Java aplikacija za vođenje kućnog proračuna

Milardović, Paula

Master's thesis / Diplomski rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:031015

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij računarstva

JAVA DESKTOP APLIKACIJA ZA PREGLED I VOĐENJE FINANCIJA

Diplomski rad

Paula Milardović

Osijek, 2016.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 22.09.2016.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Paula Milardović
Studij, smjer:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D 759 R, 13.10.2014.
OIB studenta:	40824296317
Mentor:	Doc.dr.sc. Krešimir Nenadić
Sumentor:	Tomislav Galba
Predsjednik Povjerenstva:	Doc.dr.sc. Alfonzo Baumgartner
Član Povjerenstva:	Tomislav Galba
Naslov diplomskog rada:	Java aplikacija za vođenje kućnog proračuna
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Objasniti stavke koje se uzimaju u obzir pri vođenju kućnog proračuna. Izraditi model baze podataka koja će podržavati rad takve aplikacije. Izraditi aplikaciju u Java/J++ programskom jeziku i opisati proces izrade. Testirati rad aplikacije. (Sumentor: Tomislav Galba, mag.inž.rač.)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 Postignuti rezultati u odnosu na složenost zadatka: 2 Jasnoća pismenog izražavanja: 3 Razina samostalnosti: 3
Datum prijedloga ocjene mentora:	22.09.2016.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 07.10.2016.

Ime i prezime studenta:	Paula Milardović
Studij:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D 759 R, 13.10.2014.
Ephorus podudaranje [%]:	3%

Ovom izjavom izjavljujem da je rad pod nazivom**: Java aplikacija za vođenje kućnog proračuna**

izrađen pod vodstvom mentora Doc.dr.sc. Krešimir Nenadić

i sumentora Tomislav Galba

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1
1.1 Zadatak diplomskog rada 2
2. OSNOVE JAVA PROGRAMSKOG JEZIKA 3
2.1 Prednosti Java programskog jezika 3
2.2 Razvoj Java programskog jezika 4
2.3 Struktura Java programa 4
3. MYSQL BAZA PODATAKA 9
3.1 Kreiranje baze podataka 9
3.2 ER dijagram 13
4. RAZVOJ APLIKACIJE 15
4.1 Swing aplikacija 15
4.1.1 Arhitektura swinga 15
4.2 Netbeans integrirano razvojno okruženje 16
4.3 Model-View-Cotroller arhitektura 17
4.3.1 Upotreba MVC-a u aplikaciji 18
5. ZAKLJUČAK 35
LITERATURA 36
SAŽETAK 37
JAVA APPLICATION FOR MANAGING THE FAMILY BUDGET 38
ŽIVOTOPIS 39
PRILOZI 40

1. UVOD

Razvoj Java programskog jezika započinje 1990. godine u tvrtki Sun Microsystems (koju je 2010. godine kupila tvrtka Oracle Corporation). Prvotna namjena Jave bila je primjena u području potrošačke elektronike, a tek 1995. godine James Gosling, kao zaposlenik tvrtke Sun Microsystems, dizajnira Java programski jezik kao jednostavniju ali napredniju verziju C++ jezika neovisnog o platformi.

Tijekom prvotne namjene Java programskog jezika, cilj je bio razviti jezik visoke razine koji je pogodan za kontrolu strojeva za pranje rublja kao i u telekomunikacijskim sustavima. Budući da primjena Jave u tim područjima nije donijela željene rezultate, započinje njezina primjena u Internet aplikacijama te je 1995. godine ona predstavljena kao programski jezik za Internet. Prve inačice su se zvale Oak, a tek kasnije Java. Prvi program napisan u Javi se zvao *7, a isti je bio namijenjen za inteligentni daljinski upravljač televizora.

Java programski jezik zadnjih je godina najpopularniji i najrašireniji programski jezik koji se koristi u raznim industrijama i za veliki broj aplikacija kao što su konzolne aplikacije, apleti ili aplikacije s grafičkim sučeljem te šire.

Ovaj diplomski rad se sastoji od četiri poglavlja koja objašnjavaju osnove java programskog jezika, njezinu strukturu te razvoj java desktop aplikacije kao zadatka diplomskog rada

U trećem poglavlju opisuje se početni razvoj za kreiranje desktop aplikacije, izrada dijagrama entiteta i veza (engl. Entity-Relationship Diagram), baze podataka te korištenje MySQL Workbench alata kao pomoć za kreiranje baze podataka te slanja SQL (Structured Query Language) izraza. U zadnjem poglavlju opisuje se arhitektura aplikacije, odnosno obrazac po kojem je rađena naziva Model-View-Controller (MVC) te integrirano razvojno okruženje NetBeans koje je korišteno za izradu aplikacije odnosno povezivanje svih elemenata aplikacije. Osim razvoja aplikacije, diplomski rad također obuhvaća i osnove MySQL-a, odnosno izradu jednostavne baze podataka koja se potom koristi u aplikaciji. Za uspješnu izradu ove desktop aplikacije korištene su sljedeće tehnologije: MySQL Workbench ili program za obradu teksta Notepad++ za kreiranje baze podataka, za izradu dijagrama entiteta i veza te provjeru upita na bazu korišten je alat MySQL Workbench te za kreiranje dekstop aplikacije korišteno je Netbeans razvojno okruženje. Java desktop aplikacija za pregled i vođenje financija daje nam kratki uvid u strukturu java programskog jezika, njezine mogućnosti i široliku primjenu.

1.1 Zadatak diplomskog rada

Zadatak ovog diplomskog rada je izrada Java desktop aplikacije za pregled i vođenje finacija. Aplikacija ima mogućnost pregleda, kreiranja, brisanja i mijenjanja podataka koji se nalaze u MySQL bazi podataka. Baza podataka sastoji se od sedam tablica s jedinstvenim primarnim ključevima. Povezivanje tablica omogućeno je vanjskim ključevima. Svaka tablica sadrži pripadajuće parametre ovisno o potrebama aplikacije. Diplomski rad obuhvaća osnove objektno orijentiranog programiranja. U objektno orijentiranom programiranju glavnu ulogu imaju objekti koji sadrže podatke i funkcije. Objekt je instanca određene klase, stoga je programski kod java programa podijeljen u klase. Java desktop aplikacija kreirana u okviru ovog diplomskog rada sastoji se od više datoteka (engl. package) radi lakšeg snalaženja unutar programskog koda. Svaka datoteka sadrži klase koje obavljaju određene zadatke. Budući da je aplikacija temeljena na Model-View-Controller arhitekturi., također su i datoteke podijeljene na taj način. Unutar datoteke *controller* nalaze se java klase koje komuniciraju s bazom podataka, datoteka *model* sadrži java klase s parametrima koji odgovaraju tablicama u bazi podataka, a datoteka *view* predstavlja grafičko- korisničko sučelje vidljivo krajnjem korisniku. U nastavku diplomskog rada opisane su tehnologije korištene za izradu aplikacije te uvid u primjenu Model-View-Controller arhitekture. Nadalje, rad također obuhvaća osnove MySQL-a odnosno izradu jednostavne baze podataka te način povezivanja baze podataka s aplikacijom.

2. OSNOVE JAVA PROGRAMSKOG JEZIKA

Programski jezik Java je objektno orijentiran te se temelji na principu da je klasa samostalna cjelina u programu, odnosno svaki Java program ima najmanje jednu klasu koja predstavlja neke tipove objekata ili obavlja određeni posao. Pod pojmom Java ne podrazumijeva se samo Java programski jezik već i mnoštvo drugih tehnologija i alata kao što su Java platforme, Java virtualni stroj (engl. Java Virtual Machine - JVM), Java izvedbena okolina (engl. JavaRuntime Environment - JRE), Java razvojna oprema (engl. Java Development Kit- JDK) itd. Programski jezici razlikuju se po tome izvršavaju li u realnom vremenu programski kod ili se isti prije izvođenja mora prevesti. Prvi tip su interpreterski jezici, dok je Java programski prevoditelj, kompilator, odnosno programski jezik koji zahtijeva da se izvorni kod prije izvođenja prevede. Java programi ne prevode se direktno u strojni jezik za Java virtualni stroj, već u poseban međuformat poznat kao byte-code. Java virtualni strojevi u današnje su vrijeme sastavni dio svih popularnijih operacijskih sustava. Programi pisani u Javi stoga se mogu smatrati potpuno platformski nezavisnima. Za razvijanje Java programa potrebno je također imati instaliranu Java razvojnu opremu čime se automatski kreira i Java razvojna okolina na računalu. Nadalje, spomenuta Java platforma dijeli se na tri različite platforme sa specifičnim namjenama a to su Java standardno izdanje (engl. Java Standard Edition, Java SE) koje se koristi na kućnim računalima, Java mikro izdanje (engl. Java Micro Edition, Java JME) koje se koristi za mobilne uređaje, tablete i slične uređaje, te Java poslovno izdanje (engl. Java Enterprise Edition, Java EE) namijenjeno aplikacijama, poslužiteljima itd.

2.1 Prednosti Java programskog jezika

Java ima moštvo prednosti nad ostalim programskim jezicima i okruženjima te je odličan alat za izradu mnogih projekata. Budući da je Java jezik opće namjene njome se mogu razvijati gotovo sve vrste aplikacija. Prema posljednje objavljenim statistikama u svijetu se nalazi devet milijuna Java programera te se na gotovo 5 milijardi uređaja ona pokreće. Statistike također tvrde da je vrijednost poslova obavljenih upotrebom Java tehnologija oko 100 milijardi dolara godišnje, da se tržište igara na mobilnim telefonima procjenjuje na oko 3 milijarde dolara, te da 70% bežičnih aplikacija koristi Javu. Mnoštvo je radnih mjesta na kojima Java programer može naći zaposlenje te su sva raznovrsna budući da je i sam pojam Java širokog obujma. Činjenica da u današnje vrijeme gotovo svako veće poduzeće posjeduje informatičku (engl. Information Technology, IT) službu, potreba za Java programerima se svakim danom povećava.

2.2 Razvoj Java programskog jezika

Kao što je navedeno u uvodu, razvoj Java programskog jezika započinje 1990. godine u tvrtki Sun Microsystems kao Stealth project, kasnije nazvan Green project. Cilj projekta bio je omogućiti povezivanje više elektroničkih uređaja s jednim centralnim uređajem i njihovu međusobnu komunikaciju. Kako bi se ideja projekta realizirala, javila se potreba za razvijanjem nove računalne sklopovske podrške (engl. Hardwer) te njemu prikladnog programskog jezika i kreiranje operacijskog sustava s grafičkim sučeljem koje bi bilo pristupačnom krajnjem korisniku. Dotadašnji programski jezici nisu zadovoljavali potrebe projekta. Stoga se počeo razvijati novi programski jezik imena Oak. Temeljne karakteristike Oak programskog jezika kojima su njegovi izumitelji težili su jednostavnost, pouzdanost, sigurnost i prenosivost. Nadalje, nakon tri godine rada i razvoja Oak programskog jezika, nastao je program naziva *7 korišten za inteligentni daljinski upravljač televizora. Međutim, zbog naglog razvoja ovakvog oblika tehnologije u ondašnje vrijeme, projekt je doživio neuspjeh na tržištu. Oak programski jezik je u međuvremenu preimenovan u Java programski jezik zbog problema oko autorskih prava. Pojavom Interneta, uočeno je da Java sadrži dobre funkcionalnosti i mogućnosti za kvalitetnu izradu web aplikacija te su sukladno tome provedene određene izmjene. Otada pa nadalje, Java je omogućila izradu dinamičkih web stranica, složenih web i desktop aplikacija, neovisno o platformi na kojoj se izvodi. Nadalje, primjena Jave na tržištu mobilnih telefona također je doživjela uspjeh. Time je dokazano da se Java može koristiti u svim područjima tehnologije te je svojim uspješnim projektima postala sastavni dio današnjeg Interneta.

2.3 Struktura Java programa

Java je objektno orijentirani programski jezik, što znači da se java program sastoji od objekata koji međusobno razmjenjuju poruke. Klasa je osnovna programska cjelina u Javi, tj. programski kod Java programa pisan je unutar klasa koje mogu imati metode (funkcije) i svojstva (atribute). Metode su radnje koje klasa obavlja, a atributi predstavljaju karakteristike navedene klase. Klasa je nacrt po kojem se objekti kreiraju a njome su definirane varijable i metode zajedničke za objekte određenog tipa. Kreiranje objekta neke klase naziva se instanciranje objekta, a iste nazivamo instancama određene klase. Izraditi instancu klase znači u memoriji računala izraditi objekt prema definiciji u klasi. Za pokretanje programa pisanog u Javi, najprije se traži `main` metoda te se ona izvršava. Klasa Java programa koja sadrži `main` metodu naziva se izvršna klasa. Uobičajeno je da postoji samo jedna klasa s ovom metodom, a ostale se klase po potrebi

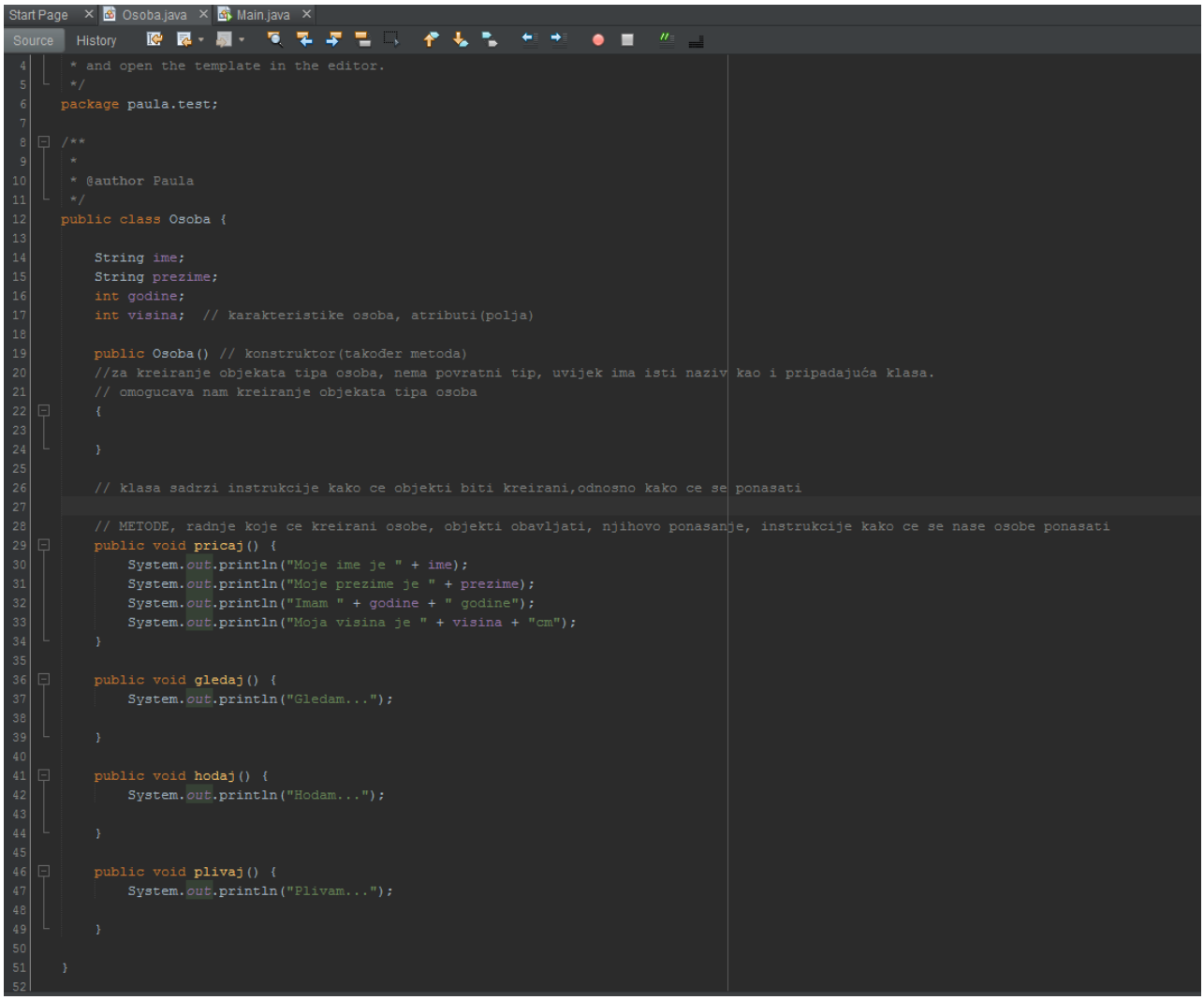
pozivaju iz navedene glavne metode. Važno je naglasiti da postoji više načina na koje se može pristupiti metodama ili varijablama u Java programskom jeziku. Pristupni modifikatori u Javi specificiraju dostupnost podataka u metodi, konstuktoru ili klasi te postoje četiri takva tipa: *private*, *default*, *protected* i *public*. Ukoliko se varijabla unutar neke klase deklarira kao *private*, pristup toj varijabli omogućen je unutar pripadajuće klase. Nadalje, varijable i/ili metode deklarirane modifikatorom *public* mogu se koristit bez ograničenja, odnosno pristup istima dozvoljen je iz svih klasa koje instanciraju dani objekt. Korištenjem modifikatora *protected* pristup varijabli i/ili metodi dozvoljen je iz svih klasa u paketu i svih nasljednih klasa. *Defaultni* pristup omogućava pristupanje varijabli iz svih klasa unutar paketa. Uz spomenute glavne modifikatore mogu se još koristiti modifikatori *final* i *static*. *Final* modifikator koristi se za definiranje konstanti, odnosno za čuvanje vrijednosti koje se nikada ne mijenjaju, dok se modifikator *static* koristi za definiranje varijable koja je zajednička svim objektima koji instanciraju danu klasu. Nadalje, vrijednosti koje predstavljaju objekt nazivaju se varijablama te mogu biti različitih tipova. U Java programskom jeziku razlikuju se dvije kategorije tipova podataka, a to su primitivni tipovi podataka (jednostavni) i reference (složeni). Prema tablici 2.1 može se uočiti koji tipovi podataka pripadaju spomenutim kategorijama

