PANEVROPSKI UNIVERZITET APEIRON

FAKULTET INFORMACIONIH TEHNOLOGIJA

BANJA LUKA

Seminarski rad

KONVERTOR VALUTA U PROGRAMSKOM JEZIKU "JAVA"

Nastavni predmet: ***Profesionalna praksa***

Predmetni nastavnik:
Prof. dr Željko Stanković

Student:
Siniša Božić
192-20/RITP

Banja Luka, 2021.

SADRŽAJ

UVOD

Predmet "Profesionalna praksa" predviđa praktično angažovanje studenta u nekoj od firmi iz oblasti koju student izučava. Budući da je školsku 2020/2021 godinu obilježila nepovoljna epidemiološka situacija, preduzeća koja su inače primala praktikante to sada nerado čine. Iz tog razloga, a kao student smjera Programiranje i softversko inženjerstvo autor se odlučio na samostalno izučavanje programskog jezika Java (koji se inače proučava na 3. godini), uz korištenje raznih resursa, od mrežnih upustava, primjera kôdova, video lekcija, knjiga i sl. Od ranije je poznavao osnove programskog jezika Python koji ipak ne slijedi C sintaksu i mnoga druga obilježja C grupe jezika kao što to Java čini, pa je bio potreban određen period privikavanja.

Obzirom da ovaj predmet predviđa umjesto prakse izradu elaborata u vidu seminarskog rada, iskorištena je mogućnost da se spoji ugodno sa korisnim i u vidu akademskog rada obradi jedan jednostavan program kao što je to konverter valuta. Znanja koja su potrebna da se ovakav, početnički, projekat uspješno izvrši su:

- poznavanje osnova sintakse i struktura podataka u Javi,
- poželjno korištenje razvojnog korisničkog okruženja,
- mogućnost povezivanja raznih dijelova kôda, njihovo kombinovanje, odnosno iskorištavanje mogućnosti eksternih klasa,
- dobavljanje podataka iz eksternih izvora,
- sposobnost razumijevanja, čitanja i kombinovanja kôda.

Program koristi konzolu za ulaz i izlaz podataka, što možda nije vizuelno atraktivno, ali bi sa stanovišta učenja jezika trebalo da bude zadovoljavajuće, jer je od toga implementacija grafičkog interfejsa udaljena samo jedan dodatni korak, osim toga, vodeći računa o obimu rada, objašnjavanje nekog složenijeg kôda bi vrlo lako i uduplalo sam rad što bi vjerovatno izlazilo iz nekih uobičajenih granica obima jednog seminarskog rada.

Kako je napomenuto u samom radu, program će da funkcioniše ispravno dok je aktivna pretplata na servis Exchangeratesapi.io, a to će biti čitavog Juna mjeseca ove, 2021. godine. To je sasvim dovoljno vrijeme za evaluaciju ovog rada a i nakon isteka tog roka iz prezentovanih snimaka ekrana (tzv. screenshotova) se može vidjeti kako program koristi podatke dohvaćene sa Internet servisa te ispisuje izlazne vrijednosti.

1 STRUKTURE I LOGIKA KORIŠTENI U RADU

1.1 BufferedReader, Scanner (KORISNIČKI ULAZ)

Gotovo svaki program uzima ulazne vrijednosti od korisnika, bilo putem periferije (tastatura, miš) ili putem ulaznih podataka (fajlova). Dva glavna načina importovanja korisničkog ulaza u Javi su putem korištenja mogućnosti klasa `BufferedReader`[1] te `Scanner`[2]. Za potrebe ovog rada će se objašnjavanje mogućnosti ovih klasa pojednostaviti i odnositi isključivo na unos korisnika u konzolu putem tastature, te, unos valutnog kursa sa URL adrese, odnosno od provajdera kursnih podataka. Na idućem primjeru su navedena dva jednostavna unosa podataka tipa `String` i `int` pozivanjem ove dvije klase:

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
String name = reader.readLine();
String sAge = reader.readLine();
int nAge = Integer.parseInt(sAge);
```

odnosno

```
Scanner scanner = new Scanner(System.in);
String name = scanner.nextLine();
int age = scanner.nextInt();
```

U prvom primjeru korištenjem naredbi klase `BufferedReader` smo kreirali novi objekat `reader` koji je instanca navedene klase[3]. Nakon toga su deklarisane dvije varijable tipa `String`, jedna za iščitavanje podataka tog tipa iz konzole, a druga za brojeve. Obadvije varijable koriste objekat `reader` sa metodom `readLine()` a zadnja linija kôda konvertuje varijablu `sAge` u numeričku vrijednost budući da `BufferedReader` ne podržava direktno iščitavanje numeričkih vrijednosti. Na taj način smo snimili korisnički ulaz u dvije varijable kojima možemo dalje manipulisati.

Drugi primjer koristi klasu `Scanner` koja u ovom primjeru radi istu stvar, na vrlo sličan način, samo su komande nešto drugačije. Stvara se novi objekat `scanner` te se deklarišu dvije varijable, jedna tipa `String`, druga `int` (budući da `Scanner` može direktno da iščitava brojeve pa nema potrebe za konverzijom), koristeći metode, respektivno,

[1]Za detaljan opis pogledati Oracle Java Documentation:
https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/BufferedReader.html
[2]Za detaljan opis pogledati Oracle Java Documentation:
https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/util/Scanner.html
[3]Zapravo smo stvorili novi objekat koji je stvorio novi objekat druge, klase `InputStreamReader` uzevši kao parametar `System.in` koji je podrazumijevani način za unos u konzolu u Javi.

`nextLine()` odnosno `nextInt()`. Krajnji rezultat je isti[4], imamo dvije varijable čiji sadržaj je memorisani korisnički ulaz.

1.2 DECIMALFORMAT (FORMATIRANJE ISPISA)

Često je potrebno formatirati decimalne brojeve tako da sadrže određen broj decimalnih mjesta, najčešće dva. Budući da su decimalni brojevi u Javi podaci tipa `double` ili `float`, oni se po podrazumijevanim podešavanjima prikazuju u višestrukim decimalnim mjestima (6 i više) . Postoji više načina kako da se kontroliše broj ispisanih decimala a prikazani metod je vjerovatno najkorišteniji. `DecimalFormat` predstavlja klasu u Javi[5] koja sadrži metode kojima je moguće prilagoditi ispis numeričkih varijabli putem unaprijed predefinisanih načina. Importovanjem te klase omogućava se njeno korištenje a za naše potrebe navešćemo četiri operatora koji se najčešće koriste:

`0`	Cifra - uvijek prikazana, čak iako broj ima manji broj cifara (tada se ispisuje 0)
`#`	Cifra, vodeće nule se ne prikazuju.
`.`	Decimalni separator[6]
`,`	Separator hiljada

U slijedećem primjeru ćemo prikazati jednostavan prikaz broja formatiranog na različite načine:

```
import java.text.DecimalFormat;
public class DecimalFormat {
    public static void main(String[] args) {

    //formatiranje brojeva sa prikazom dva decimalna mjesta
DecimalFormat df = new DecimalFormat("0.00");
System.out.println(df.format(123456.788888));
System.out.println(df.format(12345678.999999));

    //formatiranje brojeva sa odvajanjem hiljada i prikazom
    //dva decimalna mjesta
        df = new DecimalFormat("###,##0.00");
        System.out.println(df.format(123456.788888));
        System.out.println(df.format(1234567.899999));

   //formatiranje brojeva sa odvajanjem hiljada i
   //prikazom tri decimalna mjesta
```

[4]Iako su ove klase slične, postoje određene razlike između njih, za detalje pogledati https://www.java67.com/2016/06/5-difference-between-bufferedreader-and-scanner-in-java.html

[5]Za detaljan opis pogledati Oracle Java Documentation: https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/text/DecimalFormat.html

[6]Budući da je Java nastala u SAD, gdje se koriste decimalni i operatori za hiljade na obrnuti način kao u kontinentalnoj Evropi, podrazumijevani decimalni operator je . a operator hiljada , . U programskom kôdu je moguće definisati i lokalna podešavanja za svaku zemlju ponaosob.

