

**SVEUČILIŠTE U SPLITU  
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I  
BRODOGRADNJE**

**ZAVRŠNI RAD**

**IZRADA MODULA ZA OBRAČUN KAMATE  
PRI OVRŠNOM POSTUPKU**

**Kristijan Pavičić**

Split, rujan 2019



SVEUČILIŠTE U SPLITU  
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I  
BRODOGRADNJE



Preddiplomski stručni studij: **Računarstvo**

Smjer/Usmjerenje: /

Oznaka programa: 550

Akadska godina: 2018./2019.

Ime i prezime: **Kristijan Pavičić**

Broj indeksa: 432-2015


## ZADATAK ZAVRŠNOG RADA

Naslov: **IZRADA MODULA ZA OBRAČUN KAMATE PRI OVRŠNOM POSTUPKU**

Zadatak: U sklopu rada potrebno je opisati razvojno okruženje za izradu modularnih aplikacija. Poseban naglasak treba biti na arhitekturi i korisničkom sučelju. Za „back-end“ aplikaciju koristiti i ukratko opisati: Java , SQL i db2 baze podataka, a za „Front-end“ aplikaciju koristiti i ukratko opisati Javascript, html ,jquery i css. U praktičnom dijelu rada potrebno je opisati i razraditi aplikaciju za testnu okolinu koja na osnovu troška, glavnice i obračunate kamate izračunava naknadu koju je vjerovnik dužan platiti za postupak ovrhe.

Datum obrane: 27.09.2019.

Mentor:

  
prof. dr. sc. Goran Petrović

## IZJAVA

Ovom izjavom potvrđujem da sam završni rad s naslovom IZRADA MODULA ZA OBRAČUN KAMATE PRI OVRŠNOM POSTUPKU pod mentorstvom prof. dr. sc. GORANA PETROVIĆA pisao samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući naveo u završnom radu citirao sam i povezao s korištenim bibliografskim jedinicama.

Student

*Kristijan Pavičić*

Kristijan Pavičić

<b>1</b>	<b>UVOD</b>	<b>1</b>
<b>2</b>	<b>JAVA</b>	<b>2</b>
2.1	Java virtual machine (JVM)	3
2.2	Java je objektno-orijentiran jezik	4
<b>3</b>	<b>C++</b>	<b>6</b>
3.1	C++ compiler i glavne datoteke	6
3.2	Hello World u C++-u	8
3.3	Arhitektura aplikacije u C++-u	9
<b>4</b>	<b>KONEKCIJA SA BAZOM PODATAKA</b>	<b>11</b>
<b>5</b>	<b>OPIS APLIKACIJE</b>	<b>13</b>
5.1	Temeljni zahtjevi aplikacije	13
5.2	Zatezne stope	13
5.3	Vrste zakonskih kamata	13
5.4	Metode obračuna kamate	14
<b>6</b>	<b>IZRADA APLIKACIJE</b>	<b>15</b>
6.1	Upoznavanje sa aplikacijom	15
6.2	Specifikacija baznog djela aplikacija	16
6.3	Odabrane ovisnosti i realizacija istih	19
6.4	Realizacija korisničkog sučelja i metoda koje se vežu na njih	20
<b>7</b>	<b>ZAKLJUČAK</b>	<b>28</b>
	<b>LITERATURA</b>	<b>29</b>
	<b>POPIS OZNAKA I KRATICA</b>	<b>30</b>
	<b>SAŽETAK</b>	<b>31</b>
	<b>KLJUČNE RJEČI</b>	<b>32</b>
	<b>TITLE</b>	<b>33</b>
	<b>SUMMARY</b>	<b>34</b>
	<b>KEYWORDS</b>	<b>35</b>

# 1 UVOD

Poslovni informacijski sustavi su potrebe široke mase ljudi koje konstantno koristimo bez obzira jesmo li tog svjesni. Škole, banke, fakulteti, razne poslovne institucije itd. Kad govorimo o poslovnom informacijskom sustavu prvi i glavni razlog njegovog korištenja jest sigurnost i obrade ogromne količine podataka koje su od strateške vrijednosti.

Za realizaciju velikog poslovnog ustava koriste se *legacy* programski jezici poput C, C++ i Java. Za izradu ove aplikacije koristit ću C++ jezik, u razvojnoj okolini Qt-creator. Kroz ovaj završni rad ću proći i objasniti na koji način sam koristio C++ kao i arhitekturu i logiku aplikacije. Uz to ću spomenuti i objasniti koje zakone aplikacija mora uzeti u obzir.

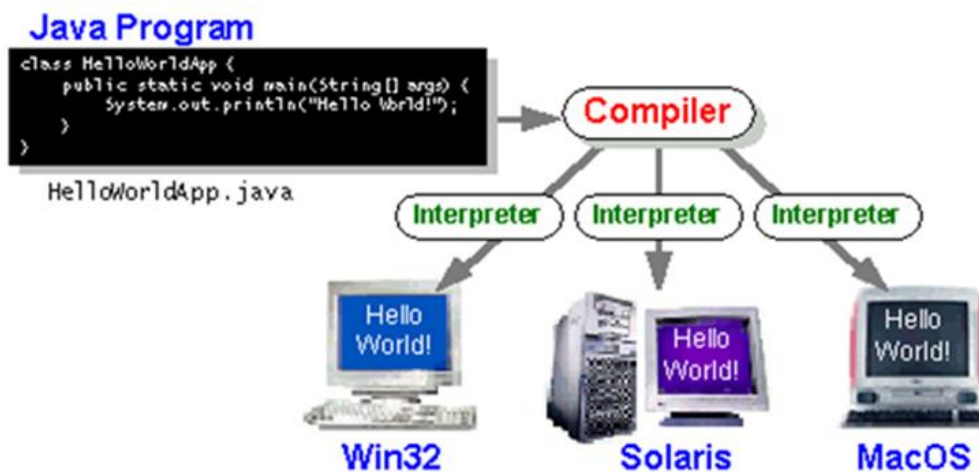
Namjena aplikacije je da službenik Fine može dati točan izračun iznosa kamate za važeću osnovu. Budući da je Fina poslovni sustav koji rukuje sa strateškim važnim podacima prikaz prave aplikacije neće biti moguće te će tu ulogu preuzeti demo zamjenske aplikacije koji ću realizirati i objasniti kroz ovaj završni rad.

## 2 JAVA

Java je jezik opće namjene no najzvučniju primjenu nalazi u programiranju za internet, mobilne aplikacije i velike poslovne informacijske sustave. Neovisno o platformi omogućava kreiranje izvršne datoteke na jednom računalu a izvršavanje na drugom. To je esencijalno za veliki poslovni informacijski sustav kojeg FINA koristi, koji je heterogena okolina sastavljena od vrlo različitih uređaja od osobnih računala, super računala, hardware-a i vrlo različitog software koji kontroliraju rad ovih uređaja. Gdje se Java u FINI-om poslovnom informacijskom sustavu koristi kao „back-end“ upravo zbog mogućnosti da se izvršna datoteka kreira ne jednom a izvršava na drugom uređaju te se zbog tog razloga koriste za obradu podataka, odnosno procesuiranje podataka takozvane "batch-eve". „Klasični“ jezici kao C, C++, Fortran, Pascal prevode se direktno u odgovarajući strojni jezik u ovisnosti od operacijskog sustava i hardware-a. Šalju se direktno centralnom procesoru na obradu. [1]

Java programi se prevode i interpretiraju:

1. prevođenje : Java program se prevodi u „strojni jezik“ za **Javinu virtualnu mašinu** - takozvani *byte code* (Slika 2.1) [1]
2. interpretiranje : JVM izvršava instrukcije zapisane u bytecode-u.



Slika 2.1 Izvođenje programa u Javi

Javin prevodilac ne prevodi izvorni kod u strojni jezik da ga računalno može odmah procesuirati, (ovisan o hardware-u i operacijskom sustavu) već u poseban format, odnosno u byte code. Stoga je potreban sljedeći korak interpretacija iz byte code –a u naredbe koje računalno može interpretirati, interpretacija naredbi se izvršava sa JVM. Takva arhitektura

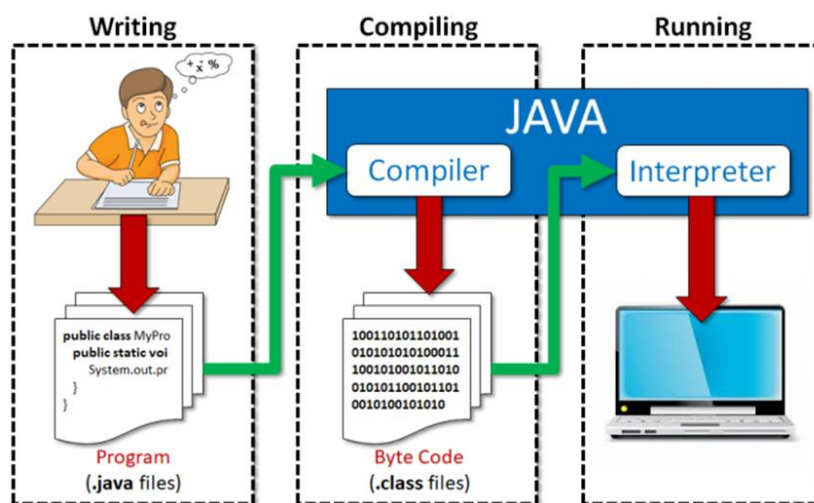
daje neovisnost byte code-a kao što je prikazano na slici, o platformi, odnosno univerzalna je i može se procesuirati na bilo kojem operativnom sustavu, ali kompajliranje pa interpretiranje naredbi smanjuje brzinu izvršavanje programa u odnosu na "klasične" jezike poput C ili C++ gdje se programski kod direktno prevodi i računalo ih izvršava čime se postižu značajne performanse obrade i brzine. JVM izvedena je u software-u (kao ova koju ćemo mi koristiti za Windows platformu) ili u hardware-u ilustracija prevođenja i interpretiranja programa pisanih u Javi. [1]