Tab. 2.1 *Tipovi podataka u Java programskom jeziku.*

PRIMITIVNI TIPOVI PODATAKA	REFERENCE
Cjelobrojni: byte,short,int,long	Nizovi(array)
Brojevi s pokretnim zarezom: float,double	Klase(class)
Znakovni: char	Sučelja(interface)
Logički:boolean	

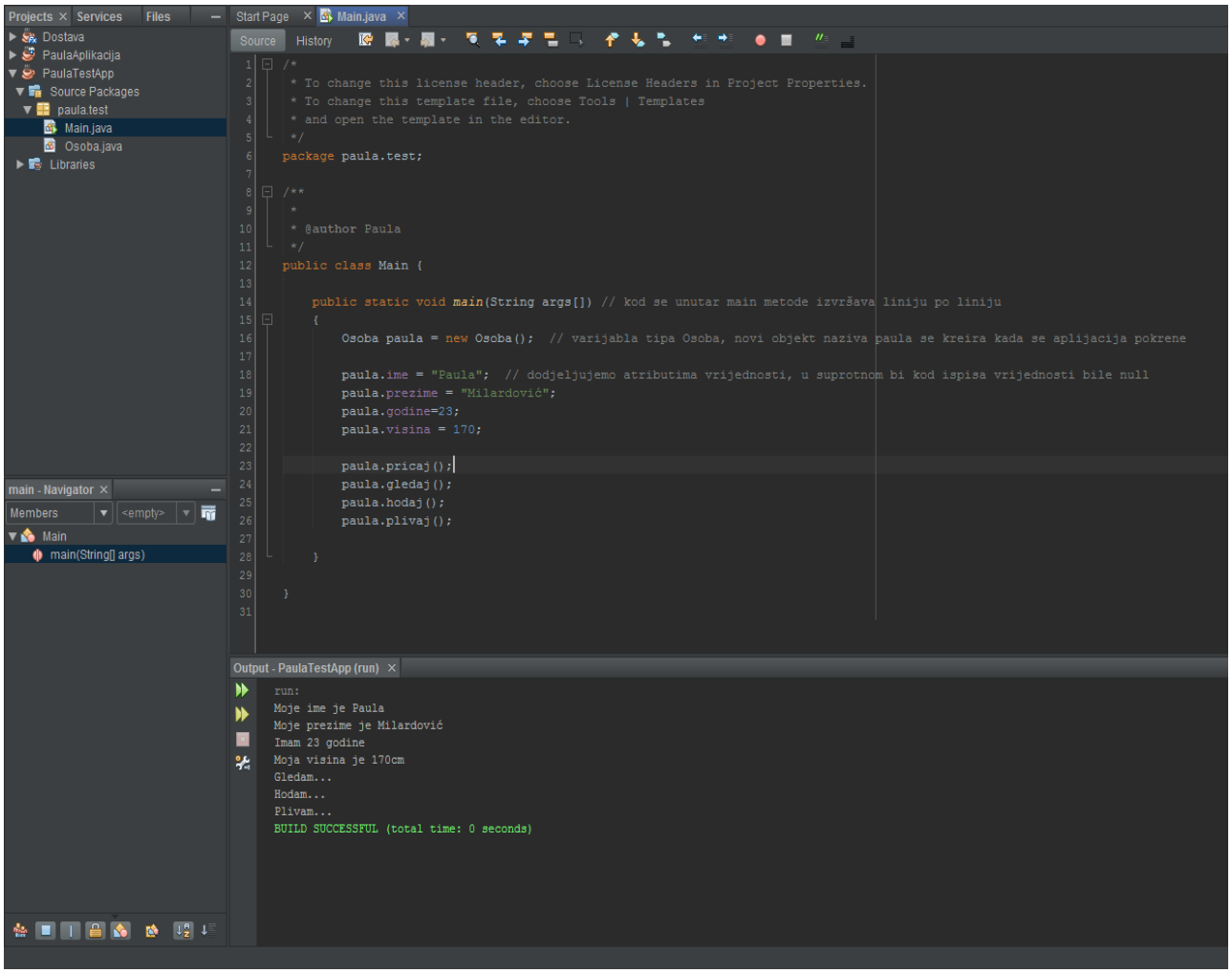
Cjelobrojni tipovi podataka koriste se za prikaz pozitivnih i negativnih cijelih brojeva dok se brojevi s pomičnim zarezom, poznati i kao realni brojevi, koriste ukoliko treba povećati preciznost kod decimalnog dijela broja. Znakovni tip podataka *char* sadržava jedan znak iz *unicode* skupa znakova, a logički tip podataka *boolean* može imati samo dvije vrijednosti: `true` ili `false`. Varijable koje su tipa iz kategorije reference sadrže samo adresu memorijske

lokacije na kojoj se objekt nalazi, odnosno ne sadrže pravu vrijednost. Niz je spremište u kojem se čuva više podataka istog tipa. Prvi element niza u Javi ima indeks 0, odnosno nizovi u Javi se prebrojavaju od nule. Kod deklariranja niza koriste se uglate zagrade i tip podataka koji sadržava spomenuti niz : `String imena[];`. Nizovi mogu biti jednodimenzionalni i dvodimenzionalni koji se deklariraju s dva skupa uglatih zagrada: `String imena[][];`. Na početku potpoglavlja definiran je pojam klase, odnosno da je klasa je nacrt po kojem se objekti kreiraju. Sučelje kao tip podatka predstavlja protokol ponašanja koji klasa može implementirati. Jedna klasa može implementirati više sučelja, te implementiranjem istih mora implementirati i sve metode iz tog sučelja. U nastavku je prikazana osnovna struktura jednostavnog Java programa pisana u NetBeans razvojnom okruženju, o kojemu će više riječi biti u zadnjem poglavlju.



Sl. 2.1 *Klasa osoba.*

Prema slici 2.1 može se uočiti strukturu jednostavne klase naziva `Osoba`. Klasa se sastoji od nekoliko atributa, konstruktora koji ima isti naziv kao i klasa te omogućuje kreiranje objekata tipa `Osoba` te nekoliko metoda koje opisuju radnje koje će objekti obavljati. Svaka klasa ima barem jedan konstruktor i on je javan. Konstruktor također možete služiti za postavljanje početnih vrijednosti ali mu je prvenstvena namjena za kreiranje objekata određene klase. Izraz `System.out.println` koristi se za ispisivanje argumenata prosljeđenih istom. `System` je Java klasa koja se nalazi unutar `java.lang` paketa, `out` je statično polje te klase i tipa je `PrintStream`, a `println` je metoda `PrintStream` klase. Nakon pokretanja aplikacije, odnosno `main` metode, u izlaznom dijelu NetBeansa razvojnog okruženja ispisivati će se prosljeđeni argumenti.



Sl. 2.2 *Main klasa kojom se pokreće napisani Java program.*

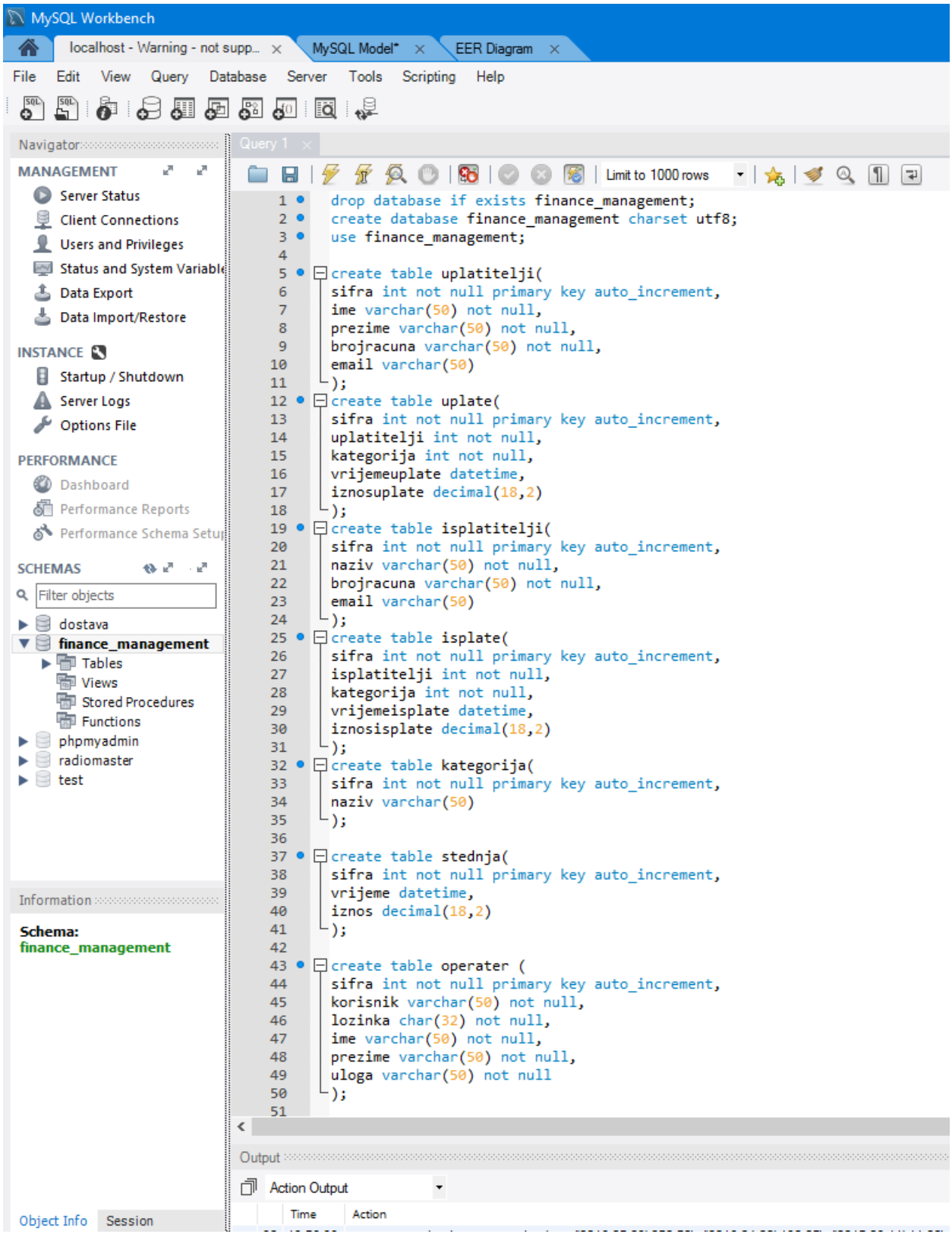
Kako bi se smanjila složenost koda, Java programski jezik uvodi pojam *paketa* (engl.package) kao načina grupiranja više sličih klasa. U spomenutom programu korišten je samo jedan *paket* naziva `paula.test` jer se program sastoji od samo dvije klase. Unutar `main` klase nalazi se `main` metoda koja omogućava pokretanje napisanog Java programa. Programski kod napisan unutar navedene metode izvršava se liniju po liniju te ispisuje u izlaznom dijelu korištenog razvojnog okruženja. Također se u ovoj klasi kreira instanca klase `Osoba`, odnosno novi objekt naziva `paula` kojemu se dodjeljuju željene vrijednosti unaprijed definiranih atributa. Zatim objekt izvodi radnje (metode) napisane u prethodnoj klasi te je u izlaznom dijelu vidljiv rezultat istoga.

3. MYSQL BAZA PODATAKA

MySQL pripada sustavima za upravljanje relacijskim bazama podataka (engl. Relational Database Management System, RDBMS) te omogućuje pristup i upravljanje bazama podataka. U navedenim sustavima se zapisi čuvaju u tablicama, a svaki zapis predstavlja se jednim redom tablice koji se sastoji od jednog ili više stupaca. Kod ovakve vrste baze podataka, organizacija podataka zasniva se na racionalnom modelu, odnosno podaci se organiziraju u skupove između kojih se definiraju određene veze. Svaka relacija/tablica mora imati definiran primarni ključ, koji određuje jedinstvenost svake tablice. Osim primarnog ključa, tablica u bazi može imati i vanjski ključ koji omogućava povezivanje tablica. Najpopularniji sustavi za upravljanje relacijskim bazama podataka su Oracle, DB2, Microsoft SQL Server, MySQL Server itd. Navedeni sustavi razumiju trenutno napoznatiji računalni jezik za kreiranje, traženje, ažuriranje i brisanje podataka iz relacijske baze podataka naziva SQL (Structured Query Language). Za izradu spomenute Java desktop aplikacije korišten je MySQL Server kao sustav za upravljanje i pristup bazi podataka. Osim navedenih sustava postoje i sustavi za upravljanje bazama koji nisu relacijski, odnosno oni ne čuvaju podatke u redovima i stupcima. Takvi sustavi nazivaju se NoSQL baze podataka. Neke on najpoznatijih NoSQL baza podataka su MongoDB, Cassandra, Couchbase itd. Kako bi se omogućilo povezivanje aplikacije s bazom koristi se JDBC (Java Database Connectivity) upravljački program. JDBC omogućuje spajanje s bazom, izvođenje upita na bazu, pohranjivanje podataka u bazu te dohvaćanje rezultata nakon postavljenih upita nad bazom podataka. Budući da je za izradu baze podataka ove aplikacije korišten MySQL sustav, također je korišten i njemu pripadajući JDBC upravljački program naziva `mysql-connector-java-5.1.38-bin` u obliku jar (Java Archive) datoteke, kako bi se omogućilo povezivanje baze s aplikacijom. Detaljniji prikaz i objašnjenje spajanja aplikacije s bazom podataka nalazi se u u narednom poglavlju.

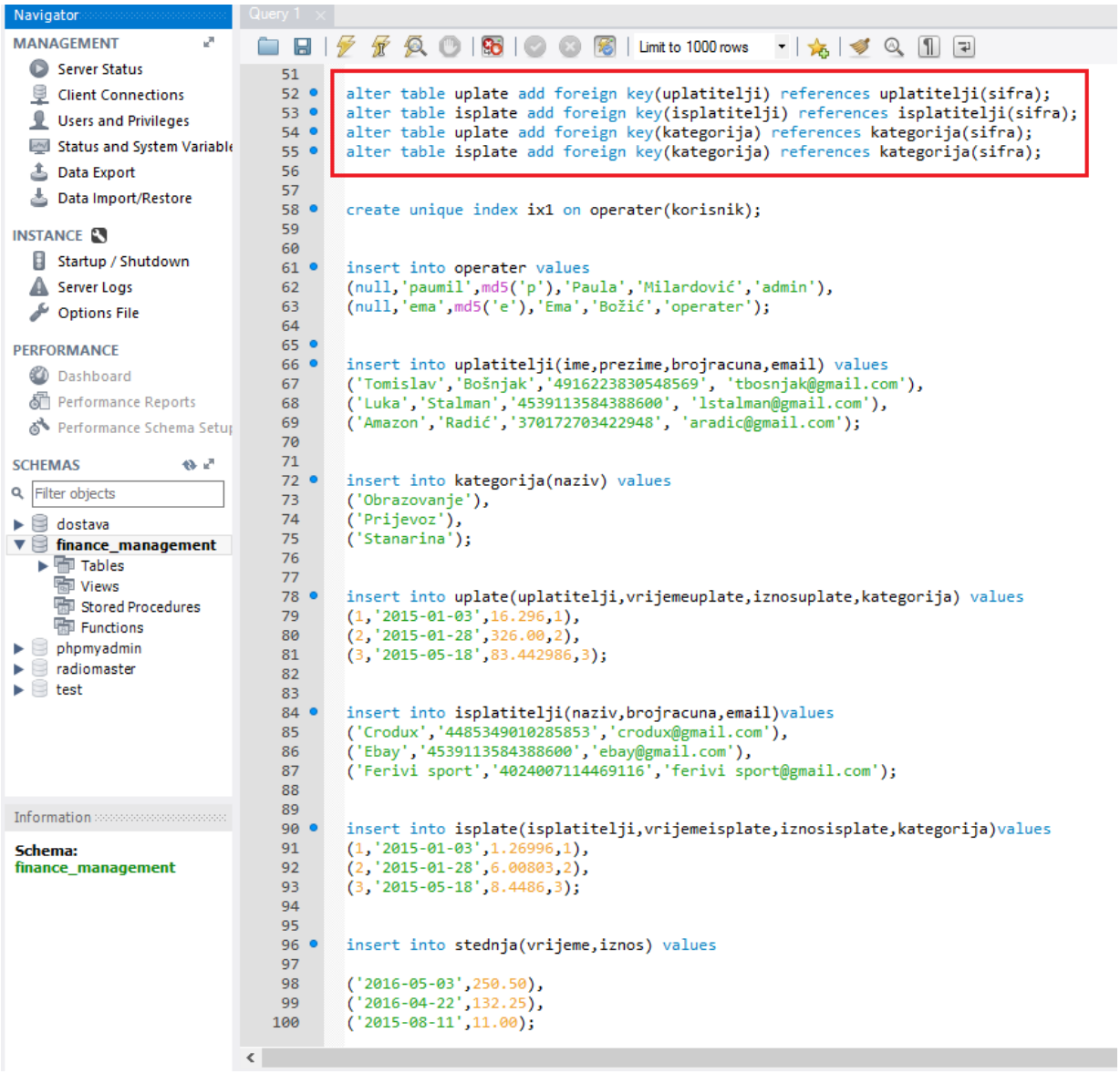
3.1 Kreiranje baze podataka

Baza podataka korištena u Java desktop aplikaciji za pregled i vođenje financija sastoji se od sedam tablica, svaka s jedinstvenim primarnim ključem te pripadajućim redovima i stupcima. Za kreiranje baze korišten je alat Notepad++, jednostavan alat za obradu teksta. Baza podataka se također može kreirati korištenjem alata MySQL Workbench što je vidljivo u nastavku.