```
        df = new DecimalFormat("###,##0.000");
        System.out.println(df.format(123456.788888));
        System.out.println(df.format(12345678.899999));
    }
}
```

Ispis programa:

```
123456,79
12345679,00
123.456,79
1.234.567,90
123.456,789
12.345.678,900
```

Budući da će nam za konverter valuta biti potreban ispis novčanih iznosa sa dva decimalna mjesta, ovim smo pokazali kako to možemo praktično da ostvarimo.

1.3 HASHMAP

HashMap je takva struktura[7] koja omogućava skladištenje podataka u parovima. Svaki par ima svoj ključ, koji je jedinstven, i njemu pridruženu vrijednost. Na primjer, ako bi htjeli skladištiti ovako uređen set podataka:

Grana nauke: disciplina		**Šifra proizvoda: naziv artikla**
Pravne nauke: Međunarodno pravo Pravne nauke: Obligaciono pravo Medicina: Anatomija Medicina: Molekularna biologija Informatika: Računarske mreže Informatika: Sistemski softver	ili	6846865: Aktuator turbine 9684684: Servo pumpa 7485664: Filter motornog ulja 8768468: Zadnji izduvni lonac

Tabela 1: Primjer tipa podataka pogodnog za skladištenje u HashMap

niti niz, niti ArrayList ne bi bili pogodni, jer oni čuvaju samo elemente istog tipa, koji nisu povezani jedni sa drugima. Iz tog razloga se HashMap (klasa u Javi, a u drugim programskim jezicima poznat kao Dictionary) naziva i asocijativni niz, zato što jedinstvenom ključu asocira neku vrijednost. Kao i kod nizova i ArrayList, i ključ i vrijednost u HashMap mogu biti bilo primitivne varijable[8], bilo objekti.

Sintaksa za kreiranje HashMap glasi:

[7]Za detaljan opis pogledati Oracle Java Documentation:
https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/util/HashMap.html
[8]Pogledati detaljan spisak primitivnih vrijednosti:
https://docs.oracle.com/en/java/javase/15/docs/api/jdk.jdi/com/sun/jdi/PrimitiveValue.html

```
HashMap<tip ključa, tip vrijednosti> primjerHashMap = new HashMap<>();
```

Jednostavan primjer deklarisanja i upisa u HashMap slijedi:

```
import java.util.HashMap;
public class Main {
public static void main(String[] args)  {

 HashMap<Integer[9], String> licniDokumenti = new HashMap<>();
        licniDokumenti.put (212133, "Jovan Jovanovic");
        licniDokumenti.put (162348, "Marko Markovic");
        licniDokumenti.put(8082771, "Marijana Marijanovic");
        System.out.println(licniDokumenti);
        }
     }
```

Ispis:

```
{212133=Jovan Jovanovic, 8082771=Marijana Marijanovic,
162348=Marko Markovic}
```

Ukoliko bi pokušali dodati novu vrijednost koristeći već korišteni ključ, ta bi vrijednost bila dodijeljena tom ključu a stara prepisana:

```
licniDokumenti.put (212133, "Dragan Draganovic");
```

Ako želimo ispisati određenu vrijednost, pristupamo joj putem ključa:

```
System.out.println(licniDokumenti.get(212133));
```

Kao što je vidljivo iz dokumentacije, HashMap sadrži još mnoštvo metoda kojima se može manipulisati sadržajem ove strukture. Još ćemo pomenuti način iteracije[10] koji je nešto komplikovaniji:

```
for (Map.Entry<Integer, String> entry:
licniDokumenti.entrySet()) {
   System.out.println(entry);
}
```

Da bi iterisali HashMap moramo koristiti `Map.Entry` klasu, definisati varijablu i koristiti `entrySet()` metod, jer se HashMap sastoji od parova ključ i vrijednost definisanih kao `Map.Entry` parovi, tako da se iteriraju parovi a ne njihovi ključevi i vrijednosti.

[9]Kolekcije kao što je HashMap za tip varijable ne koriste primitivne varijable već tzv. generics.

[10]Iako se HashMap može ispisati korištenjem standardne System.out.println metode (za razliku od nizova i ArrayList), iteracija je podrazumijevani način kako ispisa tako i drugih operacija kojima se sadržaj ove strukture na neki način mijenja.

1.4 HttpURLConnection, java.net.url (URL KONEKCIJA)

Za potrebe programa konverter valuta će biti potrebno dobaviti aktuelne vrijednosti valutnih kurseva, što će se izvesti konekcijom na određeni URL. URL predstavlja akronim od Uniform Resource Locator, u žargonu se koristi naziv "web adresa", i to je način kome se pristupa nekom od resursa na WWW (World Wide Web). Budući da upravo treba da pristupimo nekom od servisa koji nude podatke o valutnim kursevima, pristupaćemo nekom URL. U Javi se URL tretira kao podatak tipa String, a klasa koja manipuliše ovim resursom je `java.net.URL`[11] koju je potrebno importovati na početku programskog kôda. Da bi mogli da razumijemo korištenje URL, treba poznavati njegove komponente, a to su:

1. Protokol (HTTP odnosno HTTPS[12]),
2. Ime hosta, odnosno naziv servera koji skladišti sadržaj,
3. Ime fajla, odnosno putanja do konkretnog dokumenta, te,
4. Broj porta (opciono).

Na primjeru iduće WWW stranice vidimo njene komponente:
https://eregistar.banjaluka.rs.ba/wp-content/uploads/2021/01/GUBL-OB-02-02.pdf
Protokol, ime hosta, putanja do dokumenta.

Da bismo dalje procesuirali željeni URL koji smo prethodno definisali, potrebno je iščitati podatke iz njega. Java ima ugrađenu klasu za to naziva HttpURLConnection[13][14] i ona omogućava izvršavanje osnovnih HTTP upita bez korištenja eksternih dodataka, odnosno, sve je već ugrađeno u klasu HttpURLConnection koja je dio java.net nad-klase. Za potrebe ovog rada ćemo koristiti jednostavnu sintaksu koja dohvata podatke sa datog URL što se može predstaviti idućim primjerom[15]:

```
URL url = new URL("http://example.com");
HttpURLConnection con = (HttpURLConnection) url.openConnection();
con.setRequestMethod("GET");
```

[11]Za detaljan opis pogledati Oracle Java Documentation:
https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/net/URL.html
[12]Hypertext Transfer Protocol (Secure), je okosnica WWW i predstavlja klijent-server protokol za razmjenu podataka te njihovo formatiranje i prikaz.
[13]Za detaljan opis pogledati Oracle Java Documentation:
https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/net/HttpURLConnection.html
[14]U slučaju da pristupamo stranici koja koristi HTTPS protokol, naziv odgovarajuće klase je HttpsURLConnection.
[15] U nastavku rada nećemo navoditi kompletan kôd i `main` metod, već samo konkretan kôd klasa i metoda koji se objašnjavaju.

Navedeni kôd deklariše objekat `url` i inicijalizuje ga definisanom URL adresom. Potom se deklariše objekat `con` koji pripada klasi HttpURLConnection i poziva `openConnection()` metod, potom objekat `con` poziva `setRequestMethod()` koji može imati iduće parametre: `GET`, `POST`, `HEAD`, `OPTIONS`, `PUT`, `DELETE`, `TRACE`. Ovdje smo samo kreirali objekat koji je sposoban da primi podatke od željenog URL, ne i samu konekciju. Da bismo pročitali sadržaj sa URL stranice kojoj pristupamo, potrebno je da deklarišemo varijablu kojoj ćemo sadržaj inicijalizovati:

```
int status=con.getResponseCode();
```

Metod `getResponseCode()` vraća iduće kôdove:

`1xx:` Informacija

`2xx:` Uspjeh (HTTP_OK je identično kôdu 200)

`3xx:` Redirekcija

`4xx:` Klijent greška

`5xx:` Server greška

Rečeno u pseudo-kodu, ako je zahtjev uspješan, onda možemo da ga iskoristimo i deklarišemo varijablu inicijalizovanu sa dohvaćenim podacima, a ako ne, možemo da npr. obavjestimo korisnika o tome. Koristeći navedeni primjer, to se dalje u programskom kôdu može ispisati kao:

```
Reader streamReader = null;

if (status > 299) { //ako je status neuspješan
    streamReader = new
InputStreamReader(con.getErrorStream());
} else { // ako je status uspješan
    streamReader = new InputStreamReader(con.getInputStream());
}
```

Metod `getErrorStream()` dohvata eventualnu grešku koja se može iskoristiti radi debagovanja. U slučaju da je zahtjev uspješan, odnosno da nema greške, rezultat se smješta u varijablu `streamReader` koja se može dalje ispisati putem `BufferedReader` ili dalje manipulisati.

1.5 JSON (PROCESUIRANJE STRUKTURIRANIH PODATAKA)

Budući da ćemo podatke o valutnim kursevima dobiti u formatu JSON, da bismo uspješno procesuirali ovaj format potrebno je da imamo okvirno razumijevanje šta je to i kako ga importovati u programski kôd. JSON (JavaScript Object Notation) je vrlo popularan i čest tip podataka koje generišu serveri i naročitu primjenu nalazi u bazama podataka, to su zapravo tekstualni fajlovi sa određenom strukturom koja je slična HashMap u Javi, odnosno to je neuređena kolekcija parova koji sadrže ime i vrijednost[16]. Na primjer, mi ćemo od provajdera valutnih kurseva dobiti idući strukturisani podatak u JSON formatu (npr. bazna valuta euro (EUR), konverzija u dinare (RSD)):

```
{"success":true,"timestamp":1622016423,"base":"EUR","date":"20
21-05-26","rates":{"RSD":117.587941}}
```

Za potrebe ovog rada ćemo koristiti eksternu klasu JSON in Java[17], to je nezavisni projekat odnosno klasa koji se može koristiti u našem primjeru na idući način:

```
JSONObject obj=new JSONObject(response.toString());
Double kurs=obj.getJSONObject("rates").getDouble(valuta);
```

Kao i uvijek, kreiramo novu instancu klase `JSONObject`, objekat `obj` koji uzima vrijednost parametra varijablu `response` (kako ćemo kasnije vidjeti, ta varijabla sadrži odgovor od servera kada smo povukli podatke od provajdera valutnih kurseva) i tu varijablu treba konvertovati u tip podatka `String` kako uputstva klase `JSONObject` nalažu. Potom deklarišemo varijablu `kurs`[18] koja koristi, odnosno inicijalizuje se sa vrijednošću prethodno stvorenog objekta `obj` i koristeći metod `getJSONObject()` iščitava podatke iz para JSON fajla koji počinje imenom `"rates"` odnosno riječ je o ovom redu: `"rates":{"RSD":117.587941}`. Kôdom `getDouble(valuta)` se dalje procesuira JSON fajl, koji sadrži ime valute čiji kurs treba da dobijemo, ova kompletna linija kôda

```
Double kurs=obj.getJSONObject("rates").getDouble(valuta);
```

bi se pseudo-kodom mogla objasniti kao: kreiraj varijablu tipa `double` koja može da skladišti objekte tog tipa, iščitaj JSON fajl iz para podataka koji počinje sa `"rates"` te

[16]Za detaljan opis pogledati zvaničnu dokumentaciju:
https://www.json.org/json-en.html
[17]Zvanični sajt: https://github.com/stleary/JSON-java
[18]Varijabla je tipa `Double` a ne `double` iz razloga što primitivne varijable ne mogu sadržavati objekte pa je potrebno koristiti tzv. Wrapper-Class, pogledati: https://www.w3schools.com/java/java_wrapper_classes.asp

koristi ime valute da bi došao do vrijednosti kursa, i tu vrijednost kursa memoriši tj. inicijalizuj. Na ovaj način dobijamo vrijednost kursa kao decimalni broj koji dalje možemo koristiti u programu.

1.6. "WHILE" PETLJA

"While" petlja se u programskom Jeziku java deklariše kao[19]:

```
while (uslov)
{
neki kôd;
}
```

Pri tome, uslov je logički iskaz koji može biti `tačan` ili `netačan`. Dok je uslov `tačan`, petlja se izvršava, kad uslov postane `netačan`, petlja prestaje.

Idući primjer pokazuje kako se jednostavno mogu ispisati slova abecede korištenjem ove petlje.

```
// primjer "while" petlje
class WhilePrimjer {
public static void main(String args[]) {
    char karakter;
    karakter= 'a';
        while (karakter<'z') {
            System.out.print(karakter);
            karakter++;
          }
     }
}
```

Napomena: definisana je varijabla tipa `char` (koja se inicijalizuje jednostrukim navodnim znacima). Petlja je tako inicijalizovana da je varijabla karakter, početnog stanja `'a'`, manja od krajnjeg karaktera abecede, odnosno `'z'`. Sve dok je taj uslov tačan, petlja vrši iteraciju i ispisuje znak po znak. Nakon prve iteracije, varijabla karakter se uvećava za jedan, odnosno sa karaktera `'a'` se povećava na `'b'` i tako redom.

Suštinski, petlje "for" i "while" predstavljaju jednu te istu stvar iskazanu na različite načine. Petlja "for" omogućava i zahtjeva deklarisanje i inicijalizaciju varijable tokom deklarisanja same petlje, dok kod "while" petlje se deklarisanje i inicijalizacija kontrolne varijable obavlja prije same petlje. Obadvije petlje koriste logički iskaz za verifikaciju iteracije, dok se brojač u

[19]Za detaljan opis pogledati Oracle Java Documentation:
https://docs.oracle.com/javase/specs/jls/se15/html/jls-14.html#jls-14.12

"for" petlji također obično deklariše tokom deklarisanja same petlje[20], dok se u "while" petlji brojač deklariše na kraju izvršenja kôda unutar bloka petlje.

Podvarijanta "while" petlje je "do-while" petlja, čija je osobenost da će se izvršiti barem jednom, bez obzira na istinitost logičkog uslova.

Sumarizovano dajemo pregled ove tri petlje:

Poređenje	**For petlja**	**While petlja**	**Do while petlja**
osnovno	kontrola toka programa koja iteriše blok programa na osnovu logičkog iskaza	kontrola toka programa koja iteriše blok programa na osnovu logičkog iskaza	Kontrola toka programa koja izvršava dio programa bar jednom, bez obzira na stanje logičkog iskaza, a dalji dok izvršavanja (iteracije) zavisi od logičkog iskaza
korištenje	Ako je broj iteracija poznat	Kada je broj iteracija nepoznat ili nije fiksno zadat	Ako se ne zna tačan broj iteracija a postoji potreba da se dio programa izvrši bar jednom
sintaksa	`For(inicijalizacija ; uslov; iteracija) { neki kôd }`	`While (uslov) { neki kôd }`	`Do { neki kôd } while (uslov);`
primjer	`For (int i=1; i<=10; i++) { System.out.println( i); }`	`Int i=1; while (i<=10) { System.out.println(i ); i++; }`	`Int i=1; do { System.out.println(i) ; i++; while (i<=10) { }`

Tabela 2: Sumarni pregled karakteristika petlji u programskom jeziku Java

[20]"for" petlja se može definisati i kao:

```
for (int i=0; i <10) {
neki kôd;
i++
}
```

2 PROGRAM: KONVERTER VALUTA

Konverter valuta je program koji u konzoli, dakle tekstualnim putem, omogućava korisnički izbor jedne od pet predefinisanih novčanih valuta, odabir valute u koju će se konverzija izvršiti, te unos sume za konverziju. Prema važećim valutnim kursevima program vrši preračun i ispisuje krajnji iznos u željenoj valuti. Nakon konverzije i ispisa moguće je ponoviti konverziju, sve dok korisnik izričito ne prekine program.

Naravno, program može biti jednostavniji, na način da smo umjesto pretraživanja i korištenja aktuelnih valutnih kurseva jednostavno zatražili od korisnika da sam unese kurseve, a može biti i komplikovaniji, na primjer mogli smo omogućiti konverziju iz i u više valuta[21], dodati logiku za hvatanje greški[22] (try/catch), umjesto konzole koristiti GUI (grafički interfejs sa prozorima) itd., no za potrebe ovog rada smo odabrali ovu varijantu koja nije ni preduga ni prekratka što se objašnjavanja tiče. Također autor pretpostavlja poznavanje osnova Jave pa će se objašnjavati samo način funkcionisanja programa odnosno kako je isti strukturiran.

Počnimo sa ovim kôdom:

```
import java.util.HashMap;
import java.util.Scanner;

public class KonverterValuta {

    public static void main(String[] args){

     HashMap<Integer, String> vrijednostiValuta = new
HashMap<>();
    vrijednostiValuta.put(1, "USD");
    vrijednostiValuta.put(2, "EUR");
    vrijednostiValuta.put(3, "BAM");
    vrijednostiValuta.put(4, "RSD");
    vrijednostiValuta.put(5, "HRK");

     String fromCode, toCode;
     double iznos;

     Scanner sc=new Scanner(System.in);

     System.out.println("Ovo je konvertor valuta,
dobrodošli!");
```

[21]To se može postići na način da se u listu stave sve valute koje provajder koristi, te sa "if" petljom izvršiti selekciju valute koju je korisnik unio u konzolu.

[22]Npr. ukoliko korisnik umjesto brojeva unese slova, program će se prekinuti obzirom da je tip podatka koji program prihvata broj, odnosno `int`. Ovakve greške se mogu "hvatati" dodavanjem tzv. Exceptions.

```
    System.out.println("Za koju valutu želite konverziju?");
    System.out.println("1:USD (US Dollar)\t 2:EUR (Euro) \t
3:BAM (Bosanska Marka) \t 4:RSD (Srpski Dinar)\t 5:HRK
(Hrvatska Kuna)");
     fromCode=vrijednostiValuta.get(sc.nextInt());

     System.out.println("U koju valutu konvertujete?");
     System.out.println("1:USD (US Dollar)\t 2:EUR (Euro) \t
3:BAM (Bosanska Marka) \t 4:RSD (Srpski Dinar)\t 5:HRK
(Hrvatska Kuna)");
     toCode=vrijednostiValuta.get(sc.nextInt());

     System.out.println("Unesite željeni iznos za konverziju:
");
    iznos = sc.nextDouble();

     //još samo fali izvor za kurseve valuta

     System.out.println("Hvala Vam za korištenje konvertora
valuta!");
     }
}
```

Ispis:

```
Za koju valutu želite konverziju?
1:USD (US Dollar)    2:EUR (Euro)   3:BAM (Bosanska Marka)
4:RSD (Srpski Dinar)      5:HRK (Hrvatska Kuna)
1 (pretpostavljeni unos korisnika)
U koju valutu konvertujete?
1:USD (US Dollar)    2:EUR (Euro)   3:BAM (Bosanska Marka)
4:RSD (Srpski Dinar)      5:HRK (Hrvatska Kuna)
2 (pretpostavljeni unos korisnika)
Unesite željeni iznos za konverziju:
1 (pretpostavljeni unos korisnika)
Hvala Vam za korištenje konvertora valuta!
```

U `main` metodu smo deklarisali HashMap naziva vrijednosti `Valuta`, njemu smo dodijelili parove sa brojevima koje predstavljaju valute, te valutama koje ćemo u programu koristiti. Potom smo deklarisali dve varijable tipa `String`, `fromCode` (koristićemo za obilježavanje valute iz koje vršimo konverziju) te `toCode` (u koju se valutu vrši konverzija) te varijablu tipa `double` iznos (za konverziju).

Potom definišemo objekat `Scanner` koji nam treba za uzimanje korisničkog unosa. Ispod njega ispisujemo redove koji objašnjavaju korisniku šta program radi i šta se očekuje od njega da u program unese. Linija:

```
fromCode=vrijednostiValuta.get(sc.nextInt());
```

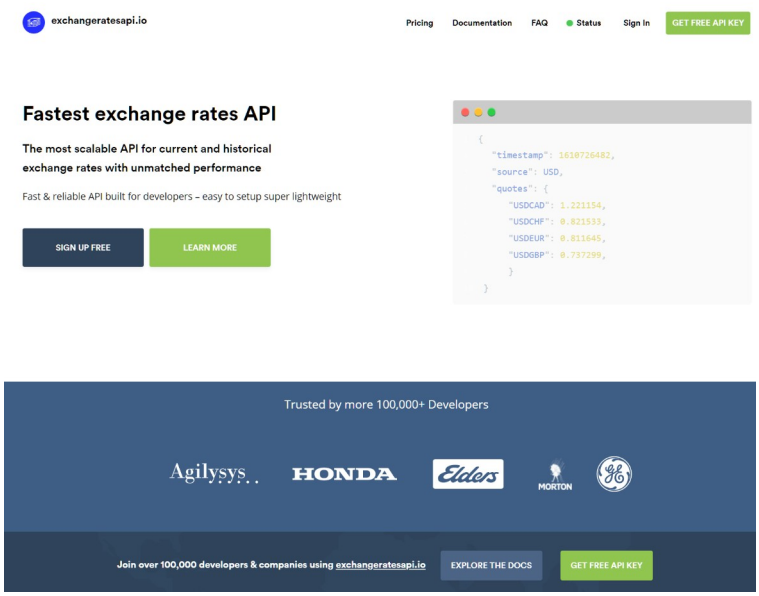
smješta u varijablu `fromCode` onu oznaku valute koju korisnik unese u konzolu - budući da su valute predefinisane u `HashMap` kao parovi brojeva i njima pridruženih oznaka valuta, ako korisnik unese npr. broj `1` to se odnosi na valutu USD. Dalje program ispisuje obavještenja u koje se valute želi konverzija izvršiti i traži od korisnika unos. Potom varijabla `toCode` analogno varijabli `fromCode` uzima odgovarajuću oznaku valute.

Još se od korisnika traži unos iznosa za konverziju i program završava prigodnim obavještenjem.

Program funkcioniše i u ovom momentu još samo trebamo da dodamo logiku za uzimanje vrijednosti valutnih kurseva.

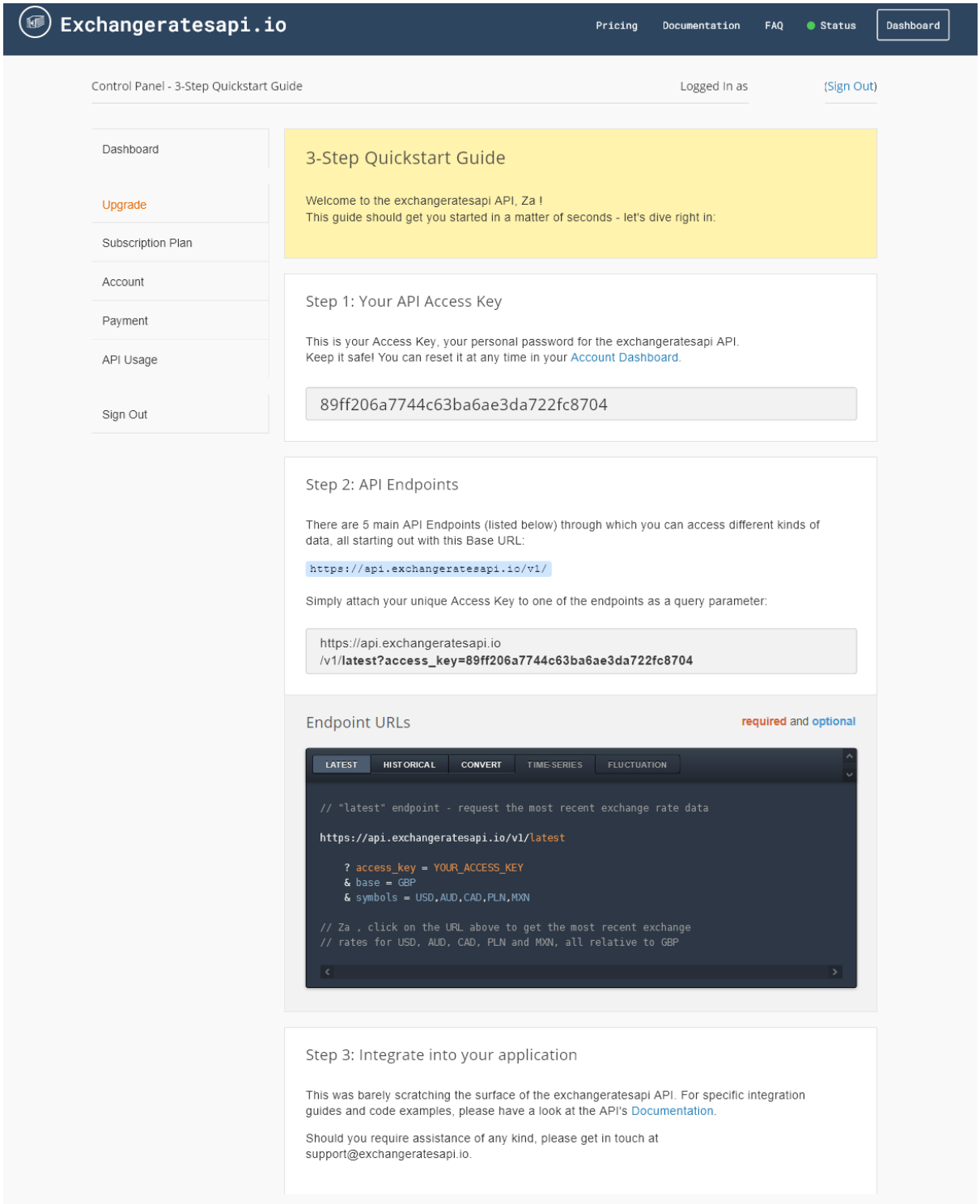
2.1 PROVAJDER PODATAKA Exchangeratesapi.io

Da bismo dobili vrijednosti valutnih kurseva moramo koristiti neki od Internet servisa. Autor nije uspio naći servis koji bi bio besplatan a koji bi se mogao tako podesiti da odgovara potrebama ovog rada i programa[23], tako da je iskorišten servis Exchangeratesapi.io gdje je kupljen mjesečni paket:



Prilog 1:Exchangeratesapi.io

[23]Ili bar ne na zadovoljavajući način u smislu ovog rada i programa – velika većina servisa nudi osnovni dio besplatan, a za svake dodatne opcije se mora izvršiti kupovina paketa.



Prilog 2: Podešavanja plaćenog paketa i uputstva za importovanje

Napomena: za potrebe ovog rada, dok traje pretplata (mjesec dana), dotle će da servis funkcioniše i vraća vrijednosti valutnih kurseva, kad pretplata istekne, program će vraćati obavještenje da je došlo do greške prilikom dohvatanja podataka.

Kao što se iz Priloga 2 vidi, servisu se pristupa putem slijedećeg linka (link uključuje ključ koji označava da je korisnik platio premium paket):

https://api.exchangeratesapi.io/v1/latest?access_key=89ff206a7744c63ba6ae3da722fc8704

Kada se ovom linku pristupi, dobijamo iduće informacije:


