android.permission.INTERNET"    />
    <!--You need Write External Storage for saving mp3 files in folder-
->
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <!--You need Read External Storage for reading mp3 files from folder-
->
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <!--You      need      Record      Audio      for      recording      -->
    <uses-permission    android:name="android.permission.RECORD_AUDIO"/>
    <!--You    need    Write    Internal    Storage    for    scanners    data-->
    <uses-permission
android:name="android.permission.WRITE_INTERNAL_STORAGE"/>

    <!--With    this    you    filter    applications    that    don't    satisfy    your
requirement,    in    this    case    you    need    camera    with    auto    focus    -->
    <uses-feature        android:name="android.hardware.camera"        />
    <uses-feature   android:name="android.hardware.camera.autofocus"   />
```

Primjer 1: AndroidManifest.xml, traženje dopuštenja od korisnika

U manifestu su također definirane ulazne točke korisnika u program, aktivnosti. Aplikacija BarcodeTalker se sastoji od dvije aktivnosti *SplashActivity i MainActivity*. Zadatak aktivnosti je prikaz korisničkog sučelja programa i omogućavanje interakcije korisnika s programom. Više međusobno povezanih aktivnosti čine aplikaciju. Aplikacijske komponente se aktiviraju preko namjera (engl. *intents*). Pomoću filtera za namjere (engl. *intent filter*) definira se početna aktivnost. Obvezni filteri su *MAIN* koji definira da će po pokretanju aplikacije prva aktivnost biti *SplashActivity* i *LAUNCHER* za pokretanje same aplikacije.

```
      <application
      <!--SplashActivity-->
        <activity
            android:name="hr.spvp.barcodetalker.SplashActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar"                >
            <intent-filter>
                <action   android:name="android.intent.action.MAIN"   />

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

        <!--MainActivity-->
        <activity
            android:name="hr.spvp.barcodetalker.MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar"                >
        </activity>
    </application>
```

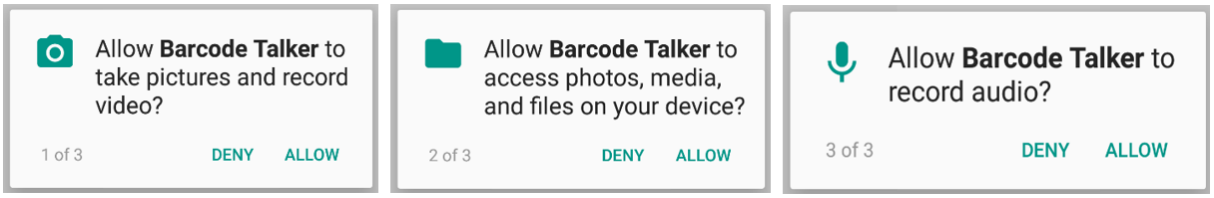
Primjer 2: AndroidManifest.xml, definiranje aktivnosti u aplikaciji

5.2. SplashActivity

Prva aktivnost koja se otvara pri pokretanju aplikacije prema definiciji u manifestu je SplashActivity. To je početno grafičko korisničko sučelje pri kojem se u pozadini odrađuju sljedeće funkcionalnosti:

- Metoda *onCreate()* – prilikom pokretanja bilo koje aktivnosti koja ima grafičko korisničko sučelje iz onCreate-a poziva se savedInstanceState. U njemu su spremljene postavke aplikacije gdje je aplikacija prije prethodnog zatvaranja stala s korištenjem. U našem slučaju aplikacija se svaki put pokreće od nule pa nema savedInstanceState-a.
- setContentView() – dodjeljuje aktivnosti izgled sučelja definiran u activity_splash.xml datoteci
- dbAdapter – kreira lokalnu SQLite bazu prilikom prvog pokretanja aplikacije. Bazu je potrebno otvoriti i zatvoriti.
- provjera verzije Androida – za verzije starije od 6.0 ne traže se dopuštenja od korisnika, za novije verzije Android sustava od 6.0 moraju se dobiti dopuštenja korisnika radi pristupa zaštićenim podacima
- traženje dopuštenja za: upotrebu kamere, pristup medijima i datotekama te korištenje mikrofona

Prozori na kojima se traže dopuštenja se otvaraju prilikom prvog pokretanja aplikacije. Dok se ne odgovori potvrdno na sva dopuštenja, aplikacija neće moći raditi. Prozori su prikazani na slici 6.