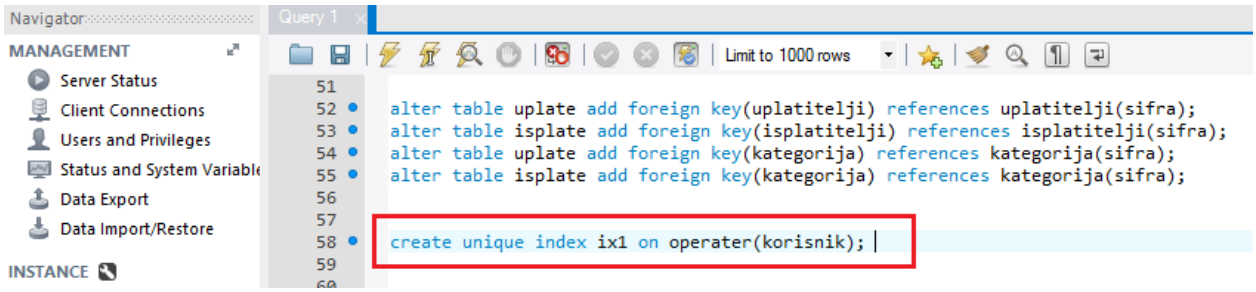
Sl. 3.1 *Kreiranje tablica MySQL baze podataka.*

Svaka tablica u bazi podataka sastoji se od nekoliko elemenata/stupaca s pripadajućim tipom podataka. Svaka tablica ima primarni ključ naziva `šifra`, koji određuje jedinstvenost tablice. Nakon kreiranja tablica baze podataka, iste se povezuju prema potrebi aplikacije. Prema slici 3.1 tablica `uplate` sadrži vanjski ključ/strani ključ tablice `uplatitelji` i tablice `kategorija`, odnosno poveznicu na spomenute tablice. Tablica `isplate` sadrži poveznicu na tablicu `isplatitelji` i tablicu `kategorija` odnosno vanjske/strane ključeve navedenih tablica. Vanjski ključ je veza između tablica, odgovara primarnom ključu te primarni i vanjski ključ moraju biti isti tip podataka. Prilikom kreiranja tablica i njihovih primanih ključeva, dodaje se izraz `auto_increment` čime baza podataka automatski dodijeljuje vrijednost primarnog ključa.



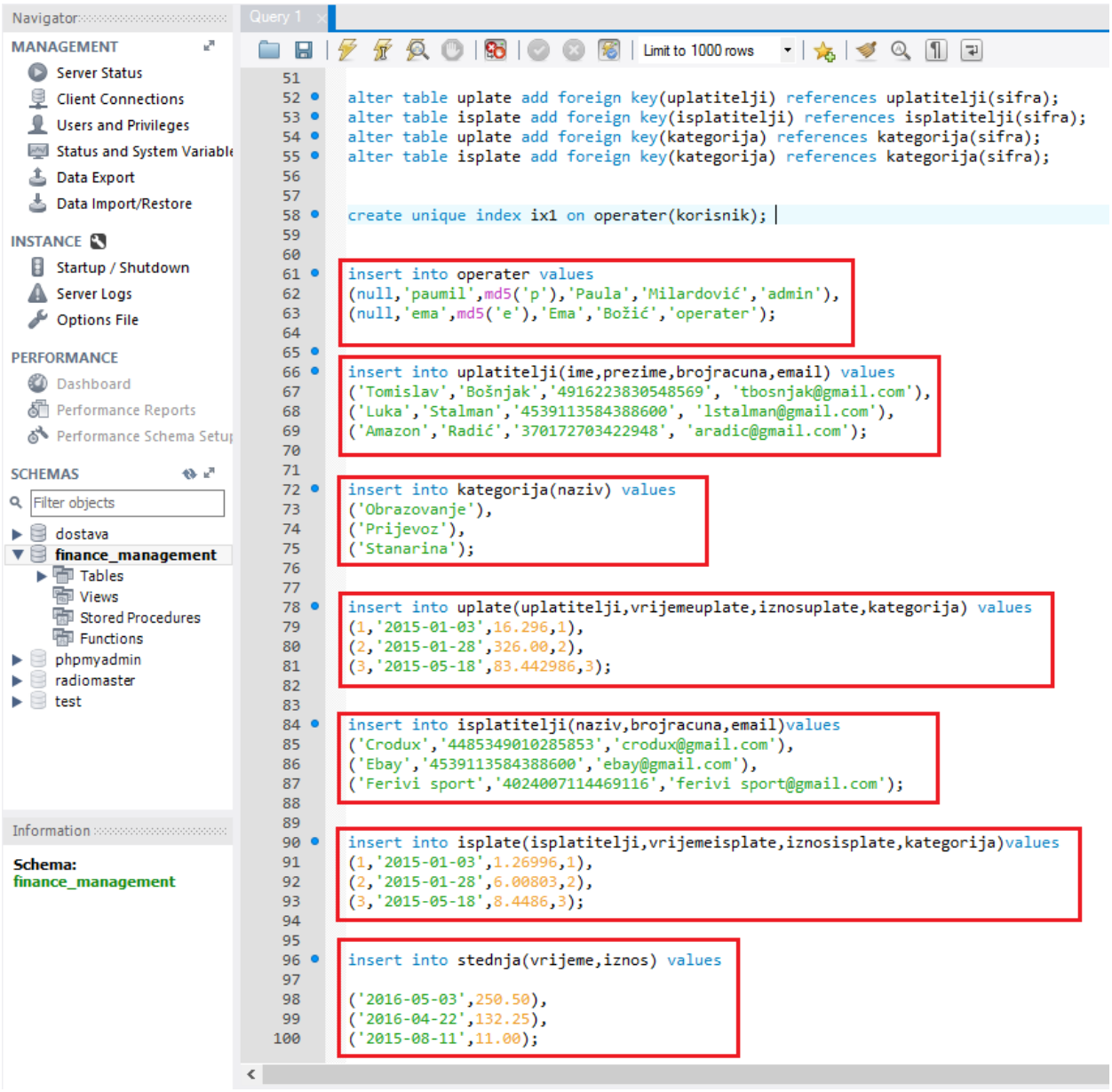
Sl. 3.2 *Povezivanje tablica SQL izrazima.*

Kako bi se povezanost navedenih tablica realizirala, koristi se SQL izraz `alter table` za dodavanje novog stupca u željenoj tablici, `add foreign key` za dodavanje vanjskog/stranog ključa, te izraz `references` koji upućuje na unešenu tablicu s njegovim primarnim ključem. U tablicu `uplate` dodan je vanjski ključ naziva `uplatitelji`, dok je u tablici `uplatitelji` primarni ključ `šifra`. Na taj su način povezane tablice `uplate` i `uplatitelji`, te se isti princip povezivanja koristi i za ostale tablice. Tablica `operater` ima specifičnu namjenu, odnosno podaci koji se nalaze unutar te tablice koriste se za prijavu u desktop aplikaciju. Kako bi se spriječila mogućnost pojavljivanja dvostrukih podataka unutar tablice `operater`, odnosno cilj je da postoje jedinstveni administratori, koristi se SQL izraz `unique index` na tablicu `operater` i željeni stupac tablice.



Sl. 3.3 *Kreiranje jedinstvenog operatera.*

Kako bi baza podataka imala svoju funkcionalnost, potrebno je dodati podatke u nju. Za dodavanje podataka u određene tablice koristi se SQL izraz `insert into`, zatim naziv tablice u koju se podaci dodaju, zagrade s pripadajućim stupcima te tablice i na kraju pripadajuće vrijednosti. Nakon dadavanja podataka u bazu, s istima se može upravljati, dodavati nove, uređivati i brisati postojeće.



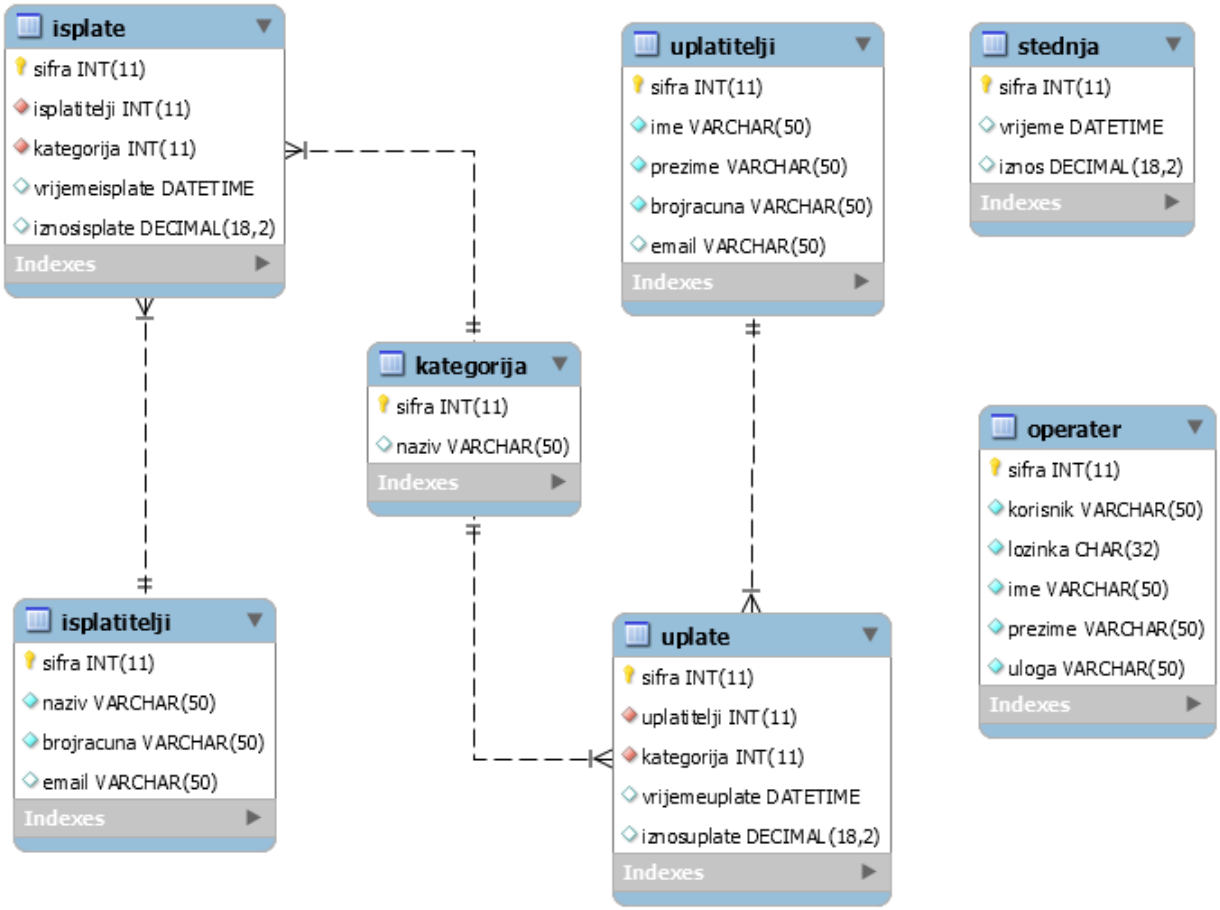
Sl. 3.4 *Dodavanje podataka u stupce svake tablice.*

Nakon kreiranja baze podataka, MySQL Workbench omogućava upotrebu alata koji iz kreirane baze podataka stvara dijagram entiteta i veza (engl Entity-Relationship Diagram). Više riječi dijagramu entiteta i veza te njegovoj namjeni je u sljedećem potpoglavlju.

3.2 ER dijagram

ER (engl. Entitiy-Relationship) dijagram predstavlja detaljan logički prikaz podataka preko entiteta, njihovih atributa i međusobnih veza. Isti olakšava korake pisanja programskog koda te

daje uvid u pojedine tablice, njima pripadajuće stupce, primarne i vanjske ključeve te veze među njima.



Sl. 3.5 *ER dijagram baze podataka.*

Unutar MySQL Workbench alata za kreiranje ER dijagrama, omogućeno je pritiskom na liniju koja povezuje dvije tablice vidjeti primarni i vanjski ključ pojedine tablice što uvelike olakšava uvid u veze među tablicama i njihovim atributima. Pritiskom na liniju koja povezuje tablicu `uplate` i `kategorija`, vanjski ključ `kategorija` iz tablice `uplate` upućuje na primarni ključ `šifra` u tablici `kategorija`. Nadalje, stupci pojedinih tablica koji ne smiju biti `null` vrijednosti označeni su svijetloplavim znakom, dok su stupci čije vrijednosti mogu biti `null` vrijednosti označeni bijelim znakom. Crveni znak pored pojedinih stupaca označava da se radi o vanjskim ključevima u tablici. ER dijagram uvelike olakšava razumijevanje kreirane baze podataka, kakve su veze između tablica te koje tipove podataka pojedina tablica sadrži.

4. RAZVOJ APLIKACIJE

Java desktop aplikacija rađena u okviru ovog diplomskog rada temelji se Model-View-Controller arhitekturi, s bazom podataka pisanom u MySQL-u. Aplikacija ima mogućnost pregleda, unosa, uređivanja i brisanja podataka iz baze za što se ujedno koristi i naziv CRUD (Create, Read, Update, Delete). U ovom poglavlju biti će dataljno opisane tehnologije korištene u izradi aplikacije te detaljniji prikaz i pojašnjenje pojedinih dijelova aplikacije.

4.1 Swing aplikacija

Za izradu grafičkog korisničkog sučelja (engl. Graphical User Interface, GUI) ove aplikacije korišten je skup alata, odnosno grafička biblioteka naziva *Swing* koja je dio Sun-ovog paketa grafičkih komponenti uz biblioteke *java.awt* te *Java 2D*. Tvrtka Netscape Communications Corporation je prva razvila navedenu biblioteku pod nazivom *Internet Foundation Classes* 1996. godine, a godinu dana kasnije tvrtka Sun Microsystem, koja je razvila Java programski jezik, uključuje biblioteku kao standardni dio Java jezika. Isti također omogućuje interakciju korisnika s aplikacijom putem miša, tipkovnice ili drugih ulaznih uređaja. Komponente se nalaze u paketu *javax.swing* koji je prvi put uveden u Java 1.2 inačicu, primarno kreiran kako bi ispravio nedostatke prethodne *java.awt* biblioteke. Nadalje, *Swing* biblioteka je kreirana s istim komponentama kao i njen prethodnik, ali joj je ovisnost o grafičkim sučeljima platformi manja. *Swing* biblioteka nastoji minimalno koristiti resurse platforme na kojoj se program izvršava i stoga daje programskom sučelju trajan izgled na različitim platformama. Pored toga, *Swing* također ima znatno bogatiji izbor grafičkih komponenti, omogućava odabir grafičkih tema pod nazivom *Look and Feel* koje se jednostavno kreiraju i mijenjaju čime se povećava *Swing*-ova prilagodljivost svakoj platformi. Isti je danas standardni paket u Javi za kreiranje korisničkog sučelja, međutim *Swing* biblioteka nije potpuna zamjena za *AWT* (Abstract Windowing Toolkit) biblioteku i obje biblioteke se često koriste zajedno.