## 2.1 Java virtual machine (JVM)

JVM(Slika 2.2) je program (obično napisan i C-u ili C++-u) i specifičan je za odgovarajući operacijski sustav. Isto vrijedi za prevodilac. [2]

Windowsi :

- prevodilac : **javac.exe**
- interpreter (tj. JVM) : **java.exe**



Slika 2.2 Java Compiler i Interpreter

**Javac.exe** prevodilac prevodi programski kod odnosno skup instrukcija u strojni kod te ih potom interpreter ili kraće JVM izvodi ili interpretira na računalu te se zadani skup instrukcija u obliku programskog koda pokreće na računalu. Za interpretiranje se koristi **java.exe** izvršni proces koji se odvija u pozadini te je neprimjetan prilikom pokretanja aplikacije jer se odvija u pozadini, gdje se korisniku pokretanje i interpretiranje aplikacije čine kao jedan proces. [3]

## 2.2 Java je objektno-orijentiran jezik

Svi programski jezici rade sa apstrakcijom :

- asemblerski jezici – mala apstrakcija samog hardware-a
- C,Pascal Fortran(proceduralni jezici) – apstrakcija asemblerskih jezika
- daljne apstrakcije : C++ - proceduralan i objektno orijentirani jezik
- Java – čisti objektno orijentirani jezik

Osnovne Karakteristike OOP :

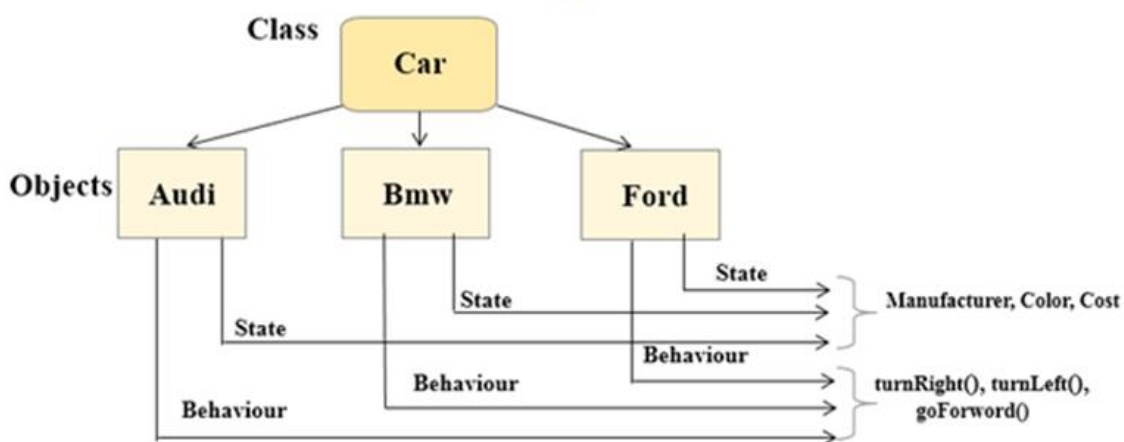
1. **Sve je objekat.** Objektni su zapravo "varijable" koje sadrže podatke, ali im možemo slati zahtjeve u obliku metoda da izvrše odgovarajuću operaciju koje pripadaju tom objektu ili utječu na druge objekte.
2. **Program je skup objekata koji komuniciraju jedni s drugima preko metoda.** Mehanizam je poziv metode koji pripada danom objektu i daje mu određenu funkcionalnost i mogućnost komunikacije.
3. **Objekat ima svoju memoriju,i koja je opet sastavljena od objekata i koju je moguće dinamički alocirati odnosno rezervirati.** Drugim riječima, novi objekti se kreiraju ili instanciraju iz postojećih i alociraju potrebnu „količinu“ memoriju „pakirajućih“ ih na određen način.
4. **Svaki objekt ima tip.** Svaki objekt je instanca neke klase. Sav izvorni kod Jave je podijeljen u klase!
5. **Svi objekti istog tipa mogu primiti iste poruke.** Drugim riječima instance iste klase imaju iste metode, dijele isto ponašanje i stanje.



Objekt :

1. ima podatke
2. ima metode
3. jedinstvenu adresu u memoriji
4. instance konkretne klase npr. sve ptice pripadaju klasi ptice, ali mi uvijek vidimo jednu konkretnu pticu, npr. vrapca, koji je instance klase svih ptica

Na sljedećoj slici(Slika 2.3) je prikazana klasa sa pripadajućim objektima te metodama koji predstavljaju komunikaciju između objekata.



Slika 2.3 Klase i objekti

Klasa služi kao instanca za objekte gdje za primjer vidimo klasu „Car“ iz koje je vidljivo da objekti koji su nastali iz klase „Car“ su automobili različite marke koja ujedno predstavlja njihov identitet. Objekti iste klase dijele isto stanje i ponašanje. Ponašanje svih automobila je da voze te im je zajedničko poput stanja da su vozila za transport. Metode Manufacturer, Color, Cost, predstavljaju stanje objekata, dok metode „turnRight()“, „turnLeft()“ i „goForword()“ predstavljaju ponašanje svih objekata pripadajuće klase. Poput Jave i C++ je objektno orijentiran jezik za kojeg vrijede ista mogućnost implementacije i korištenja klasa i objekata. [3]

### 3 C++

C++ (Slika 3.1) se koristi kao „front-end“ za izradu ovog djela funkcionalnosti aplikacije, C++ za razliku od Java koja je izrazito aplikacijski jezik, uglavnom se koristi za sistemsko programiranje, za pisanje device driver-a i OS-a (operacijskih sustava, na nižoj razini Linux /Unix operacijskog sustava). [7]



Slika 3.1 C++

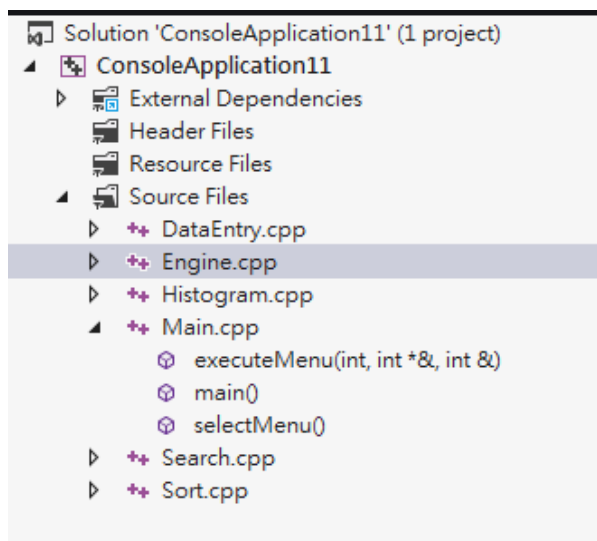
C++ se koristi za „front-end“ zbog toga što pruža bolje performanse uključujući bolju optimizaciju unosnih operacija i brzinu, jer za razliku od Java koja koristi interpreter i kompajler, C++ koristi samo kompajler prevodeći kod direktno u strojni jezik čime postiže bolju brzinu. Za razliku od Java, C++ nije pogodan za obradu velikih količina podataka isključivo jer nema ugrađenu podršku za niti poput Java koja ih podržava te preko njih omogućava obrade različitih podataka i velike količine podataka na različitim nitima dok C++ zahtjeva biblioteku treće strane za podršku niti čime opada produktivnost i kvaliteta koda pogotovo za kompleksne operacije. [4]

#### 3.1 C++ compiler i glavne datoteke

„.cpp“ je naziv ekstenzije (engl. extension) svake C++ datoteke. Poput općenito poznatih ekstenzije „.txt“, „.pdf“, „.doc“, „.docX“ i slično. Primjerice „Programiranje.cpp“ će označavati da se radi o C++ datoteci čija je ekstenzija .cpp i ime „Programiranje“. [5]

Mape datoteka (engl. folder) omogućuju jednostavnije i smislenije grupiranje odnosno bolji pregled i strukturu dokumenata i resursa potrebnih za realizaciju aplikacije za određene potrebe (Slika 3.2), prikaz određenih datoteka poput „Source Code“ gdje se nalaze datoteke sa ekstenzijama „.c“, „.cpp“, i h. Unutar foldera nalaze se četiri unaprijed određene mape koje se

nalaze kao dio predloška koji predstavlja početni temelj izrade aplikacije i generiran je u određenoj programskoj okolini.