Slika 7: Potrebna dopuštenja od korisnika za ispravan rad aplikacije

U slučaju da korisnik odbije neki od zahtjeva za navedena dopuštenja, taj zahtjev se dodaje na listu *listPermissionNeeded*. Ukolika je ta lista prazna, odnosno svi upiti su prihvaćeni, metodom *makeFolder()* stvara se datoteka za spremanje mp3 zapisa te se metodom *moveToMain()* prelazi u glavnu aktivnost aplikacije MainActivity.

Metoda u jeziku Java je skupina naredbi (blok) koje grupno obavljaju neku operaciju. Metoda se izvršava pozivanjem (*calling or invoking*) iz nekog dijela programa gdje je korištenje metode potrebno. Svaki program mora sadržavati bar jednu metodu da bi obavio neki zadatak, te mora sadržavati metodu „main".

5.3. MainActivity

Nakon *SplashActivity*-a prelazi se u glavnu aktivnost u aplikaciji: *MainActivity*. U nastavku je opisan pseudokod programa *MainActivity*:

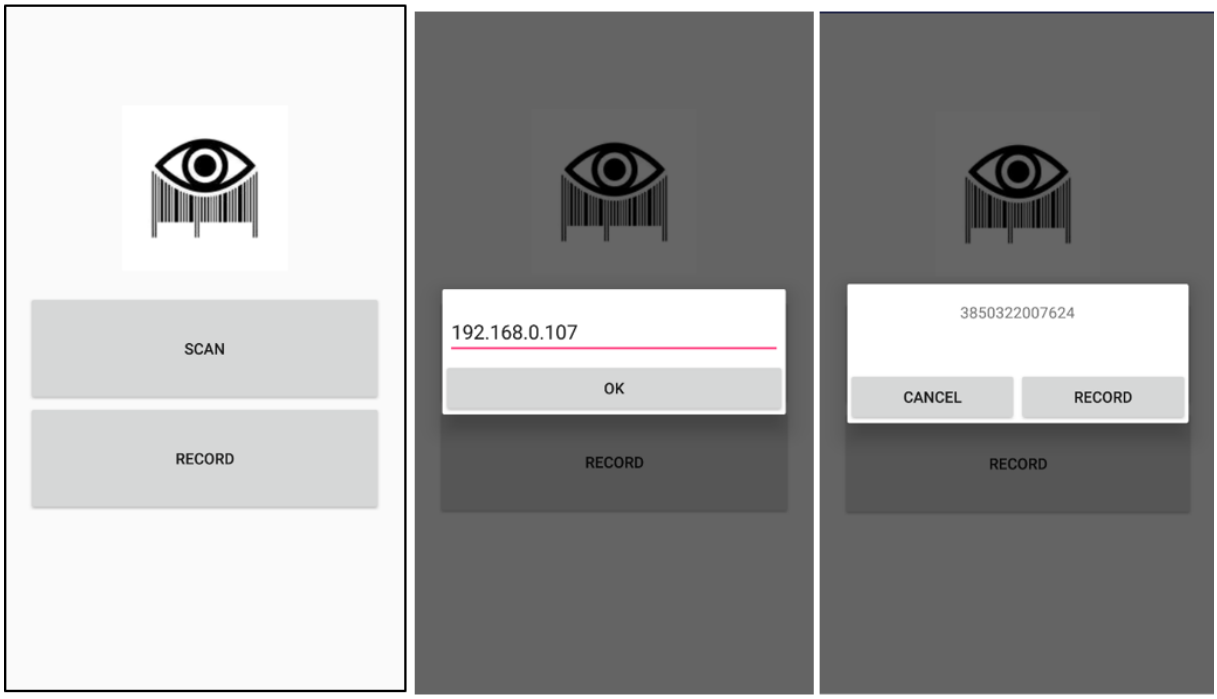
- definiranje stringova, Media player, Media recorder i JSON parser
- stvaranje sljedećih klasa:
  - configDbAdapter tipa ConfigDbAdapter – dohvaćanje IP adrese servera
  - logoView tipa *ImageView* te mu pridijeljuje metoda *changeIpAddressDialog()* koja se aktivira u slučaju *setOnLongClickListener*-duljim držanjem logo-a na ekranu, otvara se dijalog u koji je potrebno upisati IP adresu servera
  - *buttonScan* i *buttonRecord* tipa *Button* i pridijeli im odgovarajuće aktivnosti *scanScanRequest* te *scanRecordRequest* uz *setBeepEnabled(true)*, omogućen *beep* nakon uspješnog skeniranja bar koda

- onActivityResult metoda tipa protected void
  - u klasu *result* tipa *IntentResult* sprema se rezultat aktivnosti *parseActivityResult* – ta aktivnost vraća podatke o skeniranom bar kodu: *barcodeValue=result.getContents();*
  - provjeri koji je zahtjev zatražen: *scanScanRequest* ili *scanRecordRequest*
    - if(scanScanRequest) → requestCode==1
      - ukoliko result.getContents() != null
        - provjera nalazi li se taj bar kod u lokalnoj bazi podataka
          - ako da, dohvaća se putanja do .mp3 datoteke odgovarajućeg bar koda
          - pokreće se Media Player i reproducira .mp3 datoteka
          - nakon završetka reprodukcije zatvara se Media Player *myMediaPlayer.stop()*
        - ako nema zapisa u lokalnoj bazi, tada potraži zapis na serveru
    - if(scanRecordRequest) → requestCode==2
      - ukoliko result.getContents() != null
        - sprema se u *barcodeValue* podatak o skeniranom bar kodu
        - poziva se aktivnost *recordDialog(barcodeValue)* koja snima naziv novog artikla u .mp3 formatu i sprema ga u internu bazu i na server