```
{"success":true,"timestamp":1622098923,"base":"EUR","date":"2021-05-
27","rates":
{"AED":4.481561,"AFN":96.544464,"ALL":123.316464,"AMD":637.713438,"ANG":2.1
99225,"AOA":784.652736,"ARS":115.299554,"AUD":1.574386,"AWG":2.196126,"AZN"
:2.077972,"BAM":1.957852,"BBD":2.473737,"BDT":103.765575,"BGN":1.954256,"BH
D":0.459977,"BIF":2421.550193,"BMD":1.22007,"BND":1.620654,"BOB":8.447025,"
BRL":6.481129,"BSD":1.225156,"BTC":3.185343e-
5,"BTN":89.057553,"BWP":13.020155,"BYN":3.075641,"BYR":23913.375019,"BZD":2
.469623,"CAD":1.477865,"CDF":2445.020627,"CHF":1.094793,"CLF":0.032278,"CLP
":890.650927,"CNY":7.780999,"COP":4566.722587,"CRC":757.085085,"CUC":1.2200
7,"CUP":32.331859,"CVE":110.3781,"CZK":25.433337,"DJF":218.111366,"DKK":7.4
35842,"DOP":69.650947,"DZD":162.795421,"EGP":19.120918,"ERN":18.30348,"ETB"
:52.65184,"EUR":1,"FJD":2.471801,"FKP":0.868098,"GBP":0.864487,"GEL":4.0017
21,"GGP":0.868098,"GHS":7.081014,"GIP":0.868098,"GMD":62.406049,"GNF":12032
.889228,"GTQ":9.457691,"GYD":256.304725,"HKD":9.469148,"HNL":29.444033,"HRK
":7.506845,"HTG":110.704786,"HUF":349.153572,"IDR":17420.100656,"ILS":3.965
836,"IMP":0.868098,"INR":88.610091,"IQD":1787.44547,"IRR":51371.053915,"ISK
":147.884998,"JEP":0.868098,"JMD":182.827571,"JOD":0.865042,"JPY":133.13710
4,"KES":131.218391,"KGS":101.281933,"KHR":4990.099444,"KMF":492.145803,"KPW
":1098.063282,"KRW":1361.958181,"KWD":0.367021,"KYD":1.020971,"KZT":524.480
309,"LAK":11562.094845,"LBP":1847.54453,"LKR":243.197434,"LRD":209.360891,"
LSL":16.800374,"LTL":3.602551,"LVL":0.738009,"LYD":5.449731,"MAD":10.798353
,"MDL":21.557156,"MGA":4606.077177,"MKD":61.573597,"MMK":2016.655972,"MNT":
3478.31398,"MOP":9.794717,"MRO":435.564835,"MUR":49.290009,"MVR":18.850471,
"MWK":977.082255,"MXN":24.214981,"MYR":5.051701,"MZN":73.764796,"NAD":16.80
0238,"NGN":503.510465,"NIO":42.788298,"NOK":10.213421,"NPR":142.476809,"NZD
":1.671606,"OMR":0.46974,"PAB":1.225165,"PEN":4.684138,"PGK":4.356914,"PHP"
:58.563394,"PKR":190.179456,"PLN":4.49846,"PYG":8227.704943,"QAR":4.442247,
"RON":4.915783,"RSD":117.726594,"RUB":89.662345,"RWF":1227.343173,"SAR":4.5
75676,"SBD":9.725588,"SCR":20.131323,"SDG":503.279335,"SEK":10.13838,"SGD":
1.614519,"SHP":0.868098,"SLL":12505.718852,"SOS":713.7409,"SRD":17.268849,"
STD":25299.111799,"SVC":10.720248,"SYP":1534.281248,"SZL":16.932638,"THB":3
8.126579,"TJS":13.972117,"TMT":4.282446,"TND":3.322861,"TOP":2.742657,"TRY"
:10.309568,"TTD":8.331412,"TWD":33.917337,"TZS":2829.342165,"UAH":33.741501
,"UGX":4346.589391,"USD":1.22007,"UYU":53.885214,"UZS":12998.048548,"VEF":2
60887872305.10638,"VND":28128.717401,"VUV":132.264409,"WST":3.069635,"XAF":
656.681005,"XAG":0.043998,"XAU":0.000642,"XCD":3.2973,"XDR":0.847234,"XOF":
656.683699,"XPF":119.802222,"YER":305.017638,"ZAR":16.740046,"ZMK":10982.07
5865,"ZMW":27.557496,"ZWL":392.862961}}
```


Odnosno vidimo da servis isporučuje valutne kurseve za sve valute koje podržava, sa baznom valutom EUR. Nama treba takvo podešavanje da možemo izabirati baznu i valutu u koju vršimo konverziju. Iz dokumentacije je vidljivo da se pomenuti link prilagođava dodacima, na način da dodatak `"&base="` označava odabir bazne, a `"&symbols="` valute u koju se vrši konverzija, prema tome ako formatiramo link na npr. ovaj način:

```
https://api.exchangeratesapi.io/v1/latest?
access_key=89ff206a7744c63ba6ae3da722fc8704&base=RSD&symbols=U
SD
```

dobićemo konverziju iz EUR u RSD, odnosno

```
{"success":true,"timestamp":1622098923,"base":"RSD","date":"20
21-05-27","rates":{"USD":0.010364}}
```

Kada sada povežemo ranije rečeno o načinima dobijanja informacija sa WWW, možemo napisati iduće:

```
String GET_URL = "https://api.exchangeratesapi.io/v1/latest?
access_key=89ff206a7744c63ba6ae3da722fc8704&base=" + toCode +
"&symbols=" + fromCode;
        URL url = new URL(GET_URL);
        HttpURLConnection httpURLConnection =
(HttpURLConnection) url.openConnection();
        httpURLConnection.setRequestMethod("GET");
        int responseCode =
httpURLConnection.getResponseCode();
```

Deklarisali smo varijablu `GET_URL` koja skladišti link ka provajderu valutnih informacija, na način da smo konkantenacijom razlomili kôd tako da dijelove kojima se biraju valute pristupamo pomoću varijabli `toCode` i `fromCode`. Time ćemo dobiti mogućnost da korisnikov ulaz iskoristimo kako bi modifikovali link i dobili željene konverzije. Dalje smo iskoristili klase `java.net.url` i `HttpURLConnection` kako bi otvorili konekciju ka datom servisu.