4.1.1 Arhitektura swinga

Swing je platformski nezavisna grafička biblioteka za izradu grafičkih korisničkih sučelja Java programa te se velikim dijelom oslanja na već spomenutu Model-View-Controller arhitekturu. Taj tip arhitekture konceptualno razdvaja podatke predstavljene korisniku od sučelja preko kojeg su mu podaci prikazani, odnosno olakšava prikazivanje vizualnog dijela aplikacije i omogućava lakše razumijevanje same pozadine aplikacije, odnosno programskog koda. Platformska nezavisnost temelji se na kreiranju vizualnih komponenti neovisno o komponentama patforme na

kojoj se program izvodi korištenjem Java 2D grafičke biblioteke. Primarna razlika *Swing*-a i *AWT*-a je ta što se korištenjem *java.awt* biblioteka grafička sučelja kreiraju od postojećih komponenti sustava. Nadalje, prilagođavanje vizualnih modela *Swing* komponenti korisničkim potrebama povećava prilagodljivost i jednostavnost upotrebe *Swing*-a. Vizualni izgled komponenti određuje se kompozicijom standardnih grafičkih elemenata kao što su okviri prozora, klizači, ukrasi itd. te korisnik po potrebi može zadati grafičke elemente, boje, oblike, prozirnosti ovisno o svojstvu pojedine komponente. Osim navedenog, korisnik također može mijenjati ili nadograđivati postojeće komponente i njihove funkcionalnosti. Važno je napomenuti da je sastavni dio svake grafičke komponente *AWT*-ov *Container*. To je osnovna klasa u *AWT* (Abstract Windowing Toolkit) okviru koju nasljeđuju sve *Swing*-ove klase. Neki od *Containera* u *Swing* paketu su *JFrame, JPaneli, JContentPane* itd. Klasa *JFrame* je glavni prozor aplikacije koji nasljeđuje klasu *Frame* iz biblioteke *java.awt,* a *JPanel* predstavlja *Container* koji se koristi za grupiranje komponenti unutar prozora. Osnovna klasa *javax.swing* paketa je *JComponent* klasa te sve swing komponente nasljeđuju spomenutu klasu. Neke od osnovnih komponenti koje su korištene u aplikaciji su već spomenuti *JFrame,* prozor koji je na vrhu hijerarhije *Containera* s naslovom trakom, trakom alata te *minimize*, *maximize* i *close* gumbovima, *JPanel*, *JButton* koji predstavlja *Swing* gumb koji može sadržavati ikonu ili tekst te se koristi za otvaranje prozora ili vršenja određenih akcija, *JLabel* komponenta koja omogućava prikazivanje teksta i ikone na odgovarajućem mjestu, *JTextField* komponenta koja predstavlja tekstualno polje i omogućava unos teksta s tipkovnice, *JPasswordField* itd. U daljnjim potpoglavljima će biti detaljnije objašnjeni dijelovi i komponente korištene za izradu aplikacije.

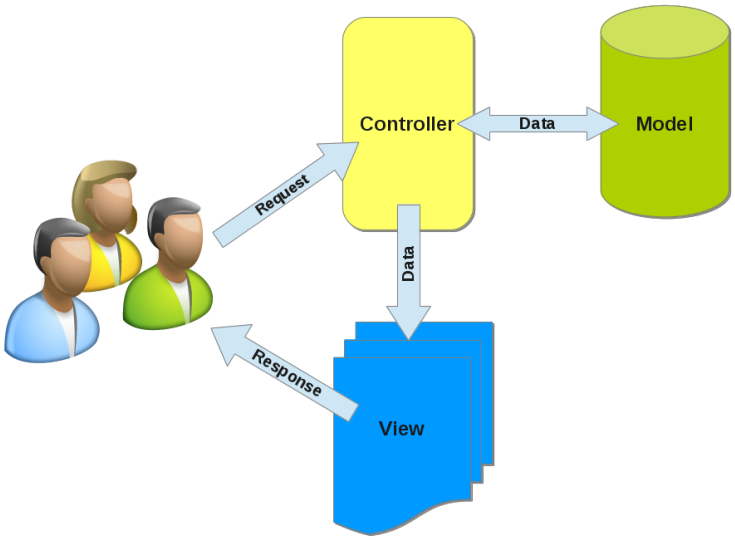
4.2 Netbeans integrirano razvojno okruženje

Dizajn i struktura aplikacije rađena je u Netbeans razvojnom okruženju koji predstavlja jedan od najkvalitetnijih *open source* razvojnih okruženja (engl. Integrated Development Environent, IDE) na tržištu, te omogućava razvoj aplikacija u mnoštvu programskih jezika ali je primarno namijenjen razvoju Java aplikacija. NetBeans je razvijen pomoću Java programskog jezika te je njegova je dostupnost omogućena svim platformama. Nastao je iz studentskog projekta zvanog Xelfi 1996. godine u Češkoj. Nakon postavljanja proizvoda na tržište, isti postiže veliki uspjeh da bi ga 1996. godine kupila već poznata tvrtka Sun Microsystems i nakon godinu dana je NetBeans objavljen pod *open source* licencom. Ono što NetBeans čini osobito zanimljivim je set integriranih alata za izradu i razvoj grafičkih korisničkih sučelja temeljenih na standardnim *AWT*

(Abstract Windowing Toolkit) i već spomenutim *JFC/Swing* (Java Foundation Classes) komponentama. Po mogućnostima i osobinama NetBeans IDE parira ostalim komercijalnim alatima, a zbog svoje jednostavnosti izabran je kao alat za izradu java desktop aplikacije ovog diplomskog rada.

4.3 Model-View-Cotroller arhitektura

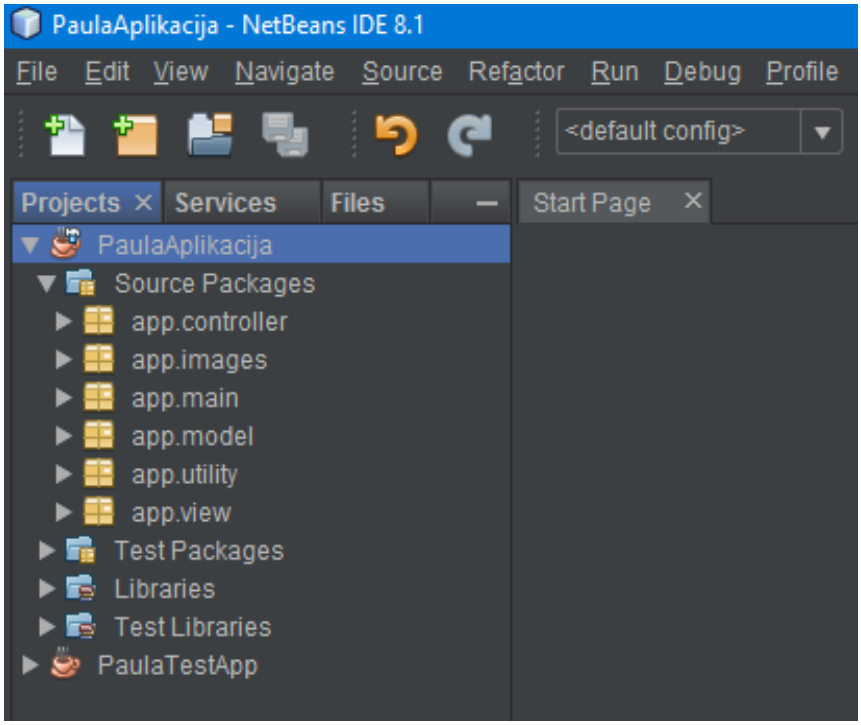
Model-View-Controller obrazac koristi se za odvajanje aplikacije na tri glavne logičke komponente poznate kao model, pogled (engl. View) i upravitelj (engl. Controller). Svaka od navedenih komponenti ima specifičnu namjenu i funkcionalnost. Takva struktura aplikacije olakšava upravljanje složenim aplikacijama zbog mogućnosti posebnog razvoja svake od komponenti aplikacije i testiranja svakog dijela posebno. Svaka od navedenih komponenti izvršava određene zadatke. Model se sastoji od podataka, poslovnih pravila i informacija koji su ugrađeni u logiku programa. Isti odgovara svim logičkim podacima s kojima korisnik radi. To može predstavljati ili podatke koji se razmijenjuju između komponenti *View*-a i *Controller*-a ili bilo koji drugi podaci poslovne logike. Nadalje, model predstavlja jednu ili više klasa sa svojim stanjima koji se prikazuju na zahtjeve *View*-a ili se mijenjaju ovisno o *Controlleru*. Pogled (engl. View) je vizualni dio aplikacije, on zahtjeva od modela podatke/informacije potrebne za vizualni prikaz modela krajnjem korisniku, odnosno za prikaz podataka, te omogućava mijenjanje podataka. Upravitelj (engl. Controller) djeluje kao sučelje između modela i pogleda kako bi obradio poslovnu logiku i zahtjeve koji dolaze, manipulirao podacima pomoću modela te obavljao interakciju s pogledom s ciljem prikazivanja krajnjih podataka, odnosno upravitelj obavlja interakciju s modelom kako bi kreirao podatke koji će se prikazivati u pogledu. Prema slici 4.1 prikazana je MVC arhitektura i njezin princip rada. MVC aplikacija komunicira s korisnikom putem pogleda, gdje korisnik unosom zahtjeva šalje naredbe upravitelju, dok upravitelj od modela traži da obavi radnju i vrati mu rezultat. Rezultat radnje zatim kontroler šalje u pogled gdje ga korisnik može vidjeti.



Slika 4.1 *Prikaz MVC arhitekture (http://perl-diving.blogspot.hr/p/blog-page.html).*

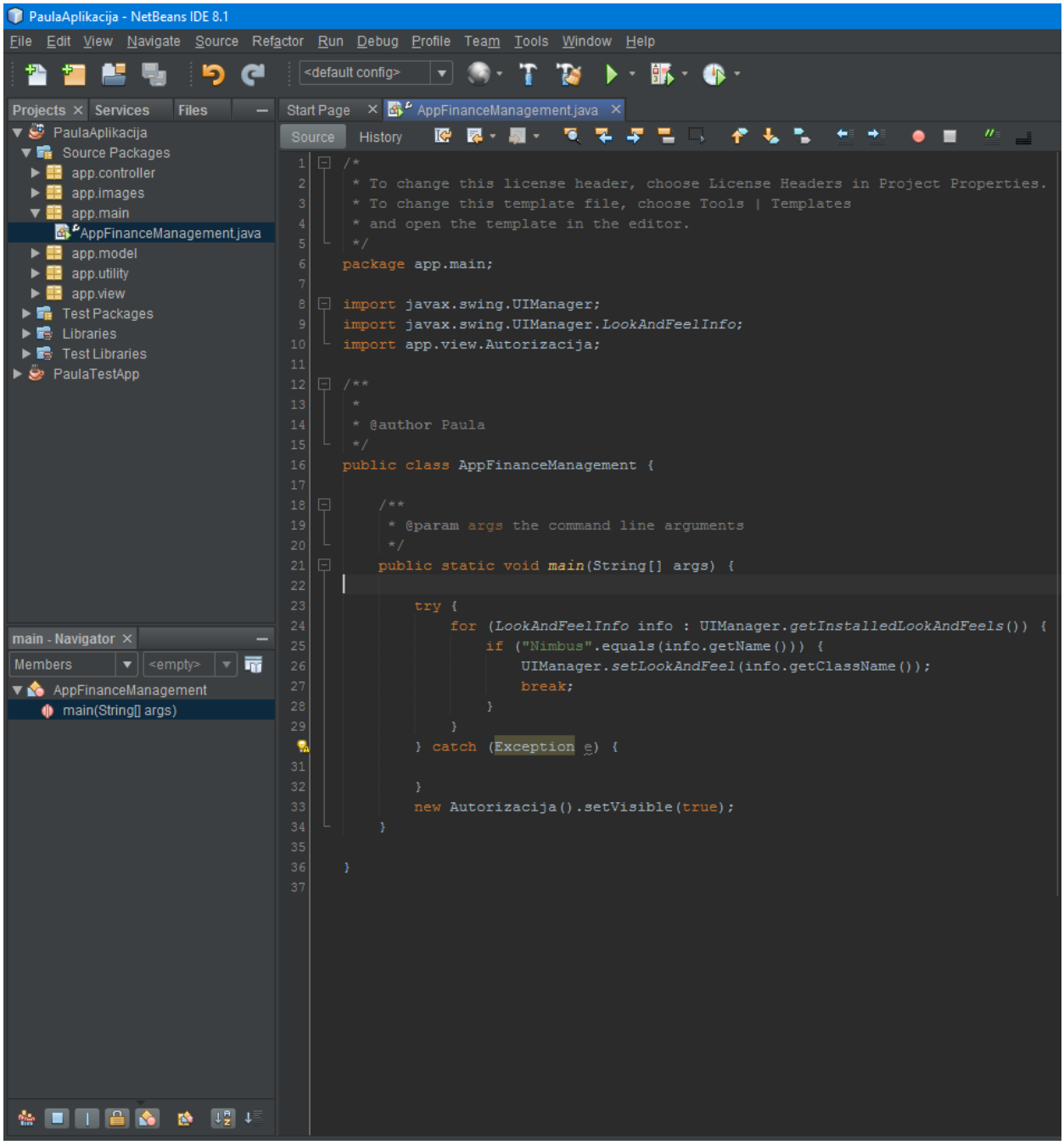
4.3.1 Upotreba MVC-a u aplikaciji

U navedenoj desktop aplikaciji MVC arhitektura primjenjena je na način da je programski kod podijeljen na više *paketa* (engl. Package). Prema slici 4.2 vidljiva je struktura aplikacije i podjela na *pakete*. Svaki od navedenih paketa sadrži određeni broj klasa od kojih svaka klasa izvršava određene zadatke.



Slika 4.2 *MVC arhitektura aplikacije podjelom na višestruke pakete.*

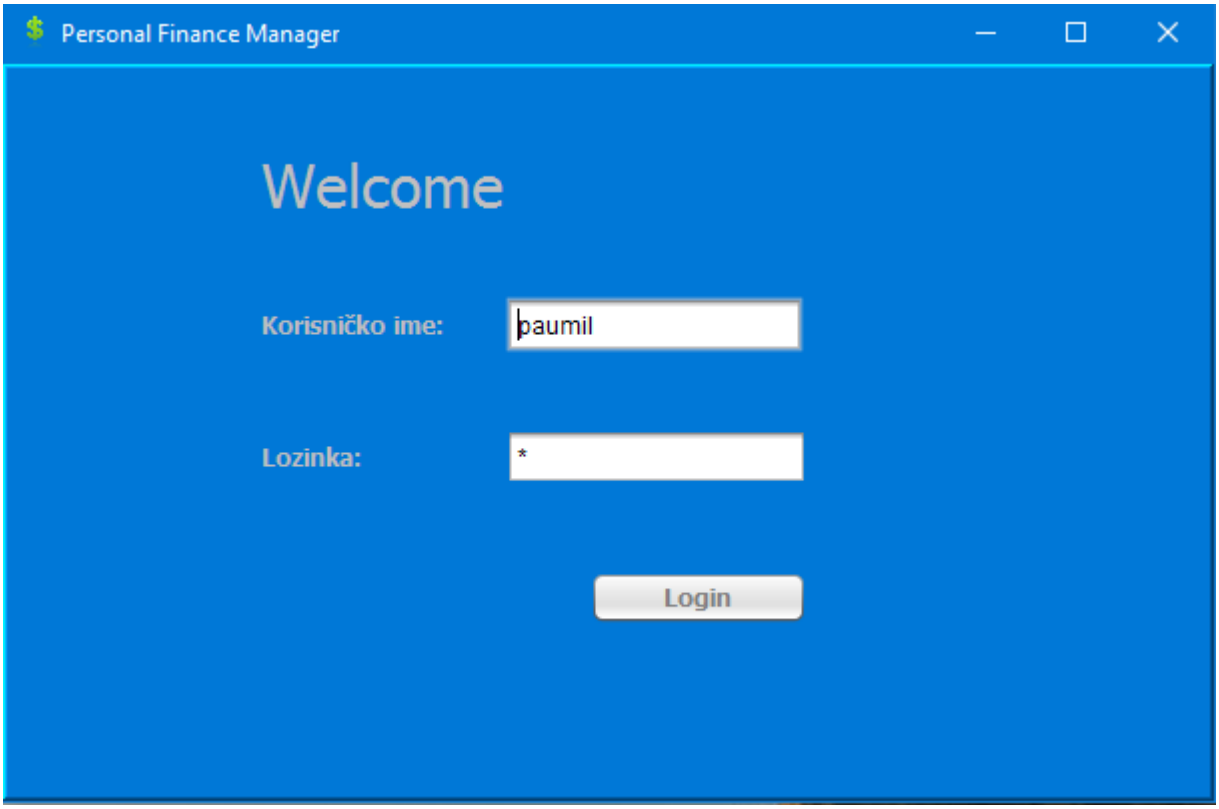
Svaki od navedenih paketa sadrže klase koje obavljaju određene zadatke. Prema slici 4.3 vidljiv je sadržaj paketa `app.main` te klasa naziva `AppFinanceManagement` koja sadrži `main` metodu.