    - else: ne bi se smjelo dogoditi jer postoje samo dva zahtjeva (u slučaju pogrešnog *requestCode*-a)
- parseActivityResult metoda tipa IntentResult
  - zaprima: requestCode, resultCode i klasu intent
  - za scanScanRequest i scanRecordRequest
    - ako je resultCode potvrdan
      - u klasu *intent* upiši podatke o skeniranom bar kodu (npr. bajtovi predstavljeni bar kodom, brojčana vrijednost bar koda)
      - stvori novu klasu *IntentResult* i u nju prekopiraj potrebne podatke iz klase *intent* te vrati tu novu klasu
    - u protivnom vrati praznu novu klasu *IntentResult*
  - u protivnom vrati null

U klasi *CheckOnServer* nalazi se više metoda koje izvršavaju zasebne zadatke. Metoda *onPreExecute()* otvara dijalog s tekstom „Checking status on server. Please wait...". Metodom *doInBackground()* u pozadini se šalje request na server (json) i čeka se odgovor (engl. *response*) sa servera. Ako je uspješan (success==1), nađen je na serveru i u JSON response-u dolazi link za skidanje mp3 zvučnog zapisa i sprema se u lokalni direktorij na pametnom telefonu, tj. pohranjuje se u lokalnoj bazi te se pokreće Media player i pušta se mp3 zapis imena skeniranog artikla. Metodom *onPostExecute()* odbacuje se prethodno spomenuti dijalog.

5.4. Izgled aplikacije (Res direktorij)

Res direktorij sadrži resurse projekta kao što su slike i XML datoteke koje opisuju izgled sučelja. U poddirektoriju drawable nalazi se slika (logo) koji smo odlučili koristiti za našu aplikaciju. Poddirektorij layout sadrži XML datoteke koje opisuju prikaz komponenti na ekranu. U poddirektoriju values sadržane su XML datoteke sa stringovima koji se koriste u projektu, bojama, dimenzijama teksta.



Slika 8: Glavni ekran; dijalog za promjenu IP adrese; dijalog za snimanje zvučnog zapisa

5.5. Baza podataka (Mihael Varga)

Moj dio projektnog zadataka sastojao se od izrade baze podataka. Baze podataka su općenito organizirane skupine nekih podataka koji predstavljaju informacije o određenom objektu vezanom uz stvarni svijet (npr. podaci o osobi: spol, visina, starost i sl.). U našoj aplikaciji baza nam služi za pohranu sljedećih podataka: ID-a pripadajućeg bar koda (redni broj bar koda), brojčane vrijednost bar koda i putanje do audio zapisa tog bar koda (mjesta gdje je pohranjen u memoriji uređaja).

5.5.1. Klasa DbAdapter

Klasa koja služi za kreiranje početnih podataka te kasnije održavanje baze. Ova klasa se poziva samo kod otvaranja aplikacije iz klase *splashActivity* i kasnije se ne poziva.