*Slika 3.2 Mape datoteka*

„**External Dependencies**“ – u njoj su pohranjene datoteke koje su potrebne za korištenje funkcija unutar C++ jezika, obično su to već gotovi paketi koji uključuju različite datoteke koje u sebi sadrže implementirana programska rješenja za neke potrebe programera pri realizaciji aplikacije. Ovisno o potrebama programera mogu se dodavati, mijenjati i uklanjati datoteke odnosno moguće je i mijenjati cijele pakete datoteka. Kako i samo ime mape kaže, datoteke smještene u ovoj mapi su one esencijalne datoteke o kojima naša aplikacija ovisi. [5]

„**Header Files**“ – u ovu mapu postavljamo vrste datoteka koje imaju ekstenziju „.h“ a koriste se pri imalo zahtjevnijem pisanju koda kao glava ostatku koda gdje su istaknute i deklarirane metode za obavljanje nekih specifičnih operacija kako bi se kompleksne aplikacije s stotinama ili tisućama linija koda rastavile u manje, logične cjeline ili primjerice ukoliko više ljudi istovremeno radi na istoj aplikaciji i svaka osoba radi na određenom dijelu aplikacije čime se postiže bolja preglednost koda koja će omogućiti lakše refaktoriranje i „rukovanje“ kodom. U tom slučaju ne bi imalo smisla koristiti samo jednu „.cpp“ datoteku već je logičnije rastaviti problem u više cjelina kako bi svaka osoba mogla raditi svoj dio posla i na kraju se sve uklopi u veliku cjelinu, čime bi se povećala preglednost koda i gdje bi se možebitne greške odrazile samo na jedan manji dio koda pritom ne utječući na cijeli projekat. [5]

„**Resource Files**“ – mapa koja sadrži resurse bitne za našu aplikaciju. Često u ovu mapu stavljamo datoteke bitne za našu aplikaciju ali koje nisu dio našeg trenutnog projekta to mogu biti čak i slike ili neki jednostavni podaci za poput ikonica ili čak zvučnih zapisa. Primjerice, datoteke koje nisu napisane u C++ jeziku i zahtijevaju prevođenje u C++ jezik se često mogu naći u ovoj mapi koja ne traži da sve datoteke budu iste ekstenzije. [5]

„**Source Files**“ – jedina mapa koja će nama trebati u svakom projektu. U njoj će se nalaziti naše „**.cpp**“ datoteke koje su srž samog programa. Kako i samo ime kaže, to je mapa namijenjena za izvorne datoteke. Naravno, u projekt je moguće dodavanje i vlastitih datotečnih mapa no potreba za takvim postupkom izlazi iz okvira ove skripte i ovisi o stupnju kompleksnosti projekta. [5]

### 3.2 Hello World u C++-u

Naredbe „**#include<iostream>**“ i „**using namespace std;**“ su jedne od mnogih predefiniranih naredbi unutar C++ jezika koje smanjuju vrijeme potrebno da programer obavi neke osnovne funkcije i ubrzava realizaciju aplikacije poput bolje iskoristivosti i uštede koda. Sama (predprocesorska) naredba „**#include**“ govori „programu prevoditelju“ da ćemo uključiti neku datoteku sa odgovarajućom funkcionalnošću koja će nam riješiti određen problem ili olakšati njegovo rješavanje „**<iostream>**“ specificira što točno uključujemo. U ovom slučaju konkretno se radi o datoteci (točni naziv za ovakvu vrstu datoteke je biblioteka (engl. Library) koja u sebi sadrži kratice „**cin**“ i „**cout**“ te služi za ispis ili kao unosna funkcija za podatke bez obzira na tip podatka te ih ispisuje ili unosi u konzolnu aplikaciju na obradu. Značenje ove dvije kratice predstavlja ulazni i izlazni tok konzole (učitavanje iz konzole (cin) i pisanje u konzolu (cout). [5]

Naredba „**using namespace std;**“ nam omogućuje korištenje unaprijed definiranih riječi, funkcija, klasa, objekata a time ujedno skraćuje duljinu koda. Oznaka „**std**“ je kratica za englesku riječ „**standard library**“ i govori programu prevoditelju da ćemo koristiti standardne (predefinirane) izraze za neke operacije. Ukoliko ne bismo napisali „**std**“ naš program prevoditelj ne bi znao što znače neke od ključnih riječi koje ćete upoznati uskoro i morali bismo mu eksplicitno navesti na što točno mislimo čime bi na više mjesta u kodu morali više puta napisati isti kod te bi preglednost i kvaliteta koda bila višestruko smanjena. [5]

Primjerice, tada bi aplikacija umjesto sljedećeg koda (Slika 3.3) :

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Hello World";
    return 0;
}
```

*Slika 3.3 Hello world sa predefiniranim izrazom*

morala biti zapisana na način (Slika 3.4):

```
#include <iostream>
int main()
{
    std::cout<<"Hello World";
    return 0;
}
```

*Slika 3.4 Hello world bez predefiniranog izraza*

Ako se ukloni „**std::**“ dio iz drugog primjera pojavit će se sintaksna pogreška jer prevoditelj ne zna na što točno se odnosi naredba „**cout**“ (iako nije specificirano da koristi biblioteku „**iostream**“ u kojoj je naredba „**cout**“ sadržana). Naredba „**return 0;**“ govori da će glavna (main) funkcija po završetku izvođenja vratiti znamenku „**0**“ (nula). Važno je napomenuti kako nije striktno definirano pravilo da se vrati vrijednost nula (0), ali je običaj kojeg se valja pridržavati vraćena vrijednost ovisi i o vrsti aplikacije koja utječe na to što će „main“ program vratiti. Ova vrijednost ukazuje da nije došlo do pogreške u izvođenju programa odnosno vrati li naša main funkcija neku drugu vrijednost znat ćemo da je došlo do pogreške. „**int main()**“ je glavna funkcija i izvođenje programa počinje s njom. Program može imati samo jednu main funkciju koja predstavlja srce programa bez obzira o kojoj je vrsti aplikacije riječ. Što su funkcije i što predstavlja oznaka int ćemo obrazložiti u nastavku ove skripte. [5]

### 3.3 Arhitektura aplikacije u C++-u

Aplikacija je implementirana u RAD Embacadero Studio 10.3 koji podržava IBM ovu DB2 Enterprise bazu podataka. DB2 Enterprise baza podataka je pod ovlastima Database administratorima koji su zaduženi za organizaciju podataka, svaku izmjenu baze podataka poput kreiranje novih relacija, modifikacija postojećih relacija ili mijenjanje podataka

potrebno je podnijeti zahtjev administratorima koji su zaduženi te imaju potpune ovlasti. Zbog ugovora o privatnosti podataka nisam u mogućnosti prikazati izvornu programsku okolinu i kod kao ni podatke. Te će se potrebe završnog rada realizirati demo verzija aplikacije u zamjenskoj programskoj okolini „Qt-creator“ pri čemu će umjesto izvornog API-ja za spajanje na bazu biti realiziran zamjenski API u Javi koji je ekvivalent pravom API-ju pri čemu će specifikacija tog API u Javi detaljno realizirati u narednoj cjelini. Na Idućoj slici je prikazano zamjensko programsko okruženje gdje će biti realizirana demo aplikacija koja će pri tomu biti ekvivalenta izvornoj aplikaciji prema funkcionalnosti.

## 4 KONEKCIJA SA BAZOM PODATAKA

Za bazu podataka je odgovoran DB2 odjel koji ima sva prava, a za izradu baze podataka za ovu aplikaciju potrebo je podnijeti zahtjev sa relacijama i pripadajućim atributima gdje je ta baza podatka integrirana u već postojeće. Prilikom čega su brzina i optimizacija na potrebnom nivou. Zadržavajući puna administratorska prava, za ovu aplikaciju su omogućena samo read-write prava odnosno unos podataka preko aplikacije te kreiranje upita za unošene podatke. Kako je aplikacija realizirana kao dio „front-end“ sustava aplikacija kao i njezin demo nema bazu podataka već se konekcija(Slika 4.1) ili spajanje vrši direktno na bazu (pretpostavimo da su u toj bazi samo dva polja datum i važeća kamatna stopa za pripadajući datum)