```
        if (responseCode == HttpURLConnection.HTTP_OK)
{//success
            BufferedReader in = new BufferedReader((new
InputStreamReader(httpURLConnection.getInputStream())));
            String inputLine;
            StringBuffer response = new StringBuffer();

            while((inputLine=in.readLine()) !=null) {
            response.append(inputLine);
            } in.close();
        else{
            System.out.println("GET request failed!");
    }
```

Sada verifikujemo odziv, odnosno ako smo uspješno uspostavili konekciju (sjetimo se da `HTTP_OK` predstavlja zapravo kôd 200 koji označava uspješnu konekciju), pozivamo `BuferedReader` kako bismo zapamtili dobijeni valutni kurs. Deklarišemo varijable

inputLine, koja je tipa String, te varijablu response, koja je instanca odnosno objekat klase StringBuffer[24].

Ovaj kôd:

```
            while((inputLine=in.readLine()) !=null) {
            response.append(inputLine);
            } in.close();

        else{
            System.out.println("GET request failed!");
    }
```

predstavlja logiku koja kaže, dok varijabla inputLine, kojoj smo dodijelili komandu koja iščitava podatke od provajdera valuta, sadrži neke podatke, te podatke dodaj u varijablu response. Na primjer, ako korisnik izabere konverziju iz EUR u RSD, u varijabli response ćemo memorisati idući sadržaj:

```
{"success":true,"timestamp":1622098923,"base":"RSD","date":"20
21-05-27","rates":{"USD":0.010364}}
```

Naravno ako je zahtjev dohvatanja podataka neuspješan (npr. ako je istekla pretplata za korisnički račun), ispisuje se poruka o grešci. Sada nam treba način kako da iz ovih podataka koji su u JSON formatu uzmemo samo valutni kurs i dalje ga proslijedimo u program. Za to ćemo iskoristiti JSONObject klasu koja će prepoznati dio kôda koji nam treba, na idući način:

```
JSONObject obj=new JSONObject(response.toString());
Double kurs=obj.getJSONObject("rates").getDouble(fromCode);
            System.out.println(obj.getJSONObject("rates"));
            System.out.println(iznos+fromCode+" = "+iznos/kurs
+ toCode);

        }
```

Ovaj kôd je već objašnjen u dijelu 1.5 ovog rada.

Sav ovaj kôd možemo dodati u main metod međutim mi ćemo kreirati novi metod sendHttpGETRequest koji će sadržavati svu prethodnu logiku dok će main metod pozivati taj metod:

```
    private static void sendHttpGETRequest(String fromCode,
String toCode, double amount) {
```

[24] StringBuffer klasa je klasa slična klasi String, sa razlikom da nije fiksne veličine.

```
String GET_URL = "https://api.exchangeratesapi.io/v1/latest?
access_key=89ff206a7744c63ba6ae3da722fc8704&base=" + toCode +
"&symbols=" + fromCode;

        URL url = new URL(GET_URL);

        HttpURLConnection httpURLConnection =
(HttpURLConnection) url.openConnection();

        httpURLConnection.setRequestMethod("GET");

        int responseCode =
httpURLConnection.getResponseCode();


        if (responseCode == HttpURLConnection.HTTP_OK)
{//success

            BufferedReader in = new BufferedReader((new
InputStreamReader(httpURLConnection.getInputStream())));

            String inputLine;

            StringBuffer response = new StringBuffer();


            while((inputLine=in.readLine()) !=null) {

    response.append(inputLine);

            } in.close();

            JSONObject obj=new
JSONObject(response.toString());

            Double
exchangeRate=obj.getJSONObject("rates").getDouble(fromCode);

            System.out.println(obj.getJSONObject("rates"));

            System.out.println();

            System.out.println(f.format(amount)+fromCode+" =
"+f.format(amount/exchangeRate) + toCode);

        }

        else{

            System.out.println("GET request failed!");

    }

}
```

U `main` metod dodajemo poziv novostvorenog metoda koji uzima tri parametra:

```
sendHttpGETRequest(fromCode, toCode, iznos);
```

2.2 SITNIJE DORADE

U ovoj fazi program funkcioniše ali može se još doraditi. Na primjer, program izbacuje vrijednosti sa 6 decimala, što je nezgodno za čitanje. Da bismo to riješili, možemo koristiti mogućnosti ranije opisane klase `DecimalFormat`, tako što ćemo prepraviti metod `sendHttpGETRequest`:

- ubacujemo

  ```
  DecimalFormat f = new DecimalFormat("0.00");
  ```

- umjesto

  ```
  System.out.println(iznos+fromCode+" = "+iznos/kurs +
  toCode);
  ```

  unosimo

  ```
  System.out.println(f.format(iznos)+fromCode+" =
  "+f.format(iznos/kurs) + toCode);
  ```

Time smo formatirali ispis na dva decimalna mjesta, što je uobičajeno.

Ukoliko korisnik unese prilikom izbora valuta neki broj koji je izvan vrijednosti 1-5, dobijamo grešku, koju možemo izbjeći ako ubacimo "while" petlju koja će kontrolisati korisnikov unos. Deklarišemo dvije numeričke varijable `from` i `to` koje predstavljaju graničnike vrijednosti za unos, te ako korisnik unese neki broj izvan navedenog intervala petlja ga vraća na ponovni unos.

```
Integer from, to;
while (from < 1 || from > 5) {
                System.out.println("Molim izaberite željenu
valutu putem brojeva (1-5)");
                System.out.println("1:USD (US Dollar)\t 2:EUR
(Euro) \t 3:BAM (Bosanska Marka) \t 4:RSD (Srpski Dinar)\t
5:HRK (Hrvatska Kuna)");
                from = sc.nextInt();
            }
            fromCode = vrijednostiValuta.get(from);

            System.out.println("U koju valutu konvertujete?");
            System.out.println("1:USD (US Dollar)\t 2:EUR
(Euro) \t 3:BAM (Bosanska Marka) \t 4:RSD (Srpski Dinar)\t
5:HRK (Hrvatska Kuna)");
            to = sc.nextInt();

            while (to < 1 || to > 5) {
                System.out.println("Molim izaberite željenu
valutu putem brojeva (1-5)");
```

```
                System.out.println("1:USD (US Dollar)\t 2:EUR
(Euro) \t 3:BAM (Bosanska Marka) \t 4:RSD (Srpski Dinar)\t
5:HRK (Hrvatska Kuna)");
                to = sc.nextInt();
            }
            toCode = vrijednostiValuta.get(to);
```

Konačno, nakon obavljene konverzije, možemo da upitamo korisnika da li želi da ponovi konverziju za neku drugu valutu, što možemo uraditi postavljanjem "do/while" petlje:

```
Boolean running=true;
System.out.println("Želite li nastaviti sa konverzijom?");
            System.out.println("1:Da \t Bilo koji drugi broj:
Ne");
            if (sc.nextInt()!=1) {
                running=false;
            }
        } while (running);
        System.out.println("Hvala Vam za korištenje konvertora
valuta!");
        }
```

Iskoristili smo logičku varijablu running da bi kontrolisali izvršavanje petlje. Budući da se "do/while" petlja izvršava sigurno bar jednom, ako korisnički ulaz nije jednak broju 1, onda se varijabli running dodjeljuje false stanje i program prestaje.

Kompletan konačan kôd programa izgleda ovako:

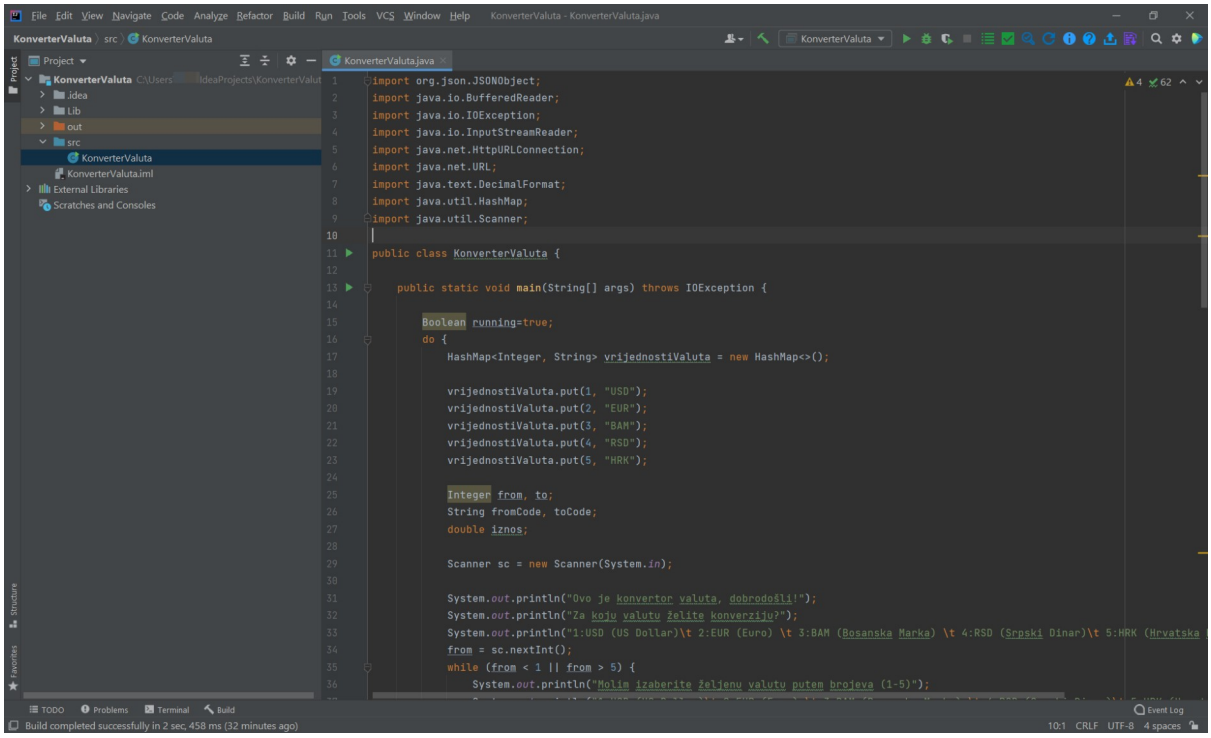
```
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.DecimalFormat;
import java.util.HashMap;
import java.util.Scanner;

public class KonverterValuta {

    public static void main(String[] args) throws IOException {
        Boolean running=true;
        do {
            HashMap<Integer, String> vrijednostiValuta = new
HashMap<>();

            vrijednostiValuta.put(1, "USD");
            vrijednostiValuta.put(2, "EUR");
```

```
            vrijednostiValuta.put(3, "BAM");
            vrijednostiValuta.put(4, "RSD");
            vrijednostiValuta.put(5, "HRK");

            Integer from, to;
            String fromCode, toCode;
            double iznos;

            Scanner sc = new Scanner(System.in);
            System.out.println("Ovo je konvertor valuta,
dobrodošli!");
            System.out.println("Za koju valutu želite
konverziju?");
            System.out.println("1:USD (US Dollar)\t 2:EUR
(Euro) \t 3:BAM (Bosanska Marka) \t 4:RSD (Srpski Dinar)\t 5:HRK
(Hrvatska Kuna)");
            from = sc.nextInt();
            while (from < 1 || from > 5) {
                System.out.println("Molim izaberite željenu
valutu putem brojeva (1-5)");
                System.out.println("1:USD (US Dollar)\t 2:EUR
(Euro) \t 3:BAM (Bosanska Marka) \t 4:RSD (Srpski Dinar)\t 5:HRK
(Hrvatska Kuna)");
                from = sc.nextInt();
            }
            fromCode = vrijednostiValuta.get(from);

            System.out.println("U koju valutu konvertujete?");
            System.out.println("1:USD (US Dollar)\t 2:EUR
(Euro) \t 3:BAM (Bosanska Marka) \t 4:RSD (Srpski Dinar)\t 5:HRK
(Hrvatska Kuna)");
            to = sc.nextInt();

            while (to < 1 || to > 5) {
                System.out.println("Molim izaberite željenu
valutu putem brojeva (1-5)");
                System.out.println("1:USD (US Dollar)\t 2:EUR
(Euro) \t 3:BAM (Bosanska Marka) \t 4:RSD (Srpski Dinar)\t 5:HRK
(Hrvatska Kuna)");
                to = sc.nextInt();
            }
            toCode = vrijednostiValuta.get(to);

            System.out.println("Unesite željeni iznos za
konverziju: ");
            iznos = sc.nextFloat();

            sendHttpGETRequest(fromCode, toCode, iznos);

            System.out.println("Želite li nastaviti sa
konverzijom?");
            System.out.println("1:Da \t Bilo koji drugi broj:
Ne");
```

```
            if (sc.nextInt()!=1) {
                running=false;
            }
        } while (running);
        System.out.println("Hvala Vam za korištenje konvertora
valuta!");
        }

    private static void sendHttpGETRequest(String fromCode,
String toCode, double iznos) throws IOException {
        DecimalFormat f = new DecimalFormat("0.00");
        String GET_URL =
"https://api.exchangeratesapi.io/v1/latest?
access_key=89ff206a7744c63ba6ae3da722fc8704&base=" + toCode +
"&symbols=" + fromCode;
        URL url = new URL(GET_URL);
        HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
        httpURLConnection.setRequestMethod("GET");
        int responseCode = httpURLConnection.getResponseCode();

        if (responseCode == HttpURLConnection.HTTP_OK) {
            BufferedReader in = new BufferedReader((new
InputStreamReader(httpURLConnection.getInputStream())));
            String inputLine;
            StringBuffer response = new StringBuffer();

            while((inputLine=in.readLine()) !=null) {
            response.append(inputLine);
            } in.close();
            JSONObject obj=new JSONObject(response.toString());
            Double
kurs=obj.getJSONObject("rates").getDouble(fromCode);
            //System.out.println(obj.getJSONObject("rates"));
            System.out.println(kurs);
            System.out.println();
            System.out.println(f.format(iznos)+fromCode+" =
"+f.format(iznos/kurs) + toCode);