Unutar *DbAdaptera* nalaze se deklaracije klasa korištenih za otvaranje baze podataka. Jedna od tih klasa je klasa *DatabaseHelper* kojoj pridružujemo klasu *SQLiteOpenHelper.* Unutar te pridružene klase nalazi se metoda s kojom postavljamo početne postavke SQLite (engl. *Structured Query Language*) relacijske baze podataka (baze unutar Android sustava). Stvara se prazna tablica i u nju se upisuju prazni unosi.

Unutar ove klase se također nalazi metoda *open()* koja pokušava kreirati funkcionalnu bazu podataka tako da upiše spomenute početne postavke u sada stvorenu bazu podataka. Ako se baza podataka ne uspije kreirati, tada se postupak ponavlja sve dok se baza podataka ne stvori. Metoda *close()* zatvara bazu nakon upisa u nju.

5.5.2. Klasa CommonDbAdapter

Ovo je klasa koja unutar sebe sadrži tri klase i tri metode korištene u *ConfigDbAdapter* i u *BadcodeDbAdapter* klasama. Ova klasa nam zapravo služi kako ne bismo morali dva puta kopirati isti dio koda u dvije gore navedene klase: mi pridružimo *CommonDbAdapter* tim klasama.

Metode sadržane unutar *CommonDbAdapter:*

- *fetchStringData* – dohvaća iz baze podataka traženi znakovni niz koji tražimo na temelju ID-a bar koda. Ova metoda zahtjeva ID bar koda, odnosno redak u kojem se on nalazi. Ukoliko pronađe taj redak, vraća znakovni niz u kojem je sadržan podatak o ID-u retka, ili vraća *null* ukoliko traženi redak ne postoji.
- *updateStringRawValue* – metoda koja služi za ažuriranje postojećeg artikla unutar baze. Potrebno joj je proslijediti ID bar koda i podatke koje želimo ažurirati u postojeći artikl. Ako ažuriranje prođe uspješno, metoda vraća *true*, a u suprotnom *false.*

- *fetchLongData* – dohvaća iz baze podataka podatke *long* tipa. Slično kao metoda *fetchStringData,* njoj prosljeđujemo ID traženog bar koda. Ako taj podatak postoji u bazi podataka, kao povratna vrijednost vraća se ID traženog bar koda, a ako podatak ne postoji u bazi podataka tada se vraća „0".

5.5.3. Klasa BarcodeDbAdapter

Ovo je klasa koja nam služi za pohranu bar kodova i putanja do pripadnih audio zapisa u *.mp3* formatu. Android sustav radi brže ako za postojanje traženog podatka pogleda u lokalnoj bazi podataka umjesto da se pretražuje cijelu mapu sa spremljenim sadržajem da vidi postoji li traženi podatak. Ovakva izvedba omogućava optimizaciju aplikacije u smislu da se ona izvodi brže.

Unutar ove klase se najprije definiraju znakovni nizovi koji opisuju naš proizvod, a to su:

- BARCODE_ID – broj retka u kojem se nalazi naš proizvod
- BARCODE_CODE – brojčana vrijednost bar koda
- BARCODE_MP3_PATH – putanja do mjesta gdje je audio zapis spremljen u memoriji

Slično kao i *DbAdapter* ova klasa sadrži metode *open* i *close* koje služe za otvaranje i zatvaranje lokalne baze podataka. Razlika je da se ove metode pozivaju kod svakog skeniranja barkoda (i kod opcije *Scan* i kod opcije *Record*).

Metoda *createRawBarcode* je ta koja ubacuje podatke u bazu podataka. Kao argumente ova metoda traži barkod proizvoda i putanju do audio zapisa. Ukoliko se novi artikl ne uspije upisati u bazu podataka metoda vraća „-1". Uspješno pisanje u bazu podataka rezultira vraćanjem ID-a spremljenog bar koda.

5.5.4. Klasa ConfigDbAdapter

Klasa koja služi za promjenu IP adrese servera. I ova klasa sadrži metode *open* i *close.* Kod prvog pokretanja aplikacije IP adresa je podešena na 192.168.1.6 zbog razloga testiranja aplikacije na mreži s fiksnom IP adresom servera. Adresa se mijenja na način opisan u dijelu teksta za korištenja aplikacije.