Slika 4.3 *Klasa AppFinanceManagement s izvršnom metodom*

Spomenuta klasa sadrži `main` metodu, stoga se može nazvati i izvršna klasa budući da se prilikom pokretanja Java programa najprije izvršava `main` metoda. Unutar `main` metode može se uočiti postavljanje grafičke teme `Nimbus` koja se primjenjuje na cjelokupnu aplikaciju. Klasa `UIManager` koja se nalazi u biblioteci `javax.swing` sadrži podklasu naziva `Nested class`

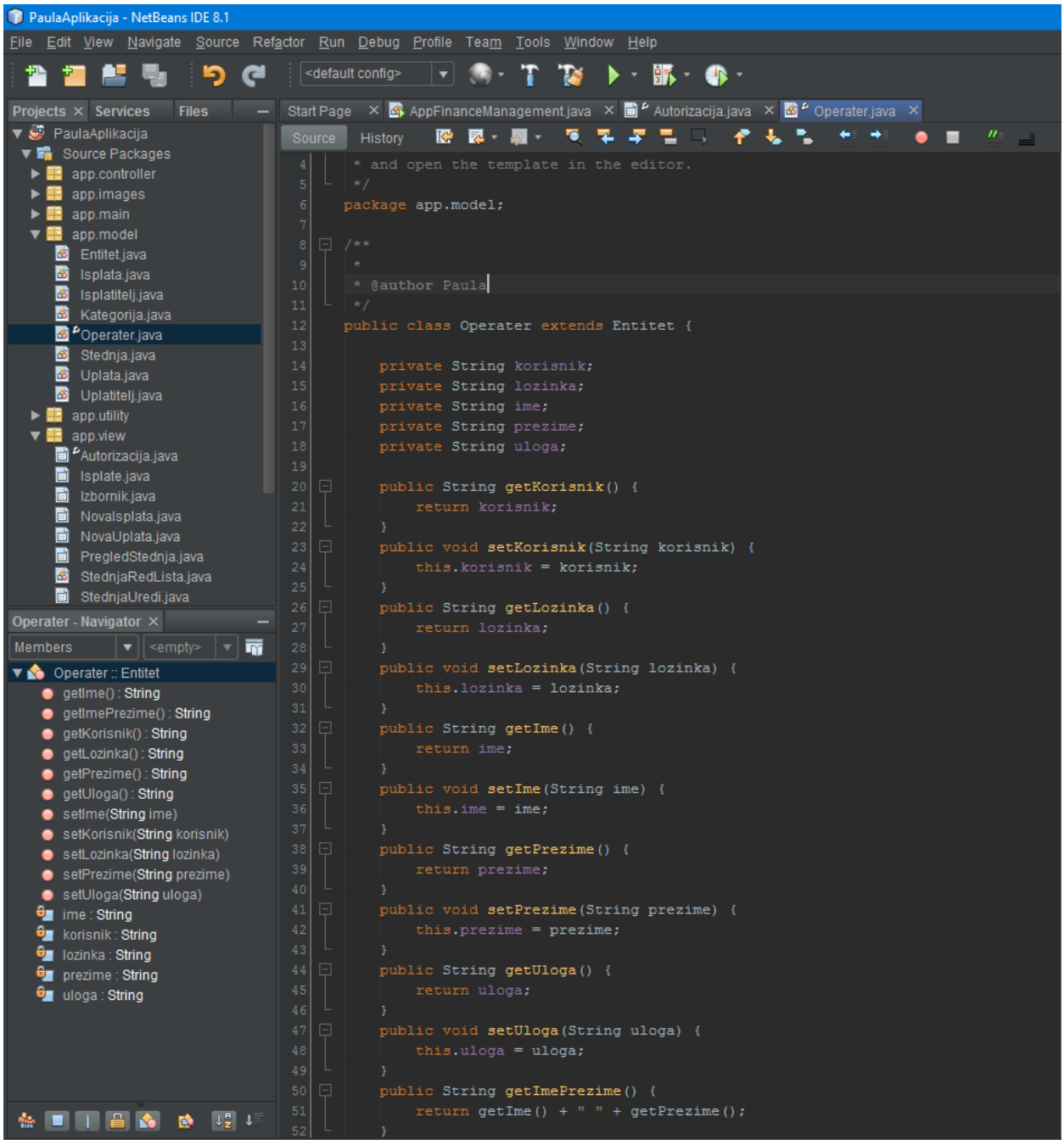
`LookAndFeelInfo` s informacijama o postojećim grafičkim temama. Metodom naziva `getInstalledLookAndFeels` nastoji se dobiti niz `LookAndFeelInfo` objekata koji opisuju svaku grafičku temu raspoloživu u trenutnom sustavu. Ukoliko tražena grafička tema postoji, metodom `getName` vraća se naziv teme u obliku koji odgovara prikazu. Nadalje, `setLookAndFeel` postavlja željenu temu, a metodom `getClassName()` vraća se ime klase koja implementira tu grafičku temu. Kao što je već spomenuto, `main` metoda je prva metoda koja se izvršava, stoga se prilikom pokretanja dekstop aplikacije primjenjuje `Nimbus` grafička tema te je prvi prozor (engl. JFrame) koji se pojavljuje, nakon instanciranja, `Autorizacija` (Sl. 4.4).



Sl.4.4 *Prvi prozor koji se pojavljuje prilikom pokretanja aplikacije*

Prozor `Autorizacija` koji se nalazi u paketu `app.view`, sastoji se od dva `JTextField`-a za unos korisničkog imena i zaporke, `JButton`-a za prijavu u aplikaciju i `JLabel`-i koje upućuju na pripadajuća polja za unos teksta. Za kreirani prozor napisan je programski kod kako bi dobio svoju funkcionalnost, točnije nakon klika na gumb Login slijedi ulazak u aplikaciju. Kako bi to bilo moguće, potreban je model s pripadajućim parametrima te upravitelj koji će slati upit na bazu radi provjere unešenih podataka. U trećem poglavlju koje opisuje kreiranje MySQL baze

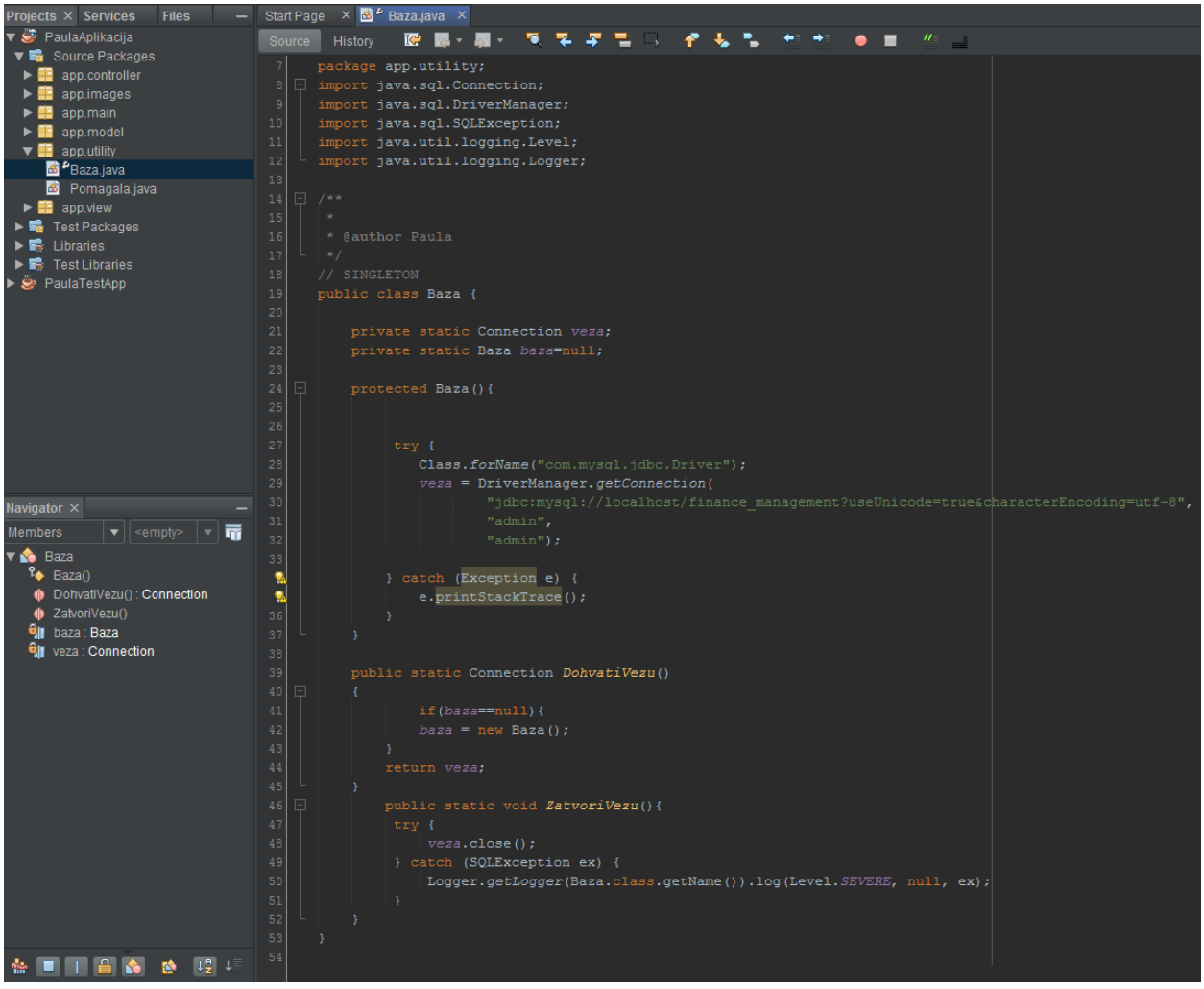
podataka, vidljiva je tablica naziva *Operater* koja sadrži parametre prema kojima se kreira model istog naziva. Osim parametara, klasa *Operater* (Sl. 4.5) također sadrži javne metode naziva *Get()* i *Set()* za pristupanje parametrima i vraćanje rezultata, te za upisivanje vrijednosti u polja. Osim spomenutog, klasa *Operater* nasljeđuje (engl. Extends) klasu Entitet koja sadrži samo parametar *šifra*. Također i sve ostale klase unutar paketa *app.model* nasljeđuju klasu *Entitet* te njezine varijable i metode. Nasljeđivanje omogućava proširivanje i definiranje novih klasa na temelju postojećih. Sve klase direktno ili indirektno nasljeđuju i razrađuju funkcionalnost iz klase koja se nalazi na vrhu hijerarhije svih klasa naziva *java.lang.Object*.



Sl.4.5 *Model Operater s pripadajućim patametrima*

Nadalje, korisnik pritiskom na gumb *Login* šalje zahtjev upravitelju, odnosno klasi koja se nalazi u paketu `app.controller` naziva `ObradaAutorizacija`. Kako bi prijava u aplikaciju bila uspješna potrebno je poslati upit na bazu podataka s ciljem provjere parametara, odnosno da li unešeni podaci prilikom prijave odgovaraju onima u bazi podataka. Kao što je već spomenuto u trećem poglavlju, za spajanje baze s aplikacijom potreban je JDBC API(Java Database Connectivity) naziva `mysql-connector-java-5.1.38-bin` koji je dodan u aplikaciju u

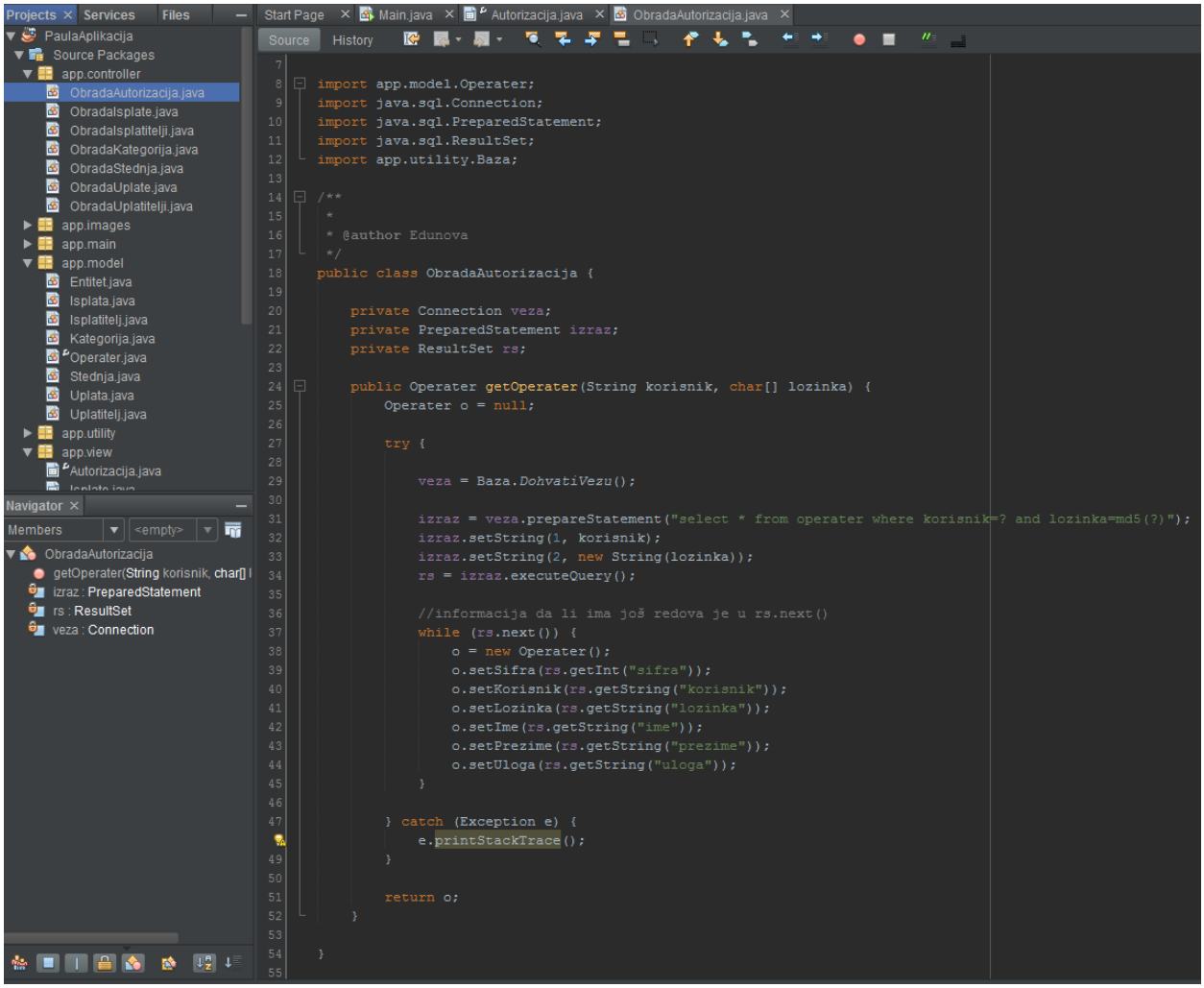
folder *lib*. JDBC upravljački programi su sučelja između Java programa i relacijske baze podataka. Java sadrži dva *paketa* u kojima se nalaze klase potrebe za rad s bazom podataka. *Paket* korišten u ovoj aplikaciji naziva se *java.sql* te sadrži klase *Connection*, *PreparedStatement/Statement* i *ResultSet*. Radi lakšeg snalaženja kreirana je zasebna klasa u paketu *app.utility* naziva *Baza* i takav princip pisanja koda naziva se *Singleton*. Isti omogućava da za sve interakcije s bazom bude zadužen samo jedan objekt, odnosno u svakoj klasi unutar paketa *app.controller* ne mora se ponovno pisati sav kod za povezivanje na bazu.



Sl.4.6 *Singleton obrazac*

Za ostvarivanje veze na bazu podataka potrebno je najprije učitati JDBC upravljački program pozivanjem metode *forName()* Java klase *Class*. Nadalje, otvaranje veze s bazom podataka ostvaruje se pozivanjem *DriverManager* klase i njegove metode *getConnection()* koja prima parametre *user*,*password* i *url*. Zatim se kreira instanca klase *Baza* unutar metode

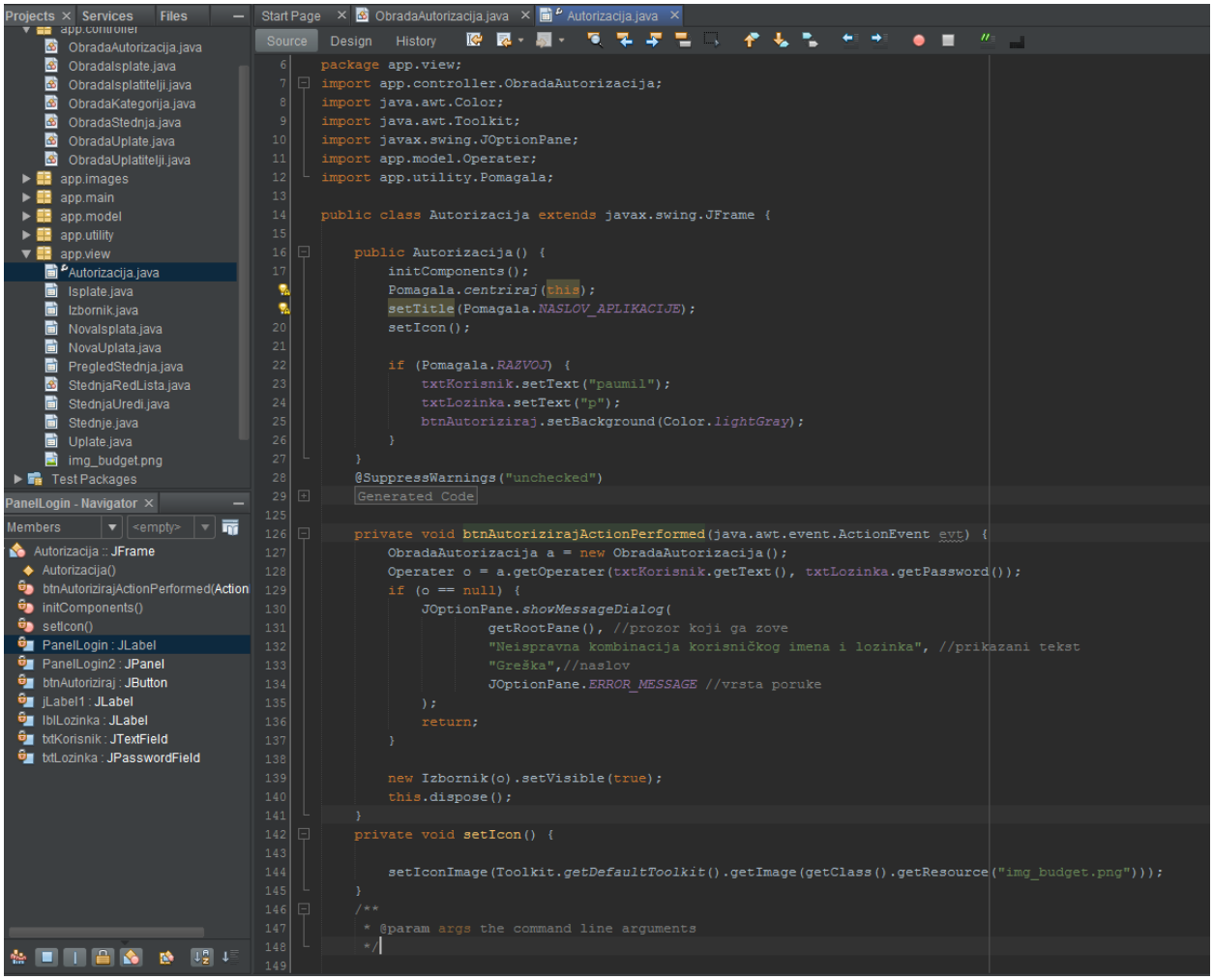
`DohvatiVezu()` koja će se potom koristit u ostalim upravitelj klasama. Za oslobađanje računalnih resursa zatvara se veza. Prema slici 4.7 vidljiva je samo jedna linija koda putem koje se ostvaruje veza na bazu. Nakon uspostavljene veze koristi se SQL izraz `SELECT` za dohvaćanje podataka iz baze. Prije slanja upita na bazu potrebno je kreirati objekt koji može izvršavati SQL. U ovoj aplikaciji korištena je klasa `PreparedStatement` za izradu spomenutog objekta koja prevodi SQL izraz prije izvršavanja. Parametre za SQL iskaz pruža metoda, u ovom slučaju, `setString`. Za obradu petlje dohvaćenih podataka koristi se objekt `ResultSet` gdje parametar `rs` predstavlja kursor. Nakon provjerenih parametara metoda `getOperater` koja prima korisnika i zaporku nam vraća model `Operater` s pripadajućim parametrima.