        }
        else{
            System.out.println("GET request failed!");
          }
     }
}
```

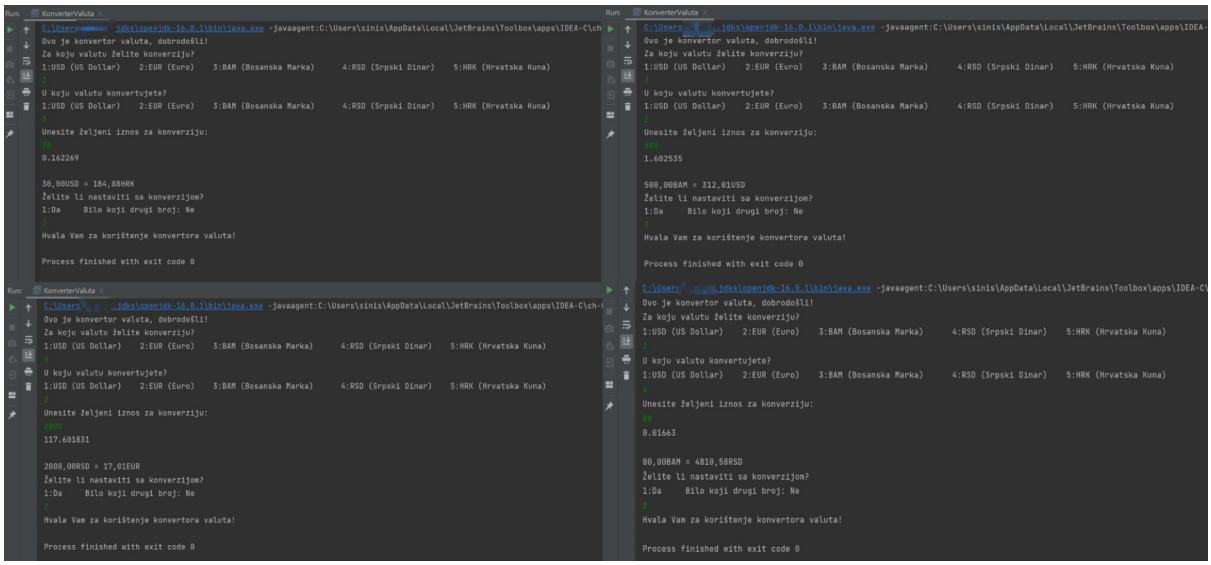
2.3 TESTIRANJE

Za izradu programa i njegovo testiranje korišteno je razvojno okruženje IntelliJ Idea.



```
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.DecimalFormat;
import java.util.HashMap;
import java.util.Scanner;

public class KonverterValuta {

    public static void main(String[] args) throws IOException {

        Boolean running=true;
        do {
            HashMap<Integer, String> vrijednostiValuta = new HashMap<>();

            vrijednostiValuta.put(1, "USD");
            vrijednostiValuta.put(2, "EUR");
            vrijednostiValuta.put(3, "BAM");
            vrijednostiValuta.put(4, "RSD");
            vrijednostiValuta.put(5, "HRK");

            Integer from, to;
            String fromCode, toCode;
            double iznos;

            Scanner sc = new Scanner(System.in);

            System.out.println("Ovo je konvertor valuta, dobrodošli!");
            System.out.println("Za koju valutu želite konverziju?");
            System.out.println("1:USD (US Dollar)\t 2:EUR (Euro) \t 3:BAM (Bosanska Marka) \t 4:RSD (Srpski Dinar)\t 5:HRK (Hrvatska
            from = sc.nextInt();
            while (from < 1 || from > 5) {
                System.out.println("Molim izaberite željenu valutu putem brojeva (1-5)");
```

Prilog 3: IntelliJ Idea



```
Ovo je konvertor valuta, dobrodošli!
Za koju valutu želite konverziju?
1:USD (US Dollar)    2:EUR (Euro)    3:BAM (Bosanska Marka)    4:RSD (Srpski Dinar)    5:HRK (Hrvatska Kuna)
3
U koju valutu konvertujete?
1:USD (US Dollar)    2:EUR (Euro)    3:BAM (Bosanska Marka)    4:RSD (Srpski Dinar)    5:HRK (Hrvatska Kuna)
1
Unesite željeni iznos za konverziju:
80
0.162269

30,00USD = 184,88HRK
Želite li nastaviti sa konverzijom?
1:Da    Bilo koji drugi broj: Ne
2
Hvala Vam za korištenje konvertora valuta!

Process finished with exit code 0
```

```
Ovo je konvertor valuta, dobrodošli!
Za koju valutu želite konverziju?
1:USD (US Dollar)    2:EUR (Euro)    3:BAM (Bosanska Marka)    4:RSD (Srpski Dinar)    5:HRK (Hrvatska Kuna)
3
U koju valutu konvertujete?
1:USD (US Dollar)    2:EUR (Euro)    3:BAM (Bosanska Marka)    4:RSD (Srpski Dinar)    5:HRK (Hrvatska Kuna)
1
Unesite željeni iznos za konverziju:
900
1.602535

500,00BAM = 312,01USD
Želite li nastaviti sa konverzijom?
1:Da    Bilo koji drugi broj: Ne
2
Hvala Vam za korištenje konvertora valuta!

Process finished with exit code 0
```

```
Ovo je konvertor valuta, dobrodošli!
Za koju valutu želite konverziju?
1:USD (US Dollar)    2:EUR (Euro)    3:BAM (Bosanska Marka)    4:RSD (Srpski Dinar)    5:HRK (Hrvatska Kuna)
3
U koju valutu konvertujete?
1:USD (US Dollar)    2:EUR (Euro)    3:BAM (Bosanska Marka)    4:RSD (Srpski Dinar)    5:HRK (Hrvatska Kuna)
2
Unesite željeni iznos za konverziju:
2000
117.601831

2000,00RSD = 17,01EUR
Želite li nastaviti sa konverzijom?
1:Da    Bilo koji drugi broj: Ne
2
Hvala Vam za korištenje konvertora valuta!

Process finished with exit code 0
```

```
Ovo je konvertor valuta, dobrodošli!
Za koju valutu želite konverziju?
1:USD (US Dollar)    2:EUR (Euro)    3:BAM (Bosanska Marka)    4:RSD (Srpski Dinar)    5:HRK (Hrvatska Kuna)
3
U koju valutu konvertujete?
1:USD (US Dollar)    2:EUR (Euro)    3:BAM (Bosanska Marka)    4:RSD (Srpski Dinar)    5:HRK (Hrvatska Kuna)
4
Unesite željeni iznos za konverziju:
80
0.01663

80,00BAM = 4810,58RSD
Želite li nastaviti sa konverzijom?
1:Da    Bilo koji drugi broj: Ne
2
Hvala Vam za korištenje konvertora valuta!

Process finished with exit code 0
```

Prilog 4: Testiranje programa kroz izvršavanje više konverzija valutnih parova

Testiranjem je utvrđeno da program funkcioniše kako je predviđeno. Nema grešaka u programskom kôdu i dok program može da dobije vrijednosti valutnih kurseva od provajdera izvršavanje će biti ispravno.

ZAKLJUČAK

Ovaj rad se bavi izradom programa koji međusobno konvertuje pet odabranih valuta dobavljanjem valutnih kurseva u realnom vremenu. Da bi se program sastavio potrebno je poznavati određene strukture kao što su HashMap, JSON, znati razliku između primitivnih i složenih varijabli, poznavati funkcionisanje petlji, način memorisanja korisničkog ulaza i još dosta sitnijih detalja do kojih se dolazi testiranjem programa. Pošto je u pitanju jednostavniji program, dovoljan je jedan zasebni metod pored glavnog metoda, nema potrebe za stvaranjem novih klasa.

Najteži problem koji treba riješiti je način kako doći do aktuelnih vrijednosti valuta i kako ih inkorporisati u programski kôd. To je riješeno korištenjem klase HttpURLConnection i StringBuffer te konkatenacijom linka provajdera odnosno njegovim parcelisanjem na dvije varijable čijom promjenom se odabira izvorna i odredišna valuta. Ostalo su praktično kozmetičke stvari kao što su način prezentovanja rezultata, petlje koje omogućavaju ponavljanje izbora i odabir samo željenih brojeva koje predstavljaju oznake valuta i tome slično.

Naravno, ovaj kao i druge probleme je moguće riješiti na višestruk broj puta, ali kojim god putem krenuli bitno je doći do rješenja koje funkcioniše na datom setu ulaznih podataka, objasniti korištene strukture i algoritme i pokazati njihovu međusobnu interakciju.

LITERATURA

[1] Core Java, Volume I-Fundamentals, Eleventh Edition, Cay. S. Horstmann, Pearson, 2019

[2] Java A Begginer's Guide, Eight Edition, Herbert Schildt, McGraw-Hill, 2019

[3] Java All-In-One, 11th Edition, Doug Lowe, John Wiley&Sons, 2020

[4] https://docs.oracle.com/javase/specs/jls/se15/html/index.html, The Java Language Specification, Java SE 15, pristupano 24.05.2021