6. Komunikacija s poslužiteljem (Dorian Kutleša)

Moj zadatak je bio riješiti poslužiteljsku stranu ovoga projekta. Iako bi se baza podataka mogla voditi i samo lokalno na mobilnom telefonu, odlučeno je kako je pogodnije koristiti bazu podataka na poslužitelju. Glavni razlog je stvaranje velike baze podataka proizvoda ako se aplikacijom služi mnoštvo korisnika. Kako je u aplikaciju potrebno unijeti zvučni zapis imena svakog proizvoda, ovakvim pristupom se višestruko ubrzava punjenje baze podataka.

Za poslužitelj smo odlučili iskoristiti program XAMPP. Ovaj program je razvijen od strane Apache Friends programerskog tima s ciljem olakšavanja implementacije Apache servera u projekte njegovih korisnika. Program je otvorenog koda te je potpuno besplatan, kako za komercijalnu upotrebu tako i za svrhu učenja. Osnovne funkcionalnosti su mu Apache HTTP server, FileZilla FTP klijent i MySQL poslužitelj baze podataka.

Naš poslužitelj je Apache HTTP. Apache poslužitelj je vrlo popularna i besplatna platforma otvorenog tipa koja radi na principu *Hypertext Transfer* protokola. Još jedna prednost XAMPP programskog alata je mogućnost izvođenja skripti s kojima se obrađuju primljeni i poslani paketi. Korišteni skriptni jezik je PHP.

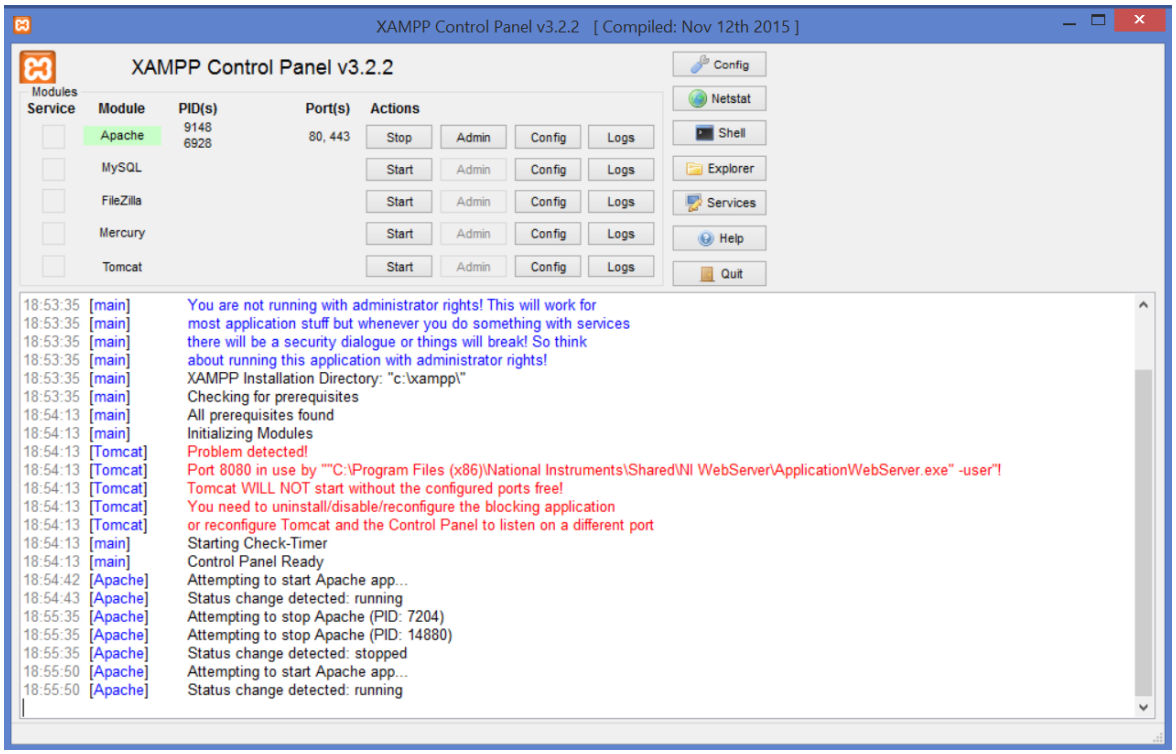
PHP je skriptni jezik nastao 1994. godine za poslužiteljsku stranu komunikacijskog kanala i jedan je od najfleksibilnijih i najkorištenijih skriptnih jezika ovoga tipa. Kompatibilan je s gotovo svim operativnim sustavima te sa većinom web poslužitelja.