Sl.4.7 *Klasa ObradaAutorizacija za provjeru operatera*

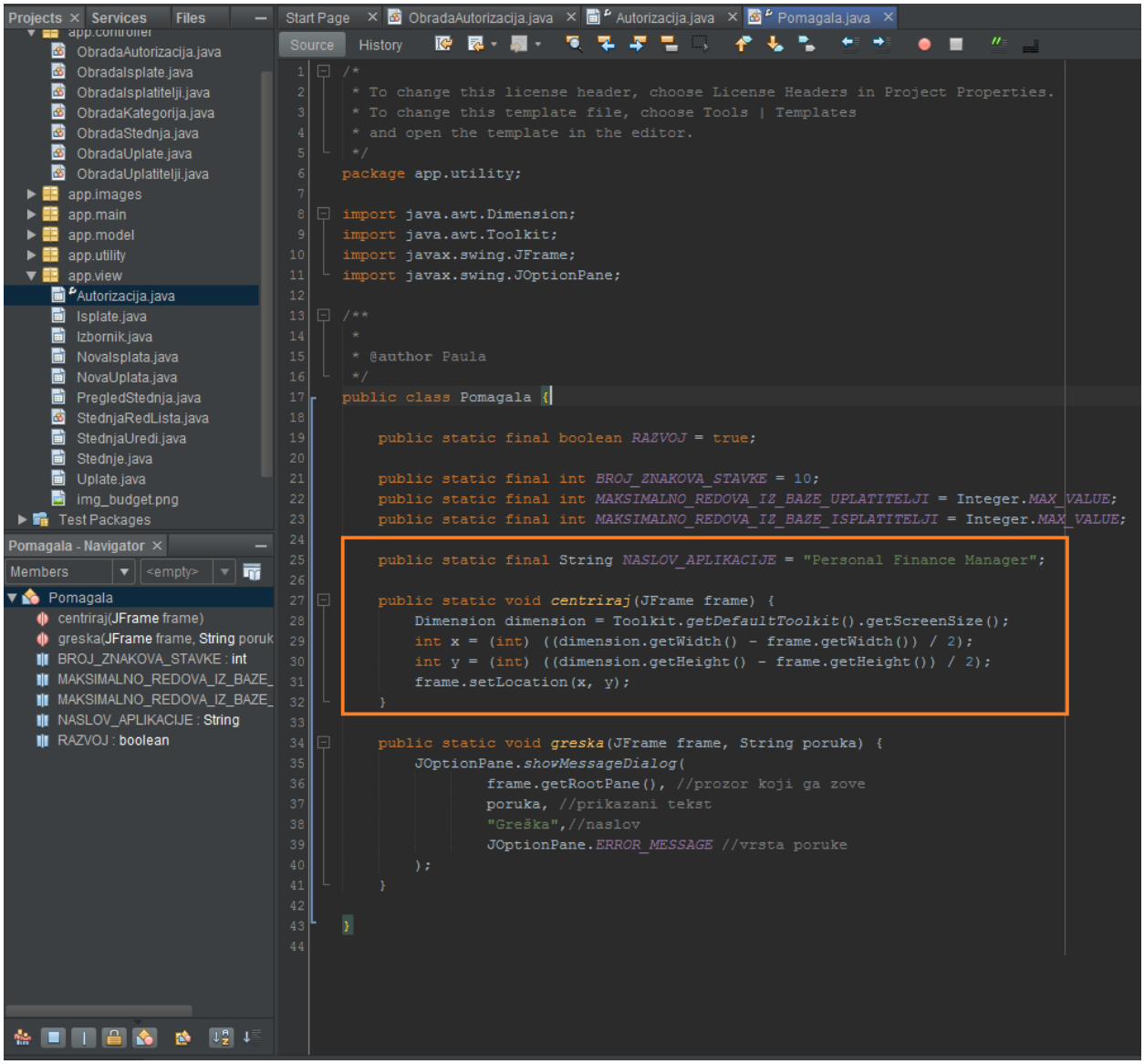
Unutar paketa `app.view` nalazi se forma za unos korisničkog imena i zaporke, to je ujedno i prvi prozor koji se pojavljuje prilikom pokretanja aplikacije (Sl. 4.4). Osim kreiranja grafičko-korisničkog sučelja potrebno je definirati koja se radnja odvija pritiskom na gumb *Login*.

NetBeans razvojno okruženje omogućava automatsko kreiranje metode naziva `OnActionPerformed` dvostrukim klikom na željenu komponentu.



Sl.4.8 *Programski kod forme za autorizaciju*

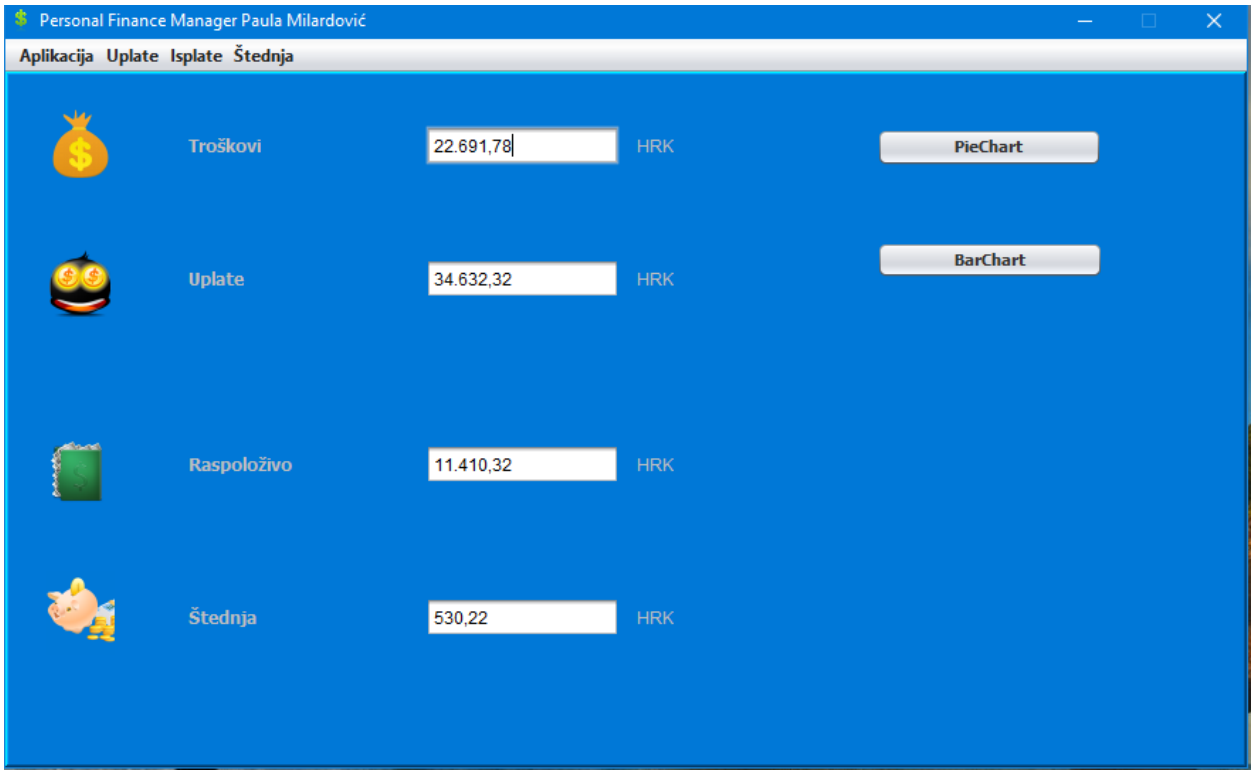
Prilikom kreiranja forme za autorizaciju, automatski se kreira i konstruktor unutra kojega se definiraju radnje koje se odmah izvode prilikom kreiranja objeka te klase, odnosno kod pokretanja programa. Metoda `initComponents()` dolazi od NetBeans razvojnog okruženja te se koristi samo za inicijaliziranje *swing* komponenti korištenih u formi. Klasa `Pomagala` kreirana je kako bi omogućila centriranje prozora i unošenje naziva aplikacije, odnosno kako se ne bi moralo unutar svake forme pisati programski kod za centriranje i naslov, već je dovoljno samo pozvati željene motode spomenute klase (Sl. 4.9).



Sl.4.9 *Klasa Pomagala s pripadajućim metodama*

Nadalje, unutar konstruktora forme `Autorizacija` nalazi si još i metoda `setIcon` koja omogućava mijenjanje Java ikone u obliku šalice kave u željenu Java ikonu. Ista je postavljena unutar konstruktora svih klasa paketa `app.view`. Slika željene ikone nalazi se također unutar paketa `app.view`. Osim postavljanja ikone, koristi se metoda `setText` te se postavlja željen tekst unutar `JTextField-ova` nazvanih `txtKorisnik` i `txtLozinka`. Tekst je postavljen kako bi se izbjeglo stalno unošenje istih podataka pilikom prijavljivanja u aplikaciju. Prema slici 4.8 vidljiva je metoda `btnAutorizirajActionPerformed` koja omogućava obavljanje određene radnje prilikom pritiska na gumb *Login*. Najprije se kreira instanca klase `ObradaAutorizacija` kako bi se ista mogla koristit, zatim se prosljeđuju podaci unešeni u

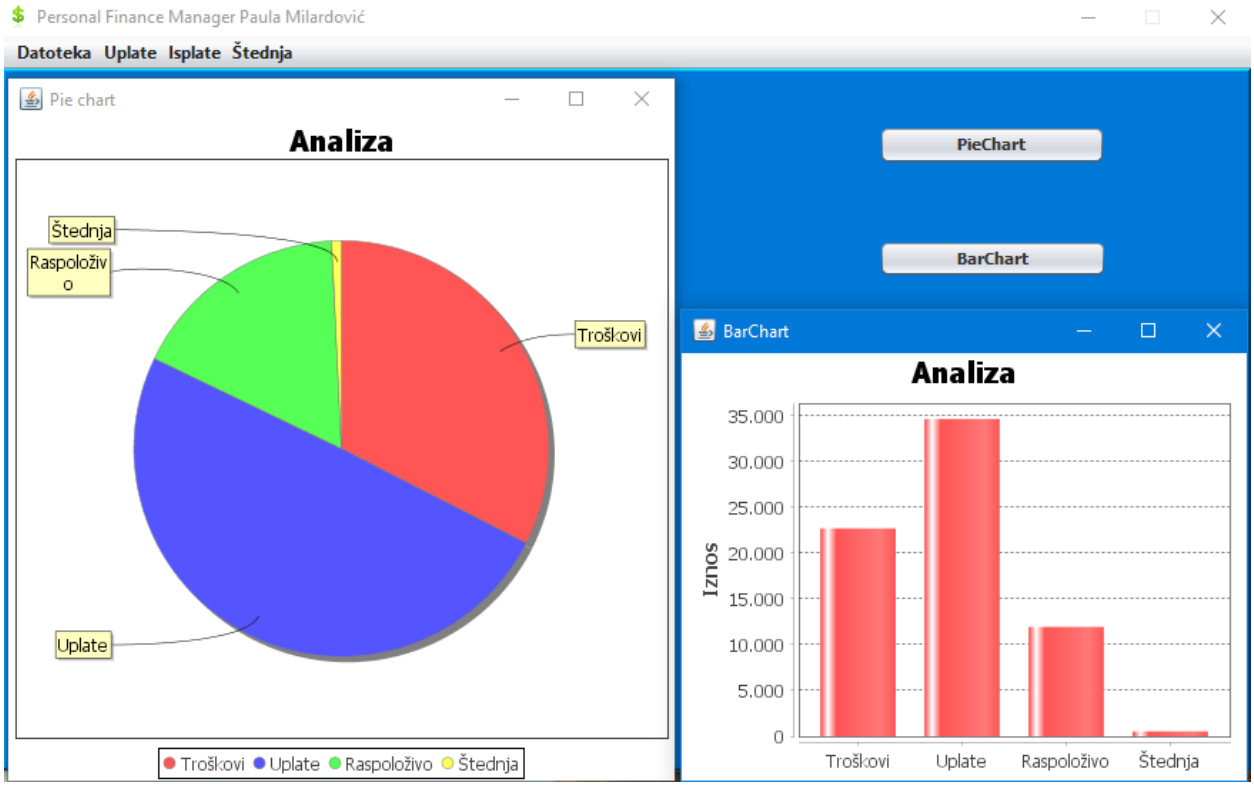
polja korisničko ime i zaporka, a upravitelj SQL izrazom provjerava unešene podatke koristeći model *Operater*, odnosno postoje li u bazi podataka, unutar tablice *Operater* , unešeni parametri. Ukoliko unešeni podaci ne odgovaraju onima u bazi podataka javlja se greška. Za prikaz greške koristi se *swing* komponenta *JoptionPane*. Nakon uspješne autorizacije, otvara se prozor Izbornik.



***Sl. 5.1** Prozor Izbornik*

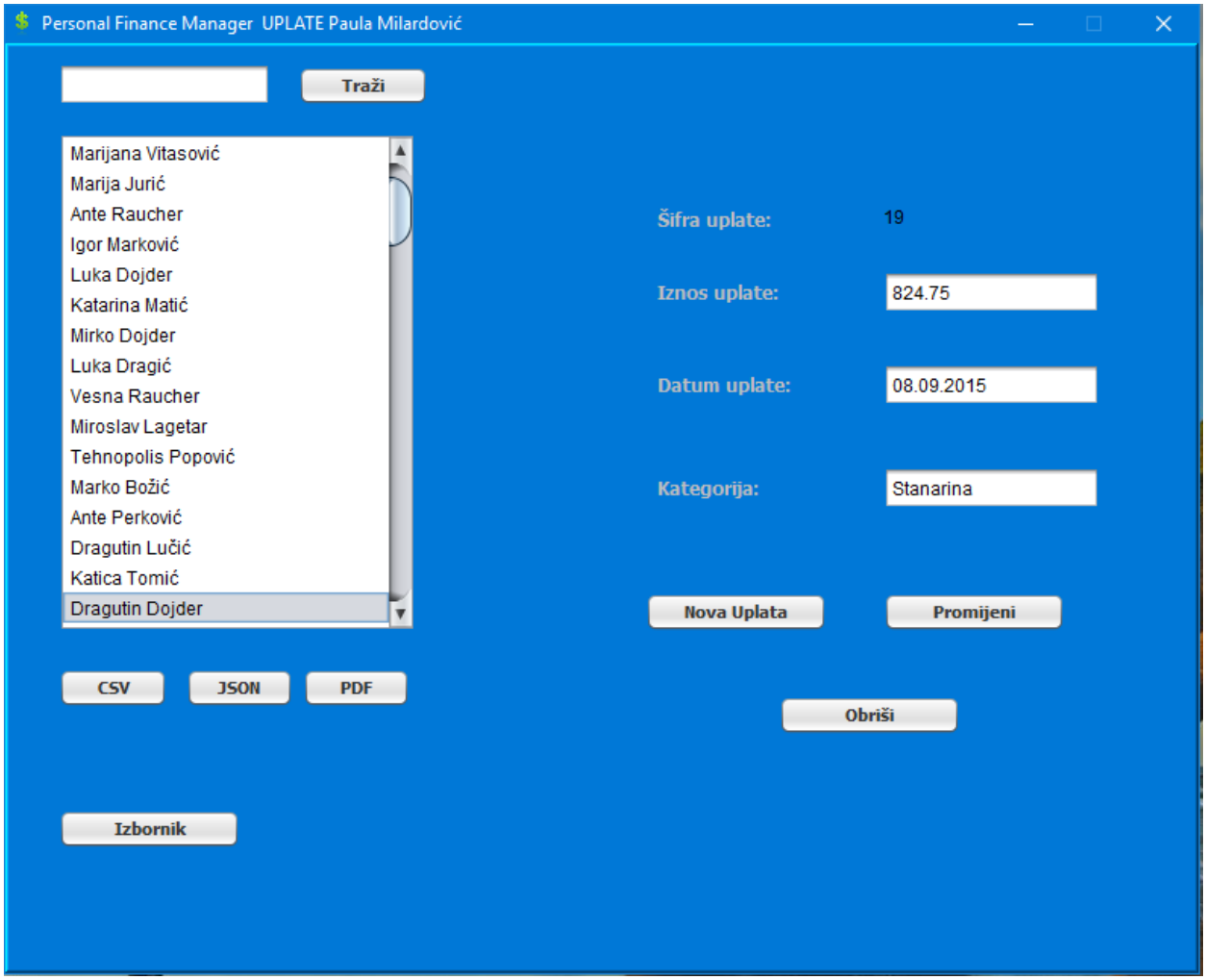
Prema slici 5.1 vidljive su komponente *izbornika*. Isti se sastoji od podataka o ukupnim isplatama s računa, uplatama odnosno prihodima, raspoloživom iznosu te iznosu koji se nalazi u štednji. Unutar tekstualnih polja nalaze se sume brojčanih podataka iz baze. Suma podataka dobivena je jednostavnim SQL izrazom naziva *SELECT SUM(naziv stupca) FROM naziv tablice* unutar upravitelj klase, te se potom vraćeni rezultat stavlja u željeno tekstualno polje. Nadalje, *izbornik* također omogućava prikaz analize spomenutih podataka u obliku tortnog i stupičastog grafa. Pritiskom na gumb *PieChart* ili *BarChart* otvaraju se novi prozori s navedenim grafovima (Sl. 5.2). Na izbornoj traci nalaze se četiri moguće stavke koje korisnik može izabrati. Unutar svake stavke, nalaze se podstavke koje otvaraju određene prozore. Stavka naziva *Aplikacija* sadrži jednu podstavku namijenjenu za izlaz iz aplikacije. Stavke *Uplate* i *Isplate* služe za pregled uplata ili isplata te dodavanje novih, te stavka *Štednja* ima opcije za

pregled transakcija štednje te dodavanje novog iznosa u štednju. Forma `Izbornik` samo olakšava korisniku odabire po potrebi te za spomenutu formu nije potreban upravitelj (engl. Controller) ni model jer se ne vrše upiti na bazu direktno vezani za formu `Izbornik`.