```
1  #include<stdio.h>
2  #include<SQLAPI.h>
3
4  int main(int argc, char* argv[])
5  {
6      // kreiranje objekta koji omogućava konekciju na bazu
7      SAConnection con;
8      SACommand cmd;
9      try
10     {
11         // spajanje na DB2 bazu podataka
12         con.Connect ("Test_BAZA", // ime baze podatka
13                     "kpavic00", // user name
14                     "Sifra01", // password
15                     DB2_Client); //DB Client
16         printf("spojeno!\n");
17         cmd.SetCommandText("SELECT kamatna_stopa,datum_vazenja_stope FROM kamata;");
18         cmd.Execute();
19
20         con.Disconnect();
21         printf("prekid konekcije!\n");
22     }
23
24     catch(SAException & x)
25     {
26         // hvata moguće iznimke
27         try
28         {
29             // roollback funkcija koja u slučaju greške vraća ono stanje kakvo je bilo
30             con.Rollback ();
31         }
32         catch(SAException &)
33         {
34         }
35         // ispisuje grešku koja se javila prilikom konekcije
36         printf("%s\n", (const char*)x.ErrText());
37     }
38     return 0;
39 }
40
```

Slika 4.1 Konekcija i dohvat

Konekcija i dohvat podatka se vrši korištenjem objekta „con“ i metodom „Connect()“ koja se sastoji od niz-a koji sadrži specifikacije o bazi te omogućava konekciju na bazu podataka. Sa objektom „cmd“ i naredbom „setCommandText()“ u kojoj se nalazi sql upit koji dohvaća podatke iz baze(pretpostavimo da su u toj bazi samo dva polja datum i važeća kamatna stopa za pripadajući datum) , sa naredbom „Execut()“ pokreće se dohvat tog istog podatka. Sa naredbom „Disconnect()“ koju koristi objekat „con“ koji ostvaruje konekciju taj isti objekt i završava konekciju. Try Catch u slučaju nekih ne predviđenih problema ili mogućih grešaka hvata i zaustavlja ih te se pokreće „Rollback()“ koji vraća stanje onakvo kako je bilo prije pokretanja trenutne konekcije ili dohvata čime se osigurava neovisnost podataka u bazi podataka



## **5 Opis aplikacije**

### **5.1 Temeljni zahtjevi aplikacije**

Modul obračuna kamate koji se vrši prilikom obrade zaprimljene osnove za plaćanje ako ima sredstava na ovršenikovim računima, te prilikom izvršenja osnove kad stigne priljev na blokirani račun ovršenika. Također se obračun vrši uvijek kad je na odabrani datum potrebno znati obavezu po osnovi u redoslijedu za plaćanje, odnosno kad je potreba na određeni datum znati ukupnu obavezu po blokiranom ovršeniku. Obračun kamate se vrši kako bi se utvrdio iznos obaveze po osnovi na datum završetka obračunskog razdoblja. Obračun kamate se vrši samo ako je to osnovom zahtijevano.

### **5.2 Zatezne stope**

Kamatne zatezne stope se mogu podijeliti u dvije grupe:

- Zakonske zatezne kamatne stope
- Stope pojedinih banaka, eskontne stope uvećane ili umanjene, različite ugovorne stope i slično koje čine grupu pomoćnih kamatnih stopa
- Svaka grupa ima više vrsta kamatnih stopa, a svaka vrsta je u sastavu jednoznačno definirana šifrom vrste kamatne stope.
- Pojedina stopa je određena visinom stope i metodom obračuna po vremenskim razdobljima.

### **5.3 Vrste zakonskih kamata**

Vrste zakonskih zateznih kamatnih stopa :

1. Do 01.01.2008. godine koriste se kamatne stope :
  - a. '1' zatezna do 07.10.89. čl.277 st. 1
  - b. '2' zatezna do 07.10.89. čl.277 st. 2
  - c. '3' kamata na javne prihode
  - d. '6' carina
  - e. '7' zatezna po ZFPPN
2. nakon 01.01.2008. godine vrtu kamatne stope određuje ugovorni odnos između ovršenika i ovrhovoditelja, pa su moguće šifre stopa :

- a. '1' trgovac      visina kamatne stope = eskontna stopa + 8
- b. '2' netrgovac   visina kamatne stope = eskontna stopa + 5
- c. 'xx' poduzetnik

3. Ako je      odabrana "Pomoćna kamatna stopa" pripadajući izbornik "Šifra kamatne" stope sadržavat će šifru i naziv kamatne stope ovisno o šifri poslovne jedinice.

Visinu eskontne stope zadaje HNB.

Ako se osnovom zahtjeva obračun kamate sa zateznom kamatnom stopom, a obračunskim period započinje prije 01.01.2008. godine potrebo je prilikom unosa podataka odabrati jednu od stopa koje vrijede do 01.01.2008. godine, a zatim jednu od stopa koja se odnosi na razdoblje nakon toga. Ako obračunski period počinje nakon 01.01.2008. godine, unosi se samo jedna od stopa koja vrijedi tog datuma.

#### 5.4 Metode obračuna kamate

Postoje dvije metode obračuna kamate !

Konformna metoda:

$$IK = \left( \left( (1 + KS)^{\frac{P}{BDG}} \right) - 1 \right) * G \quad (5.1)$$

Proporcionalna metoda:

$$IK = KS * \frac{P}{BDG} * G \quad (5.2)$$

Gdje su :

- IK** izračunati iznos kamate
- P** broj dana u obračunskom razdoblju
- BDG** broj dana u godini
- G** iznos glavnice na koji se obračunava kamata
- IK** izračunati iznos kamate

## 6 IZRADA APLIKACIJE

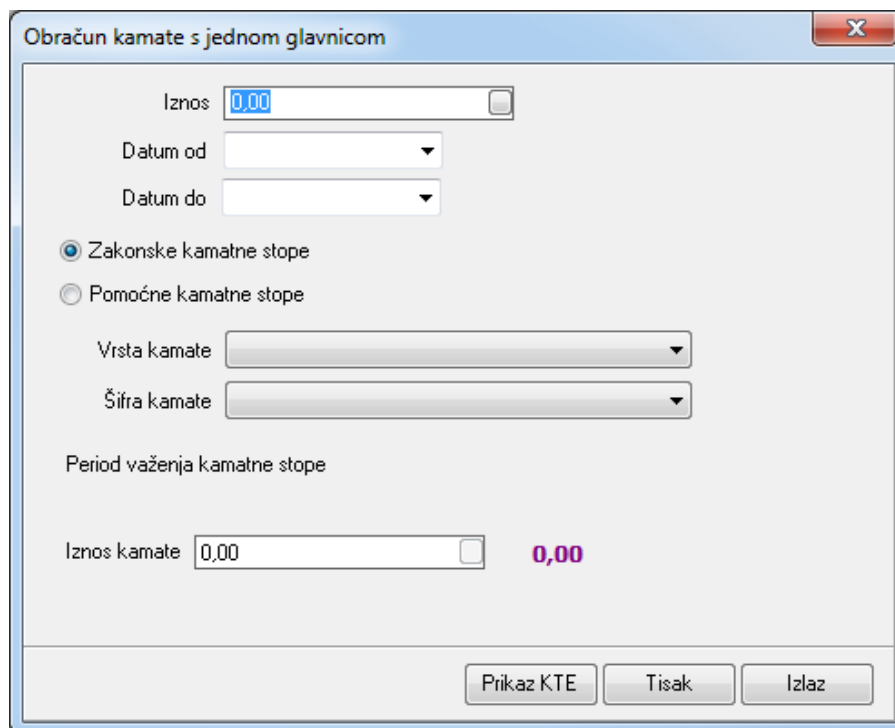
### 6.1 Upoznavanje sa aplikacijom

Aplikacija čija je glavna funkcionalnost obračun kamate pri ovršnom postupku realizirana je zamjenskoj razvojnoj okolini Qt-creator-u(Slika 6.1) umjesto u IBM-ovoj Embacadero razvojnoj okolini. Aplikacijsko sučelje je namijenjeno osobama koje rade u Fininim institucijama tj. koje su smještene na šalterima za unos osnova u sustav te ih pripremaju za procesuiranje.



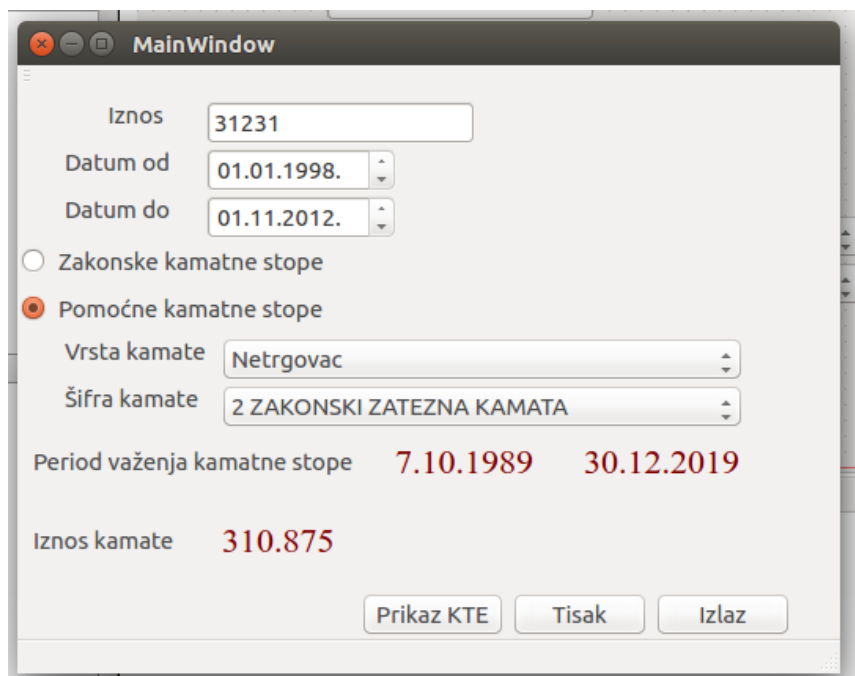
*Slika 6.1 Qt-creator*

Nakon unosa iznosa, odabira datuma i ostalih parametara osnove dobije se iznos kamate koja se potom sa pripadajućim podacima upisuje i bilježi u DB2 bazi podataka. Kod s kojim je napisana i razrađena izvorna aplikacija je poslovna tajna te će od nje biti prikazan samo korisničko sučelje(Slika 6.2).

The image is a screenshot of a software application window titled 'Obračun kamate s jednom glavnicom'. The window has a light blue title bar with a close button (X) in the top right corner. The main area is white and contains several input fields and controls. At the top, there is a text box labeled 'Iznos' with the value '0.00' and a small square icon to its right. Below this are two dropdown menus labeled 'Datum od' and 'Datum do'. Further down are two radio buttons: 'Zakonske kamatne stope' (selected) and 'Pomoćne kamatne stope'. Below the radio buttons are two more dropdown menus labeled 'Vrsta kamate' and 'Šifra kamate'. Under these is the text 'Period važenja kamatne stope'. At the bottom left, there is a text box labeled 'Iznos kamate' with the value '0.00' and a small square icon to its right. To the right of this text box, the value '0,00' is displayed in red. At the bottom right of the window, there are three buttons: 'Prikaz KTE', 'Tisak', and 'Izlaz'.

*Slika 6.2 Korisničko sučelje izvorne aplikacije*

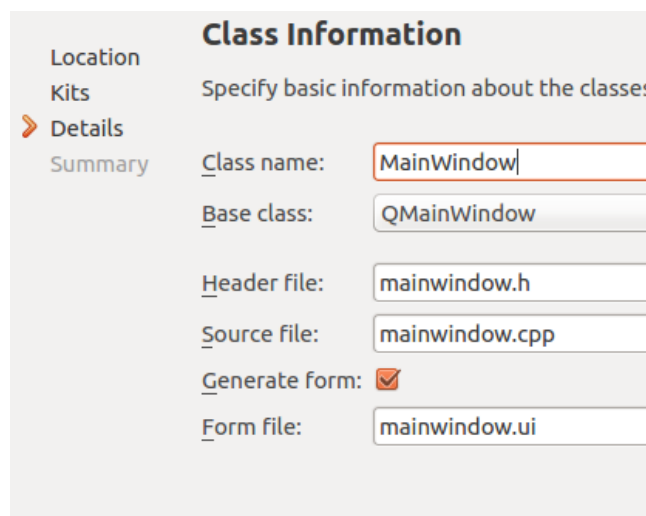
Kao što je vidljivo aplikacija je desktop aplikacija koja sadrži različite mogućnosti izračuna kamate, poput izvorne aplikacije demo aplikacija koja je pisana u istom programskom jeziku ali u različitoj razvojnoj okolini sadrži iste funkcionalnosti(Slika 6.3) uz male vizualne preinake isključivo zbog razvojne okoline. [10]



*Slika 6.3 Demo verzija aplikacije*

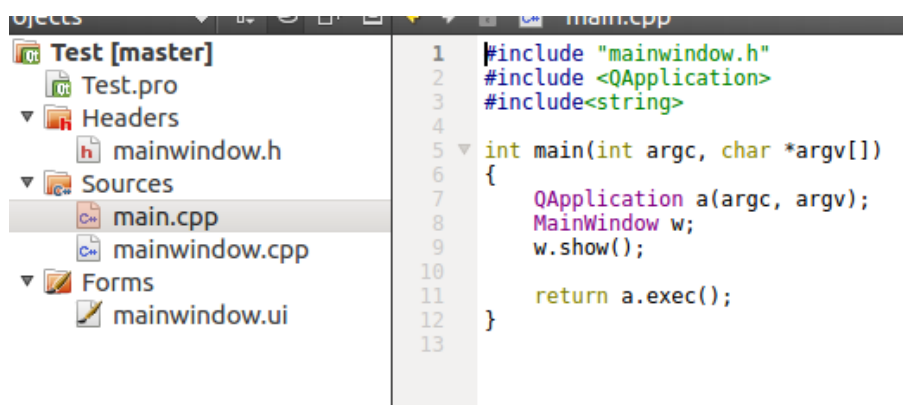
## 6.2 Specifikacija baznog djela aplikacija

Pri samoj izradi aplikacije potrebno je detaljno specificirati(Slika 6.4) izbor određenog predloška koji će nam služiti kao temelj izrade naše aplikacije.



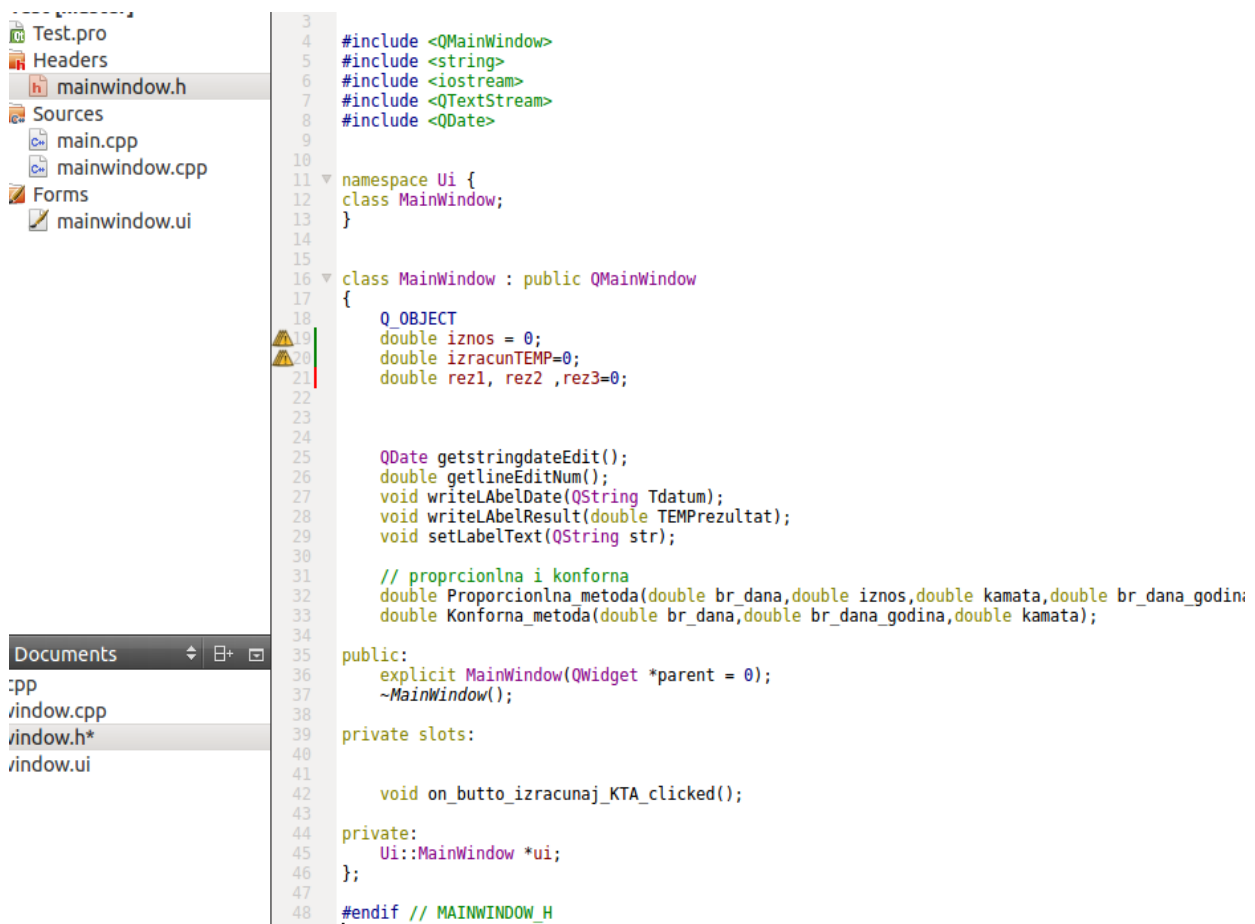
*Slika 6.4 Specifikacije odabranog predloška*

Za ime klase je odabrana klasa i njezin naziv Main Window na osnovu koje ćemo stvarati potrebne objekte i pozivati određene metode i realizirati implementaciju određene funkcije. Bazna klasa i ostali nazivi Head, Source i Forme kao i bazne klase su istog imena kao i odabrana klasa samo što imaju različitu ekstenziju. U prethodnim poglavljima je detaljno specificirano i objašnjeno značenje i funkcije spomenutih mapa(Slika 6.5) kao i njih položaj u aplikaciji. Urednost aplikacije ne ovisi samo o urednosti koda, već i o raspodjeli po datotekama kao i po paketima u ovisnosti o potrebama naše aplikacije. Raspodjela na pakete omogućava da netko tko nije sudjelovao u izradi projekta, ima olakšani pristup aplikaciji kao i procesima izrade aplikacije.



*Slika 6.5 Mape projekta*

U Headers(Slika 6.6) mapi sa ekstenzijom .h, se nalaze inicijalizacije svih važnih metoda kao i imena globalnih varijabli.



*Slika 6.6 Headers map*

U Sources mapi nalaze se datoteke main i mainwindow sa ekstenzijom .cpp gdje se nalazi kod koji izvršava instrukcije koje je odabrao korisnik aplikacije. U datoteci main(*Slika 6.7*) se izvršava poziv koji samo poziva i omogućuje prikaz i izvršavanje aplikacije i dio je izabranog predloška, dok se u datoteci „mainwindow“ nalaze metode koje izvršavaju potrebne funkcionalnosti koje će biti podobnije obraćene i specificirane u nastavku završnog rada.

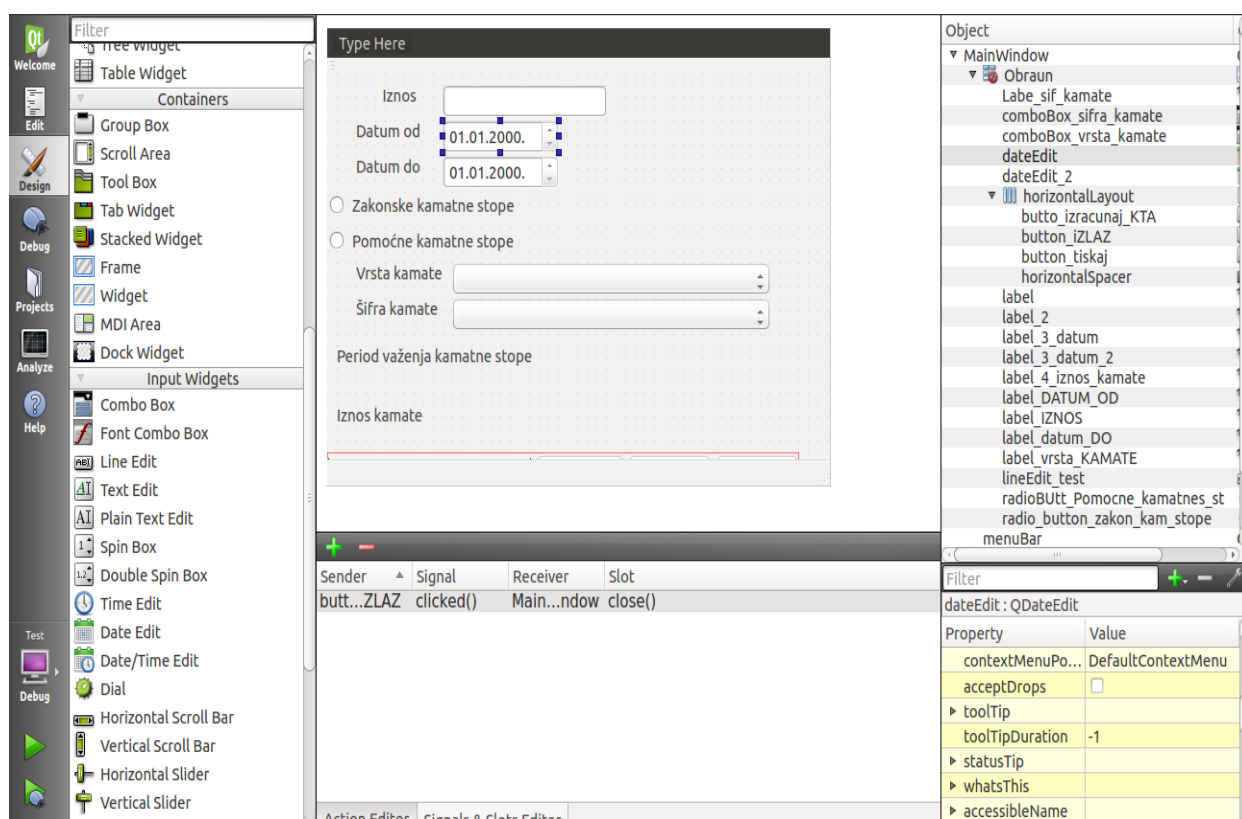
```