HTTP protokol je najrašireniji protokol za prijenos podataka. Nastao je 1989. godine te je doživio veliku reviziju 2015. godine gdje je nastala verzija HTTP/2 koja je i danas u upotrebi na gotovo svim web poslužiteljima. Ovaj protokol definira metode postavljanja i dohvaćanja podataka s poslužitelja. Metode i njihove funkcije su navedene u tablici.

Tablica 1. Metode i funkcije u HTTP protokolu

GET	Dohvaća željene podatke sa poslužitelja
HEAD	Dohvaća zaglavlje željene datoteke
POST	Postavlja podatke na poslužitelj
PUT	Sprema podatke na poslužitelj
DELETE	Briše podatke s poslužitelja
TRACE	Vraća zahtjev klijentu gdje se mogu vidjeti promjene zbog međuposlužitelja
OPTIONS	Vraća HTTP metode dostupne na poslužitelju
CONNECT	Mijenja zahtjev za spajanje u TCP/IP konekciju
PATCH	Sprema modificirane podatke na poslužitelju

Program XAMPP pokreće se pritiskom na „XAMPP Control Panel" u start izborniku nakon instalacije. Klikom na gumb Start pokreće se osluškivanje portova za Apache server. U našem slučaju korišten je standardni TCP port pod brojem 80. TCP protokol nam omogućava pouzdan prijenos podataka jer se preneseni bajtovi provjeravaju kako bi se spriječile pogreške.



Slika 9: Izgled sučelja u XAMPP programu

Pomoću ovog jezika realizirane su dvije skripte. Prva skripta se zove „checkstatusonserver" te nam ona služi za provjeru postoji li zapis s primljenim imenom u bazi podataka. Ako zapis postoji, aplikaciji na mobilnom telefonu se šalje poveznica s koje je moguće skinuti zvučni zapis imena skeniranog artikla. Ako artikl nije pronađen u bazi podataka skripta vraća oznaku greške. U ovu skriptu su također implementirane i funkcije za bilježenje uspješnosti ili neuspješnosti pronalaska u tekstualnu datoteku istog imena kao i skripta. Te funkcije su nam pomogle pri testiranju i otklanjanju problema sa samom aplikacijom i bazom podataka. U nastavku se nalazi kod prve skripte.

```
      <?php
      $file = $_GET['nameOfMp3']; //dohvacanje imena zvucnog zapisa
      error_log('['.date("F  j,  Y,  g:i  a").']  '.$file.     PHP_EOL,3,
"check_status_on_server.log");
      // generiranje tekstualne datoteke dnevnika izvođenja
      $response = array();
      $localIP = getHostByName(getHostName()); // definicije varijabli

      if (is_file($file)){ //ako zvucni zapis postoji
          error_log('['.date("F j, Y, g:i a").'] File exist' .PHP_EOL ,3,
"check_status_on_server.log");
             $response["success"] = 1;
             $response["message"] = "File found. Sending it right away!";
             $response["link"] = 'http://'.$localIP.'/'.$file;
          error_log('['.date("F j, Y, g:i a").'] '.json_encode($response).
PHP_EOL,3, "check_status_on_server.log");
                 echo json_encode($response);
      // slanje obavijesti o uspjesnom pronalasku i poveznica na zvucni zapis

      } else {
             error_log('['.date("F j, Y, g:i a").'] File does not exist'
.PHP_EOL,3, "check_status_on_server.log");
          // no file found
          $response["success"] = 0;
          $response["message"] = "File not found.";
             error_log('['.date("F       j,       Y,       g:i       a").']
'.json_encode($response). PHP_EOL,3, "check_status_on_server.log");
          // echo File not found.
          echo json_encode($response);
      }
      ?>
```

Primjer 3: „Checkstatusonserver" php skripta

Druga PHP skripta se zove „uploadfiletoserver" te nam služi kod unosa novog proizvoda u bazu podataka. Skripta obrađuje primljeni zvučni zapis te ga sprema u definiranu datoteku na poslužitelju. Ovdje su također implementirane funkcije za bilježenje uspješnosti ili neuspješnosti prijenosa te su nam puno pomogle pri ispravljanju grešaka u komunikacijskom kanalu s aplikacijom.