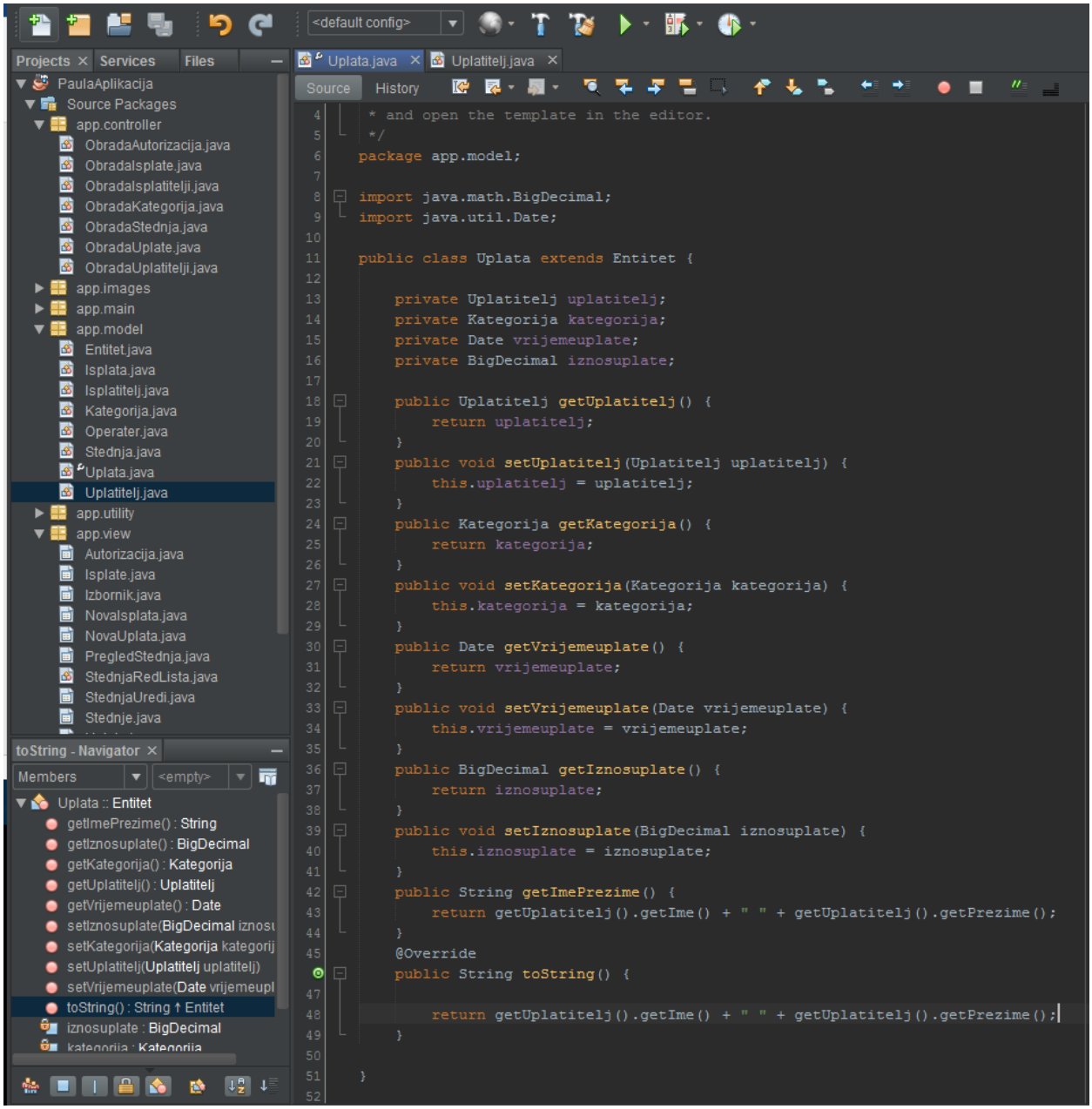
***Sl. 5.2** Prikaz analize podataka u obliku grafova*

Odabirom jedne od stavki izborne trake otvaraju se nove forme. Otvaranjem stavke `Pregled uplatitelja` otvara se prozor s listom postojećih uplatitelja prema imenu i prezimenu i detaljima uplate (Sl. 5.3). Unutar spomenute forme nalaze se podaci iz dvije, međusobno povezane, tablice iz baze podataka, tablica `uplatitelji` i tablica `uplate`. Prema slici 3.1 iz trećeg poglavlja vidljivi su parametri pojedinih tablica. Prema tablicama izrađeni su i pripadajući modeli unutar *paketa* `app.model` naziva `Uplatitelj` i `Uplata` s parametrima koji odgovaraju onima u bazi podataka. Obzirom da tablica uplate u bazi podataka sadrži vanjski ključ uplatitelji, isto je potrebno napraviti i unutar modela `Uplata`, odnosno spomenutoj klasi kao parametar dodati klasu `Uplatitelj` (Sl. 5.4). Prema slici 5.3 vidljivo je da se od parametara uplatitelja prikazuju samo ime i prezime. Prikazivanje željenog sadržaja unutar liste postignuto je korištenjem `toString` metode koja ispisuje imena i prezimena uplatitelja pomoću metoda `getIme` i `getPrezime`. Forma se sastoji od `JListe` koja radnjom `ValueChanged` omogućava mijenjanje pojedinih podataka ovisno o odabranom objektu iz liste.



Sl. 5.3 *Pregled uplata i detalji*

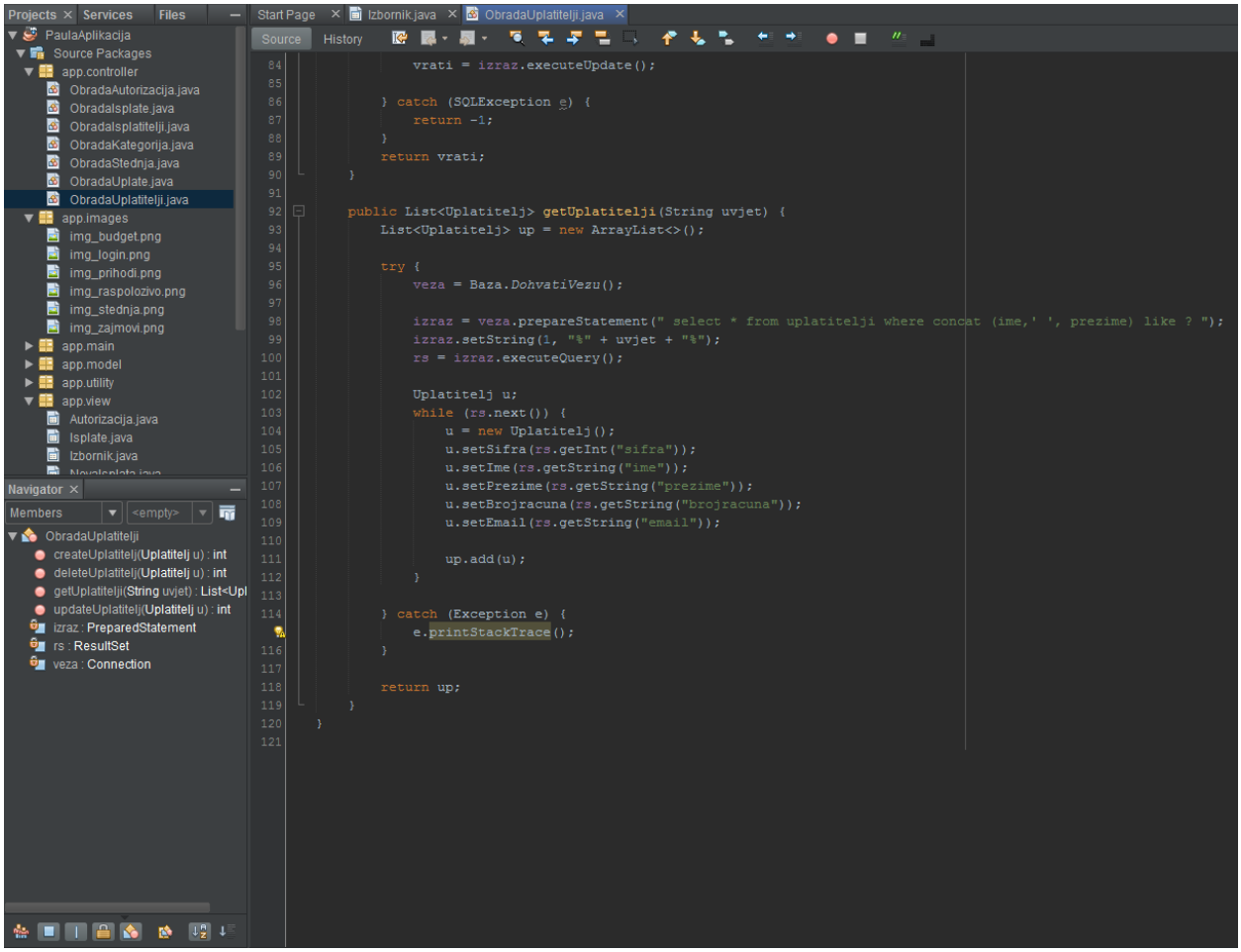
Odabirom određenog objekta iz liste, unutar tekstualnih polja prikazuju se njemu pripadajući podaci iz baze podataka. Osim mogućnosti pregleda, iste je moguće mijenjati, brisati ili dodati novu uplatu. Takva funkcionalnost aplikacije naziva se CRUD (engl. Create, Read, Update, Delete). Obzirom da su tablica `uplatitelji` i tablica `uplate` dvije različite tablice, koje su povezane vanjskim ključem, odnosno tablica `uplatitelji` se veže na tablicu `uplate`, za uspješan ispis podataka iz obje tablice te njihovo brisanje, dodavanje i uređivanje koriste se zasebni SQL izrazi o kojima će više riječi biti u nastavku.



***Sl. 5.4** Model Uplata*

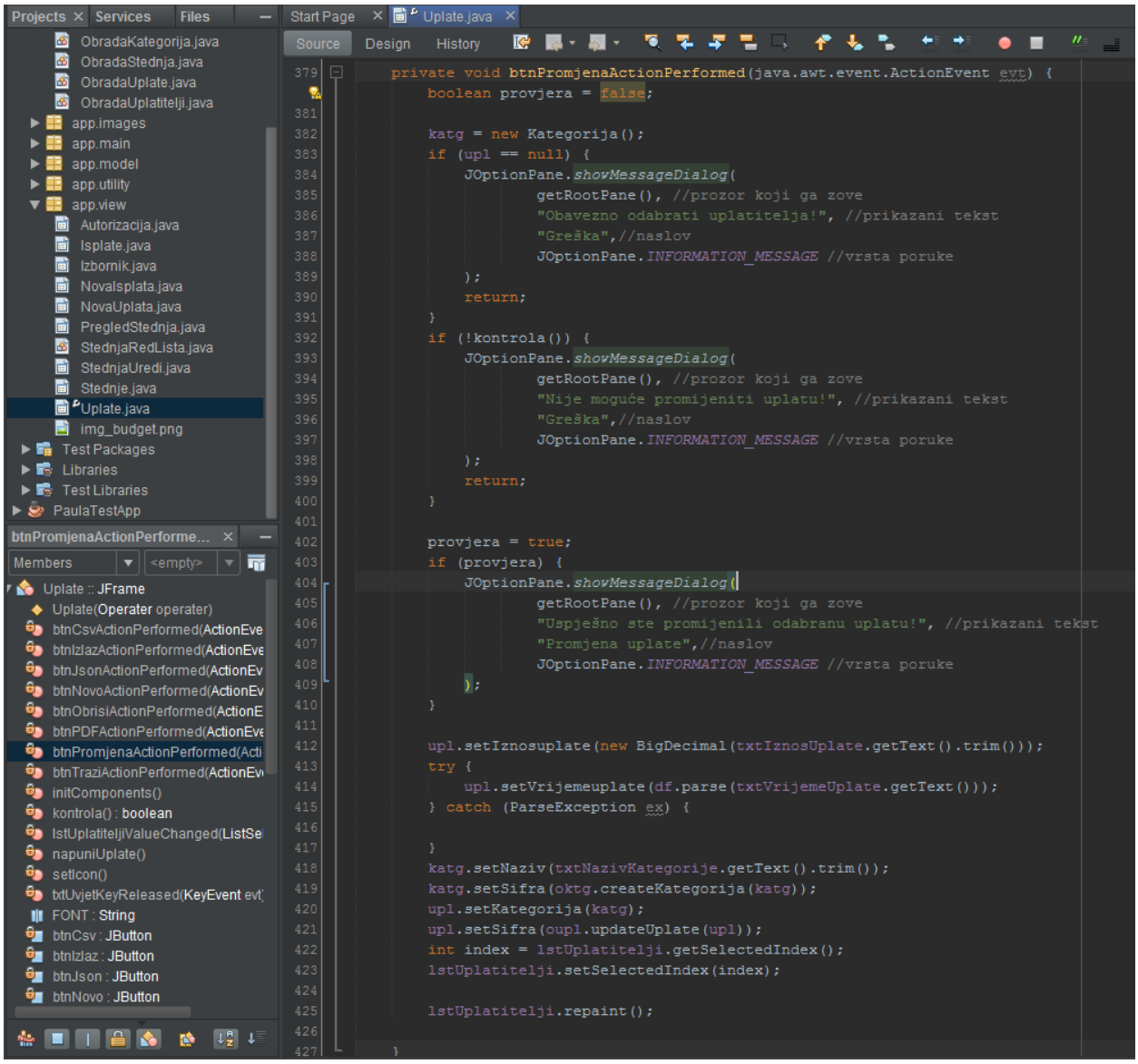
Za ispisivanje podataka iz baze, potrebno je unutar upravitelj klase poslati odgovarajuće upite na bazu koji vraćaju željeni rezultat. Za ispisivanje imena i prezime uplatitelja, unutar klase `ObradaUplatitelji` koristi se SQL izraz `SELECT` kojim se zahtijevaju sva imena i prezime iz tablice `uplatitelji` (Sl. 5.5). Unutar iste klase nalaze se i *SQL* izrazi kojima se podaci unose u bazu, mijenjaju ili brišu. Dodavanje, brisanje ili mijenjanje podataka omogućeno je gumbovima koji su vidljivi na spomenutoj slici. Unutar svake `ActionPerformed` metode gumba nalazi se kod kojim se omogućava željena radnja. Također, postoji i provjera unosa, odnosno da li je

unutar tekstualnih polja unešen odgovarajući tip podataka. Kako bi radnje bile moguće, potrebno je pozvati pripadajuće upravitelj klase unutar kojih se nalaze odgovarajući SQL izrazi (Sl. 5.6)