4
5 int main(int argc, char *argv[])
6 {
7     QApplication a(argc, argv);
8     MainWindow w;
9     w.show();
10
11     return a.exec();
12 }
13

```

*Slika 6.7 Main aplikacije*

„Form“ mapa(*Slika 6.8*) sa ekstenzijom „ui“ se odnosi na GUI sučelje te odabir određenih metoda sučelja.



Slika 6.8 Form datoteka

### 6.3 Odabrane ovisnosti i realizacija istih

Kao što je vidljivo glavne datoteke koje sadrže skoro cijeli kod aplikacije su „mainwindow.cpp“ koji se odnosi na funkcionalnost i omogućava izvršavanje odabranih mogućnosti, dok datoteka „mainwindow.ui“ odnosno GUI predstavlja korisničko sučelje i logički su povezane te će biti paralelno obrađene jer ih je nemoguće odvojiti. Svaki parametar aplikacije će biti specificiran kroz GUI sučelje kao i kroz logiku i programski kod. Prije svega je vidljivo da su sve mogućnosti koje se nalaze u GUI sučelju (Slika 6.9) specificirane i navedene sa jedinstvenim „id-jem“ pomoću kojeg im se u mainwindow.cpp datoteci daju omogućuju i implementiraju te ih se na neki način “oživljava” i postaju potpuno funkcionalne za korisničku upotrebu te ih korisnik vidi kao jednu cjelinu.

Object	Class
▼ MainWindow	QMainWindow
▼ Obraun	QWidget
Labe_sif_kamate	QLabel
comboBox_sifra_kamate	QComboBox
comboBox_vrsta_kamate	QComboBox
dateEdit	QDateEdit
dateEdit_2	QDateEdit
▼ horizontalLayout	QHBoxLayout
butto_izracunaj_KTA	QPushButton
button_iZLAZ	QPushButton
button_tiskaj	QPushButton
horizontalSpacer	Spacer
label	QLabel
label_2	QLabel
label_3_datum	QLabel
label_3_datum_2	QLabel
label_4_iznos_kamate	QLabel
label_DATUM_OD	QLabel
label_IZNOS	QLabel
label_datum_DO	QLabel
label_vrsta_KAMATE	QLabel
lineEdit_test	QLineEdit
radioBUtt_Pomocne_kamatnes_st	QRadioButton
radio_button_zakon_kam_stope	QRadioButton
menuBar	QMenuBar
mainToolBar	QToolBar
statusBar	QStatusBar

*Slika 6.9 Id-jevi GUI mogućnosti*

## 6.4 Realizacija korisničkog sučelja i metoda koje se vežu na njih

Parametri za izračun:

1. Iznos za koji se računa kamata u GUI sučelju se sastoji od „labela“ „Iznos“(Slika 6.10)



*Slika 6.10 GUI Unos*

te od „lineEdit“, (Slika 6.11) koji omogućava unos dok metoda „getLineEditNum()“ hvata taj unos i omogućuje obradu tog unosa, metoda „writeLabelResult()“ sa parametrom „TEMPRezultat“ uzima obrađeni unos na osnovu svih parametara te omogućuje ispis iznosa kamate.



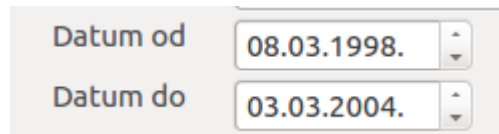
```