```
<?php
if (is_uploaded_file($_FILES['bill']['tmp_name'])) {
//ako je prijenos preko HTTP-a uspjesan
    $uploads_dir = './';
    $tmp_name = $_FILES['bill']['tmp_name'];
    $mp3_name = $_FILES['bill']['name'];
     move_uploaded_file($tmp_name, $uploads_dir.$mp3_name);
// definiranje koristenih varijabli i spremanje zvunog zapisa u
direktorij
error_log('['.date("F j, Y, g:i a").'] File uploaded successfully.'
.PHP_EOL,3, "upload_file_to_server.log");

}else{

//ako prijenos zvucnog zapisa nije uspjesan
     echo "File not uploaded successfully.";
error_log('['.date("F  j,  Y,  g:i  a").']  File  not  uploaded
successfully.' .PHP_EOL,3, "upload_file_to_server.log");
          }
?>
```

Primjer 4: „Uploadfiletoserver" php skripta

Ove skripte je potrebno postaviti u instalacijski direktorij programa XAMPP u datoteku „htdocs" te moraju imati ekstenziju php. Aplikacija na pametnom telefonu nakon incijalnog spajanja i početka komunikacije s poslužiteljom sama pokreće skripte slanjem HTTP zahtjeva na adresu poslužitelja.

7. Zaključak

Rezultat rada je funkcionalna aplikacija spremna za korištenje. Kao tim smo dobili bolje razumijevanje korištenih tehnologija i uvid u izradu Android aplikacija, kao i njihovo povezivanje sa poslužiteljem. Daljnjim razvijanjem aplikacije moguće ju je pripremiti za korištenje u stvarnom svijetu i pomoć slijepim i slabovidnim osobama.

Ako bi to bio slučaj, potrebno je omogućiti povratnu informaciju ne samo zvukom već i vibracijom što bi omogućilo lakše korištenje u bučnom okruženju. Automatski fokus bi se također trebao doraditi jer većina mobilnih telefona ima vrlo spori fokus pa je potrebno relativno dugo i mirno držati uređaj iznad bar koda.

Poslužiteljski dio sustava je trenutno implementiran na osobnom računalu koje mora biti na istoj mreži kao i mobilni uređaj te mu se pristupa preko fiksne IP adrese. Potrebno je omogućiti spajanje na udaljeni poslužitelj neovisno o mreži na koju je spojen uređaj te tako osigurati brzo osvježavanje i povlačenje informacija iz baze podataka. Aplikaciju je također potrebno prilagoditi za druge mobilne operacijske sustavne kao što su *iOS* i *Windows Phone* kako bi se povećala fleksibilnost aplikacije i kako bi se omogućilo većem broju korisnika nesmetano korištenje.

8. Literatura

[1] en.wikipedia.org/wiki/Android_(operating_system)
[2] en.wikipedia.org/wiki/PHP
[3] en.wikipedia.org/wiki/Android_Studio
[4] https://en.wikipedia.org/wiki/JSON
[5] https://en.wikipedia.org/wiki/Java_(programming_language)
[6] https://code.tutsplus.com/tutorials/android-sdk-create-a-barcode-reader--mobile-17162
[7] http://www.tutorialspoint.com/articles/run-a-php-program-in-xampp-server
[8] https://netbeans.org/kb/docs/php/configure-php-environment-windows.html
[9] http://www.dreamincode.net/forums/topic/304002-connecting-apache-server-with-android
[10] https://www.tutorialspoint.com/android/android_sqlite_database.htm
[11] https://stackoverflow.com/questions/24844514/database-in-android-studio
[12] Sams Teach Yourself Java in 24 Hours, Mark Taber

9. Pojmovnik

Pojam	Kratko objašnjenje	Više informacija potražite na
Android	Operativni sustav velikog broja mobilnih telefona ali i drugih uređaja	en.wikipedia.org/wiki/Android_(operating_system)
PHP	Skriptni jezik korišten za razvoj poslužiteljske strane sustava	en.wikipedia.org/wiki/PHP
Android Studio	Razvojni sustav za android aplikacije	en.wikipedia.org/wiki/Android_Studio
iOS	Operativni sustav za Apple mobilne uređaje	en.wikipedia.org/wiki/IOS
Windows Phone	Operativni sustav za Windows mobilne uređaje	en.wikipedia.org/wiki/Windows_Phone
JSON	Format datoteke korištenje za razmjenu podataka između poslužitelja i klijenta	https://en.wikipedia.org/wiki/JSON
Java	Vrsta objektno orijentiranog programskog jezika	https://en.wikipedia.org/wiki/Java_(programming_language)