***Sl. 5.5** Klasa ObradaUplatitelji*

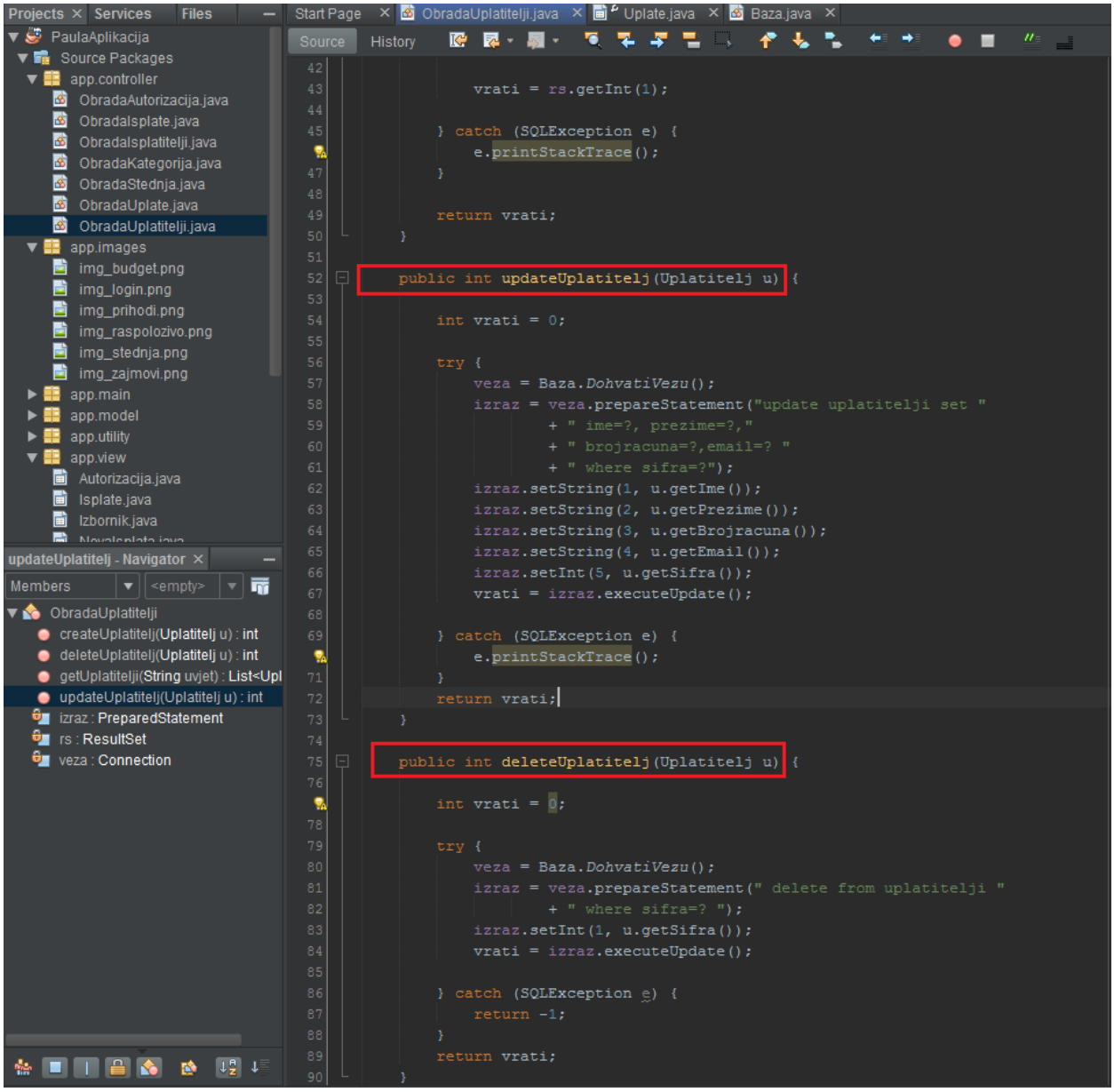
Ukoliko nije odabran uplatitelj iz liste, nije moguće napraviti promjenu i javlja se greška, te se također javlja greška ukoliko dođe do problema s aplikacijom i promjena uplate nije moguća, odnosno ako provjera/kontrola ne prođe. Ukoliko je sve uspješno odrađeno, uplata se mijenja te je promjena vidljiva odmah u aplikaciji. Obzirom da je parametar `vrijemeuplate` tipa `Date`, a *swing* komponenta `JTextField` prima isključivo `String` tip podataka, potrebno je uneseni datum iz `String` tipa podatka pretvoriti (engl. Parse) u `Date` tip podataka kako bi isti bio uspješno promijenjen u bazi. Ista metoda vrijedi i za parametar `iznosuplate` koji je tipa `BigDecimal`.



Sl. 5.6 *Programski kod za promjenu podataka pritiskom na gumb „Promijeni“*

Isti način dodavanja, mijenjanja ili brisanja podataka korišten je i kod ostalih formi. Prema slici 5.5 vidljivo je korištenje `singleton` klase za uspostavljanje veze s bazom te korištenje dosad već spomenutih klasa za slanje SQL izraza. Za brisanje podataka iz baze potrebno je unutar izraza poslati samo parametar `šifru` te se na taj način brišu svi podaci unutar stupca s pripadajućom `šifrom`. Kako bi se određeni podatak promijenio, moraju se poslati svi parametri, obzirom da je sve parametre moguće promijeniti, osim `šifre` koja je jedinstveni ključ stupca svake tablice. Prema slici 5.7 vidljiv je način promjene podataka tablice `uplatitelji`. U tablicama koje sadrže navedenu tablicu kao vanjski ključ, za promjenu podataka potreban je primarni ključ tablice `uplatitelji`. Pravilan način pisanja SQL izraza moguće je pronaći na

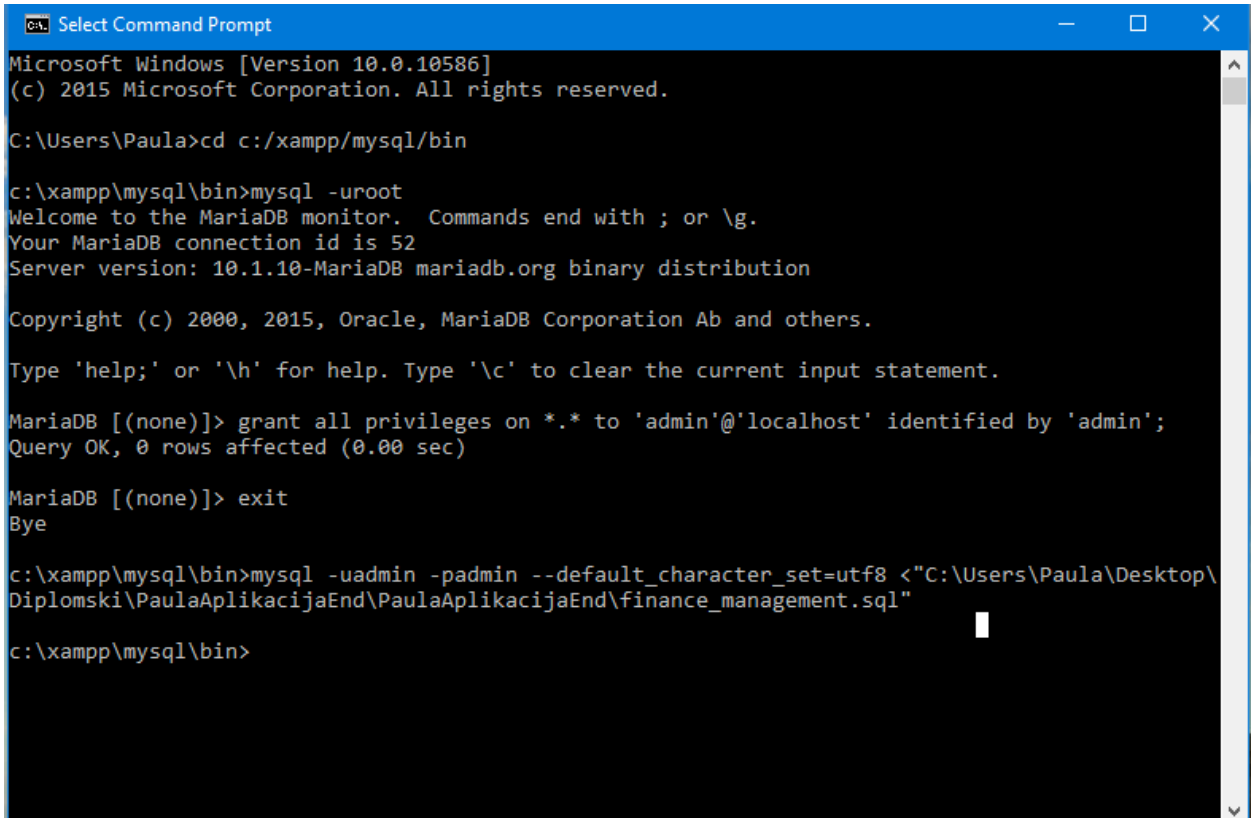
službenoj MySql stranici.



***Sl. 5.7** Brisanje i promjena uplatitelja*

Kao što je već spomenuto, unutar ostalih formi i upravitelj klasa korišteni su isti načini slanja SQL izraza kao u navedenom primjeru. Za provjeru izraza, prije korištenja u aplikaciji, može se koristiti već spomenuti alat MySQL Workbench koji omogućava prikaz rezultata nakon izvođenja SQL izraza. Kroz programski kod aplikacije primjenjuju se osnove objektno orijentiranog programiranja, kreiranje klasa i objekata te korištenje istih. Za izradu grafičko

korisničkog sučelja korišten je skup alata naziva *Swing*. Također, kreira se MySQL baza podataka te primjenjuje unutar aplikacije. Nadalje, kroz programski kod vidljv je i načina spajanja na bazu podataka te upravljanje istima kroz aplikaciju. Aplikacija ima osnovne funkcionalnosti prikazivanja, kreiranja, promjene i brisanja podataka iz baze podataka. Korištena baza podataka nalazi se lokalno, odnosno na računalu, stoga ju je prije korištenja aplikacije potrebno putem naredbenog sučelja (engl. Command Prompt, CMD) i putem *Xampp* multi-platforme ubaciti na MySQL server (Sl 5.8).



Sl. 5.8 *Naredbeno sučelje i prebacivanje baze na mysql server*

Java desktop aplikacija za pregled i vođenje financija još nema svoju potpunu funcionalnost te su potrebne još dodatne komponente kako bi se poboljšao izgled te kreirale i dodatne mogućnosti aplikacije. Trenutačne mogućnosti korisnika su kreiranje, brisanje i mijenjanje podataka, pregled te upravljanje podacima iz baze. Analiza podataka vidljiva je kroz tortni i stupičasti grafikon, te se podaci mogu spremati na računalo u različitim formatima zapisa.

5. ZAKLJUČAK

Aplikacija izrađena u okviru ovog diplomskog rada predstavlja primjer jednostavne Java desktop aplikacije korištenjem *Swing*-a. Iz prikazanog se može zaključiti da se upotrebom navedene biblioteke mogu brzo graditi korisne i prilagodljive aplikacije namijenjene korištenju na različitim računalnim platformama. Upotreba desktop aplikacija je sve manja budući da web aplikacije sve više zamjenjuju desktop aplikacije. Iste se danas koriste u primjerice velikim poduzećima poput Konzuma za potrebe blagajni na stolnim računalima. Prednosti desktop aplikacija su pak da, iako su to aplikacije na stolnim računalima, iste mogu pristupiti podacima putem Interneta, ali su neovisne o njemu, odnosno mogu raditi i ako priključak na Internet ne postoji, ovisno o zahtjevima korisnika. Aplikacije za vođenje financija su sveprisutne u današnje vrijeme, te ih ima mnoštvo što za računala, mobline uređaje, tablete itd. Cilj prilikom izrade aplikacije bio je naučiti osnove objektno orijentiranog programiranja, kreiranje MySQL baze podataka te povezivanje aplikacije s bazom. Iz svega navedenog, može se zaključiti da odabir Jave kao jezika za izradu desktop aplikacija pruža najviše pogodnosti u smislu prenosivosti aplikacije, tj. mogućnosti korištenja na raznim platformama. Java pruža niz biblioteka kojima se kreirana aplikacija može još dodatno proširiti te joj se poboljšati funkcionalnosti što je i jedan od mnogih razlog tolike popularnosti Java programskog jezika u posljednja dva desetljeća. Java programiranje je jedno od najzastupljenijih danas, što potvrđuju informacije s TIOBE Coding Standard kompanije, koja se bavi istraživanjem indeksa popularnosti programskih jezika. Prema spomenutom istraživanju, od nastanka pa do danas Java programski jezik već osam godina drži vodeću poziciju stoga je učenje Java programskog jezika odličan odabir za ulazak u svijet programiranja.

LITERATURA

[1] Oracle Corporation, https://java.com/en/download/faq/whatis_java.xml , Srpanj 2016.

[2] Baltes-Götz B., Götz J., Einführung in das Programmieren mit Java, Srpanj 2012.

[3] Fain Y., Programiranje Java, IT Expert, Srpanj 2016

[4] MySQL, https://www.mysql.com/, Srpanj.2016

[5] MySQL, http://dev.mysql.com/doc/refman/5.7/en/database-use.html, Srpanj.2016

[6] Encyclopedia.com, http://www.encyclopedia.com/doc/1O11-ERAdiagram.html, Srpanj. 2016

[7] PMF Matematički odsjek, https://web.math.pmf.unizg.hr/nastava/rp2/pred1/pred1.html, Srpanj 2016.

[8] Topolnik M., Kušek M., Uvod u programski jezik Java, Lipanj 2008. , Kolovoz 2016.

[9] Wikipedija, https://bs.wikipedia.org/wiki/Java_(programski_jezik), Kolovoz 2016.

[9] TutorialsPoint, http://www.tutorialspoint.com/java/java_object_classes.html, Kolovoz 2016.

[10] Čupić M., Programiranje u Javi, Kolovoz 2016.

[11] NetBeans, https://netbeans.org/, Rujan 2016.

[12] Imtiaz A., Absolute introduction to Object Oriented Programming in Java, Rujan 2016.

[13] http://www.tutorialspoint.com/mvc_framework/mvc_framework_architecture.html, Rujan 2016.

[14] Steve Meloan, JavaOne 2012 Review: Make the Future Java, Rujan 2016.

[15] http://pcchip.hr/softver/korisni/java-sto-kako-i-zasto/, Rujan 2016.

[16] Potter P., How many Java developers are in the world?, Rujan 2016.

[17] http://perl-diving.blogspot.hr/p/blog-page.html, Rujan 2016.

SAŽETAK

Java je objektno orijentiran i univerzalan programski jezik koji se koristi za izradu i razvoj velikog broja aplikacija, odnosno platformski je neovisan. Java program se ne izvršava direktno na operacijskom sustavu računala, nego se pokreće u softverskom okruženju koje se zove Java platforma. Za programiranje u Javi potrebni su dodatni alati naziva Java razvojna oprema (engl. Java Development Kit, JDK) te Java izvedbena okolina (engl. Java Runtime Environment, JRE) za pokretanje i provjeru Java programskog koda. Java Desktop aplikacija rađena u okviru ovog diplomskog rada sastoji se od jednostavne MySQL baze podataka s sedam tablica koje su međusobno povezane vanjskim ključevima. Za izradu grafičko korisničkog sučelja korišten je skup alata naziva *Swing* unutar NetBeans razvojnog okruženja. *Swing* je platformski nezavisna grafička biblioteka za izradu grafičkih korisničkih sučelja Java programa te se velikim dijelom oslanja na Model-View-Controller arhitekturu. Taj tip arhitekture razdvaja podatke predstavljene korisniku od sučelja preko kojeg su mu podaci prezentirani, odnosno olakšava prikazivanje vizualnog dijela aplikacije i omogućava lakše razumijevanje same pozadine aplikacije, odnosno programskog koda. Za povezivanje aplikacije s bazom podataka korišten je skup klasa iz *java.sql* biblioteke te se kreirana baza podataka nalazi lokalno, odnosno na računalu. Aplikacija omogućava pregled, kreiranje, mijenjanje i brisanje podataka.

Ključne riječi: MySQL, era dijagram, objekt, klasa, swing biblioteka, mvc

.

JAVA APPLICATION FOR MANAGING THE FAMILY BUDGET

Java is an object oriented and universal programming language used for developing a large number of applications. Universal means that the programming language is platform independent. A Java program can't be executed directly by the computer's operating system, it runs in a software environment called Java platform. Java programming requires additional tools called Java Development Kit and Java Runtime Environment for running and checking the Java code. Java Desktop application made in the context of this diploma thesis consists of a simple MySQL database with seven tables that are connected by external keys. The graphical user interface has been created by using a set of tools called Swing, within the NetBeans integrated development environment. Swing is platform independent graphical library for creating graphical user interfaces and relies mostly on the Model-View-Controller architecture. This type of architecture isolates the application logic from the user interface layer and supports separation of concerns. The controller receives all requests for the application and then works with the model to prepare any data needed by the view. The view then uses the data prepared by the controller to generate a final presentable response. For connecting the application with the database a set of classes from the *java.sql* library has been used. The created MySQL database is locally on the computer. The application allows you to view, create, modify and delete data.

Keywords: MySQL, era diagram, object, class, swing library, mvc

ŽIVOTOPIS

Paula Milardović rođena je 22.10.1992. godine u Puli, Republika Hrvatska. Osnovnu je školu završila u Čepinu, a I. Gimnaziju je završila u Osijeku 2011. godine. 2011. godine upisuje Elektrotehnički fakultet u Osijeku na kojem je diplomirala u rujnu 2016. godine.

PRILOZI

Uz rad je priložen cd koji sadrži diplomski rad u docx-formatu i pdf-formatu.
Također je priložen programski kod desktop aplikacije u docx-formatu i pdf-formatu, napisan u Java programskom jezku unutar NetBeans razvojnog okruženja.