75
76
77 double MainWindow::getlineEditNum(){
78     iznos = ui->lineEdit_test->text().toDouble();
79
80 }
81 void MainWindow::writeLabelResult(double TEMPRezultat) // ova ne radi!!
82 {
83     ui->label_4_iznos_kamate->setText(QString::number(TEMPRezultat));
84     ui->label_4_iznos_kamate->setStyleSheet("font-size: 22px;font-family: Times New Roman;color: rgb(128,0,0)");
85     izracunTEMP = 0;
86 }
87

```

*Slika 6.11 Metode za parametar unos*

2. Vremenski period računanja kamate (datum od kada se računa i datum do kada se računa) a sastoji se od dva „labela“, „Datum od“ i „Datum Do“, (Slika 6.12), atributima „dateEdit“ i „dateEdit\_2“ se dohvaćaju datumi pretvaraju u „string“,



*Slika 6.12 GUI unos za datum*

„dateEdit“ i „dateEdit\_2“. (Slika 6.13) se dohvaćaju datumi pretvaraju u stringove tj. „QString“ i omogućuje se dohvat podataka kao i njihovu obradu.

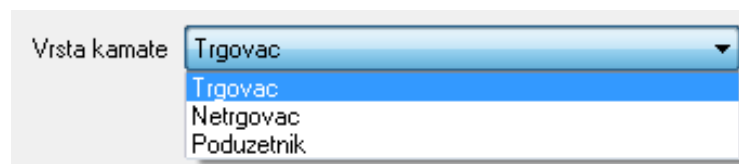
```

QString DatumOD = ui->dateEdit->text();
QString DatumDO = ui->dateEdit_2->text();

```

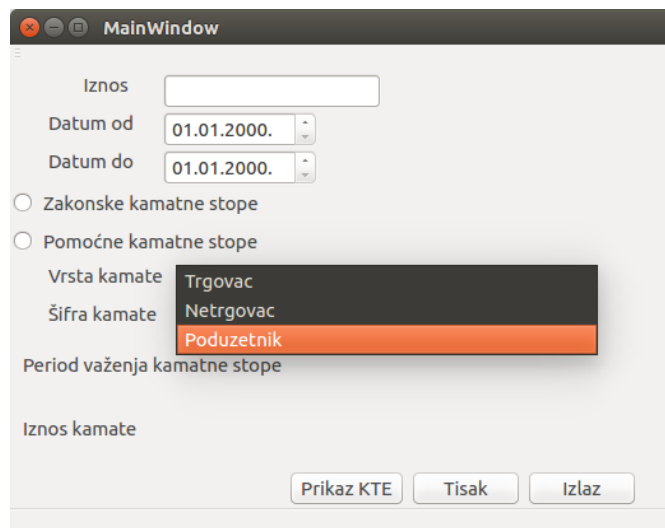
*Slika 6.13 Metode za hvatanje datuma kao parametara*

3. Vrsta kamatne stope (zakonska ili pomoćna), koja je obrađena i detaljno specificirana u jednom od prethodnog poglavlja, za implementaciju u GUI korisničkom sučelju (Slika 6.14)



*Slika 6.14 Vrsta kamate u izvornoj aplikaciji*

kako u izvornoj aplikaciji tako je korišteno i u demo verziji (Slika 6.15) aplikacije.



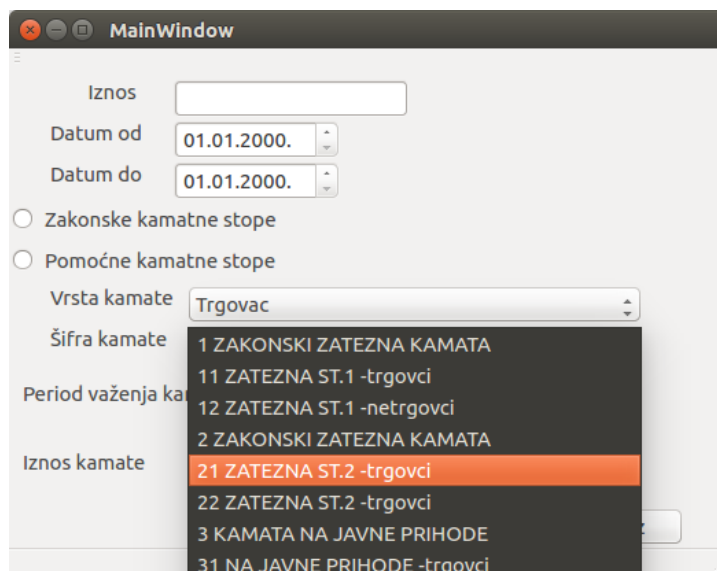
*Slika 6.15 Vrsta aplikacije u demo verziji*

Korišten je „comboBox“, (Slika 6.16) te je implementiran i realiziran u programskom kodu sa korištenjem pripadajućih metoda.

```
// drop dow lista za vrstu kamate
ui->comboBox_vrsta_kamate->addItem("Trgovac");
ui->comboBox_vrsta_kamate->addItem("Netrgovac");
ui->comboBox_vrsta_kamate->addItem("Poduzetnik");
```

*Slika 6.16 Metode za hvatanje odabira iz padajućeg izbornika za vrstu kamate*

Na sličan način je realizirana i šifra kamate (Slika 6.17) kako kroz GUI sučelje gdje je jedina razlika u



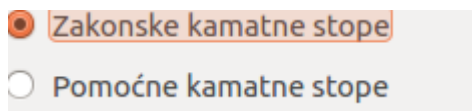
*Slika 6.17 GUI za šifru kamate*

implementaciji u kodu bila ta što je padajući izbornik za šifru kamate(Slika 6.18) sadržavao više opcija.

```
// drop dow lista za sifru kamate
ui->comboBox_sifra_kamate->addItem("1 ZAKONSKI ZATEZNA KAMATA");
ui->comboBox_sifra_kamate->addItem("11 ZATEZNA ST.1 -trgovci");
ui->comboBox_sifra_kamate->addItem("12 ZATEZNA ST.1 -netrgovci");
ui->comboBox_sifra_kamate->addItem("2 ZAKONSKI ZATEZNA KAMATA");
ui->comboBox_sifra_kamate->addItem("21 ZATEZNA ST.2 -trgovci");
ui->comboBox_sifra_kamate->addItem("22 ZATEZNA ST.2 -netrgovci");
ui->comboBox_sifra_kamate->addItem("3 KAMATA NA JAVNE PRIHODE");
ui->comboBox_sifra_kamate->addItem("31 NA JAVNE PRIHODE -trgovci");
ui->comboBox_sifra_kamate->addItem("32 NA JAVNE PRIHODE -netrgovci");
ui->comboBox_sifra_kamate->addItem("6 CARINA");
ui->comboBox_sifra_kamate->addItem("61 CARINA -trgovci");
ui->comboBox_sifra_kamate->addItem("62 CARINA -netrgovci");
ui->comboBox_sifra_kamate->addItem("7 ZATEZNA PO ZFPPN");
ui->comboBox_sifra_kamate->addItem("73 ZATEZNA PO ZFPPN -poduzetnik");
ui->comboBox_sifra_kamate->addItem("D OBRACUN ZASEBNO-DEVIZNA I SL.");
```

*Slika 6.18 Metode za šifru kamate*

Potrebno je spomenuti opciju gdje korisnik pri unosu izabire opciju sa GUI metodom(Slika 6.19), „rado button“ odabire „Zakonsku“ ili „Pomoćnu“ kamatnu stopu.



*Slika 6.19 GUI sučelje za Zakonsku i Kamatnu*

Te su time obrađeni unosni parametri na osnovu kojih se izračunava(Slika 6.20) iznos kamate. Sa metodom koja se logički povezana sa ikonicom „prikaz KTE“, ikonica „Tisak“ ispisuje rezultate aplikacije na pisanom dokumentu ali na demo verziji aplikacije nije realizirana jer za njom



*Slika 6.20 Ikonice*

nije bilo potrebe. Ikonica „Izlaz“ je realizirana i ima funkcionalnost da se klikom na ikonicu „Izlaz“ aplikacija zaustavi i prekine sa izvršavanjem. Što je u programu realizirano kroz sljedeću metodu „Sender“ odnosno gumb koji šalje signal koji je metoda „clicked()“ signal(Slika 6.21) prima „funkcionalnost“ koji koristi prorez tj. „Slot“ „close()“ koji na klik zatvara aplikaciju.

Sender	Signal	Receiver	Slot
butt...ZLAZ	clicked()	Main...ndow	close()

Slika 6.21 Metoda za izlaz

Metoda “Prikaz KTE” se dobije kad se ulazni parametri obrade odnosno da bi bolje razumjeli kad se koji parametri koriste i dobivaju potrebno je razumjeti logiku aplikacije te će prvo biti prikazan kod za metode računanja kamatnog iznosa. U ovisnosti o vremenskom intervalu za Proporcionalnu metodu se izračunavaju broj dana koje je korisnik unio(Slika 6.22).

```

146 // !!! broj dana između unosnih DATUMA !!!!!!!!!!!!!!!
147 //prvi datum
148 QStringList datem = DatumOD.split(".");
149 double dani_datum1 = QString(datem[0]).toDouble();
150 double mjeseci_datum1 = QString(datem[1]).toDouble();
151 double godine_datum1 = QString(datem[2]).toDouble();
152
153 double datum1_ukupno_dani = dani_datum1 +(mjeseci_datum1*30)+(godine_datum1*365);
154
155 //DRUGI datum
156 QStringList datem2 = DatumDO.split(".");
157 double dani_datum2 = QString(datem2[0]).toDouble();
158 double mjeseci_datum2 = QString(datem2[1]).toDouble();
159 double godine_datum2 = QString(datem2[2]).toDouble();
160
161 double datum2_ukupno_dani = dani_datum2 +(mjeseci_datum2*30)+(godine_datum2*365);
162
163 double broj_dana_razdoblja = datum2_ukupno_dani - datum1_ukupno_dani;
164

```

Slika 6.22 Metoda za izračun broja dana između datuma koje je korisnik unio

Kamatna stopa koja se dobije nakon što se izaberu opcije vrste i šifre kamate te izbora zakonske ili pomoćne

```

60 double MainWindow::Proporcionalna_metoda(double br_dana,double iznos,double kamata,double br_dana_godina)
61 {
62
63     return rez3 =kamata*(br_dana/br_dana_godina)* iznos;
64

```

Slika 6.23 Realizacija Proporcionalne metode

kamatne stope. Parametar broj dana u razdoblju važenja kamatne stope dobije se izračunom broja dana u periodu(Slika 6.24) u kojem važi ta kamatna stopa. Pozivom metode „Proporcionalna\_metoda“(Slika 6.23) koja uzima dosad navedene parametre te još i iznos

```

123 // !!! broj dana između dva intervala datuma !!!!!!!!!!!!!!!
124 //prvi datum
125 QStringList datum1 = poc_INTERVALA.split(".");
126 double dan_datum1 = QString(datum1[0]).toDouble();
127 double mjesec_datum1 = QString(datum1[1]).toDouble();
128 double godina_datum1 = QString(datum1[2]).toDouble();
129
130 double datum1_Dani = dan_datum1 +(mjesec_datum1*30)+(godina_datum1*365);
131
132 //DRUGI datum
133 QStringList datum2 = kraj_INTERVALA.split(".");
134 double dan_datum2 = QString(datum2[0]).toDouble();
135 double mjesec_datum2 = QString(datum2[1]).toDouble();
136 double godina_datum2 = QString(datum2[2]).toDouble();
137
138 double datum2_Dani = dan_datum2 +(mjesec_datum2*30)+(godina_datum2*365);
139
140 double broj_dana_intervala = datum1_Dani - datum2_Dani;
141

```

*Slika 6.24 Izračun broja dana u periodu u kojem važi kamatna stopa*

kojeg je korisnik sustava unio te na klik korisnika daje iznos kamate. Za razliku od metode „Proporcionalna\_metoda()“, metoda „Konformna\_metoda“ (Slika 6.25) ne uzima parametar iznos novca.

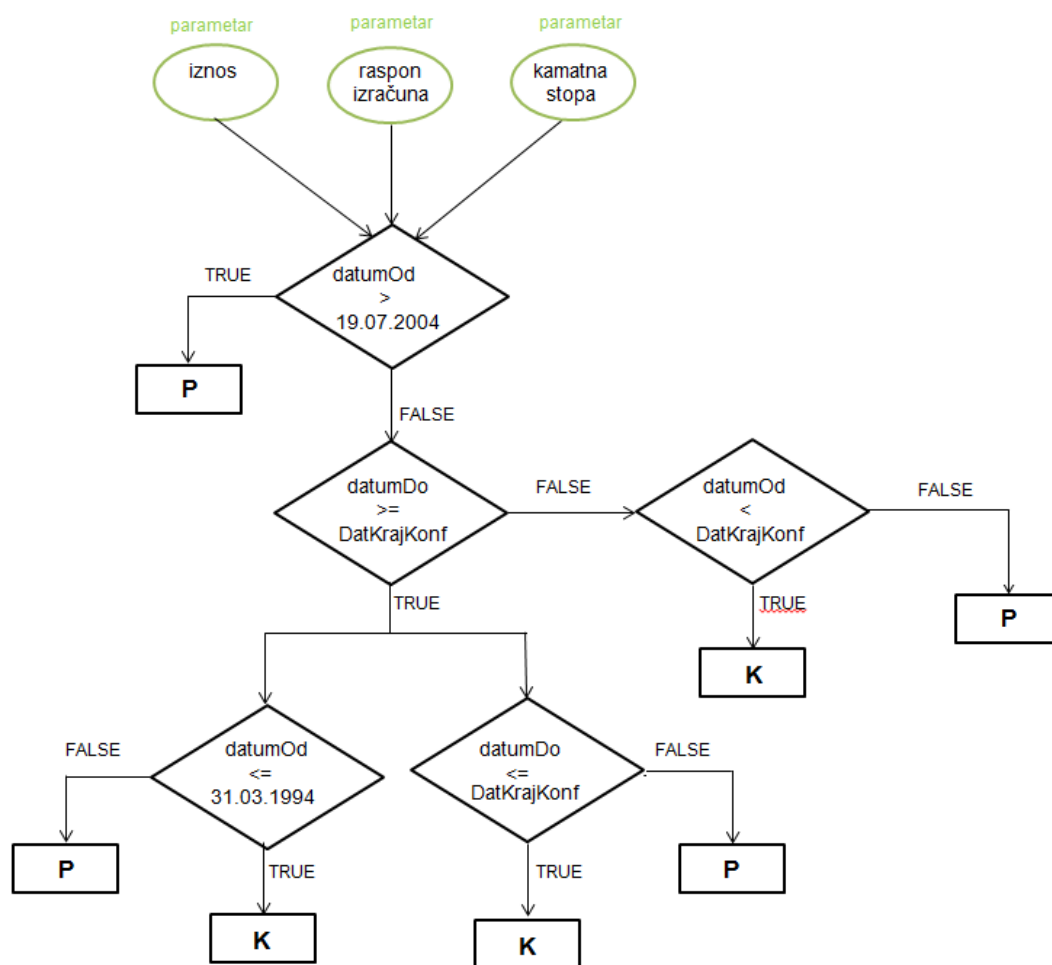
```

65 }
66 double MainWindow::Konformna_metoda(double br_dana,double iznos,double kamata,double br_dana_godina)
67 {
68
69     return rez3 = (pow((1+kamata),(br_dana/br_dana_godina))) *iznos;
70
71 }

```

*Slika 6.25 Realizacija Konformne metode*

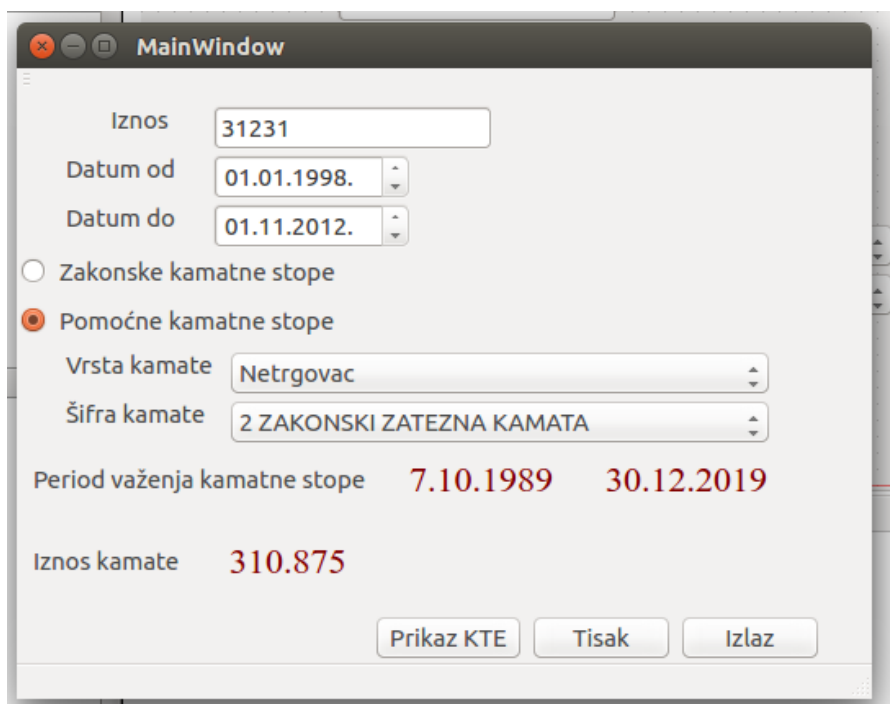
Nakon obrade GUI sučelja te pripadajući metoda potrebo je spomenuti logički tok (Slika 6.26) odnosno grananje aplikacije, na osnovu koje se određuje koja će se metoda primjenjivati i koliko i koje će parametre koristiti aplikacija pri izračunu iznosa kamate.



**K** - Konformna metoda  
**P** - Proporcionalna metoda  
**DatKrajKonf** - datum kraja obračuna konformne kamatne stope

*Slika 6.26 Logički tok aplikacije*

Naravno za kraj će biti prikazano izvođenje(*Slika 6.27*) aplikacije za prvo grananje gdje se upotrebljava Proporcionalna metoda i koja uzima i iznos koji je korisnik unio.



The screenshot shows a window titled "MainWindow" with the following elements:

- Iznos:** A text input field containing "31231".
- Datum od:** A date picker showing "01.01.1998.".
- Datum do:** A date picker showing "01.11.2012.".
- Radio buttons:** Two options are present: "Zakonske kamatne stope" (unselected) and "Pomoćne kamatne stope" (selected).
- Vrsta kamate:** A dropdown menu showing "Netrgovac".
- Šifra kamate:** A dropdown menu showing "2 ZAKONSKI ZATEZNA KAMATA".
- Period važenja kamatne stope:** Two date fields showing "7.10.1989" and "30.12.2019".
- Iznos kamate:** A text field displaying the calculated value "310.875" in red.
- Buttons:** Three buttons at the bottom: "Prikaz KTE", "Tisak", and "Izlaz".

*Slika 6.27 Demo verzija testiranje sa ulaznim parametrima*

## 7 ZAKLJUČAK

Izrada aplikacije koja je dio poslovnog informacijskog sustava zahtjeva poštivanje i pridržavanje strogih poslovnih i zakonskih pravila. Sama logika aplikacije nije posebno složena niti pretjerano zahtjevana ali razrada i implementacija u već postojeći sustav koji zahtjeva rigorozne sigurnosne mjere predstavlja pravi izazov za realizaciju. Kad programirano ovakav tip aplikacije moramo imati jasan i dobar temelj aplikacije odnosno arhitekturu, pri čemu sav kod ne možemo staviti u jednu datoteku već ga moramo rasporediti radi bolje preglednosti i mogućih promjena u bliskoj budućnosti. Aplikacija uz to mora biti razrađena na jednostavan način u funkcije i klase. Zbog ogromnih količina podataka ovaj poslovni sustav je odvojen na „front-end“ i „back-end“ dio. Ovu aplikaciju koja obavlja unosni dio podataka sam izradio u C++ koji se izravno prevodi u strojni kod i interpretira čime se postižu odlične performanse kad je u pitanju brzina za razliku od Jave. Aplikacija omogućava izračun iznosa kamate za određeni vremenski period u kojem je vrijedila specifična kamatna stopa za to razdoblje u tom vremenskom okviru te ispisuje također vremenski period važenja kamatne stope za taj isti iznos. U ovisnosti o vremenskom okviru i zakonu koji se tad primjenjivao izračun iznosa se potom izvršava sa proporcionalnom ili Konformnom metodom.



## LITERATURA

- [1] Java platforma - [https:// web.math.pmf.unizg.hr/nastava/rp2/pred1/pred1.html /](https://web.math.pmf.unizg.hr/nastava/rp2/pred1/pred1.html/) , s interneta
- [2] Java - [https:// hr.wikipedia.org/wiki/Java\\_\(programski\\_jezik\)/](https://hr.wikipedia.org/wiki/Java_(programski_jezik)/) , s interneta
- [3] Java sintaksa - [https:// people.scs.carleton.ca/~lanthier/teaching/COMP1406/Notes/COMP1406\\_Ch1\\_ProgrammingInJava.pdf /](https://people.scs.carleton.ca/~lanthier/teaching/COMP1406/Notes/COMP1406_Ch1_ProgrammingInJava.pdf/) , s interneta
- [4] C++ osnovne značajke - [https:// hr.wikipedia.org/wiki/C%2B%2B/](https://hr.wikipedia.org/wiki/C%2B%2B/) , s internet
- [5] C++ sintaksa - [https:// carpediem.hr/PublikacijeCarpeDiem/Publikacije/C++programiranje.pdf/](https://carpediem.hr/PublikacijeCarpeDiem/Publikacije/C++programiranje.pdf/) , s interneta
- [6] C++ - [https:// stackoverflow.com/questions/35679417/cannot-open-the-header-file-in-](https://stackoverflow.com/questions/35679417/cannot-open-the-header-file-in-) , s stackoverflow-a
- [7] Wikipedija drajveri - [https:// en.wikipedia.org/wiki/Device\\_driver/](https://en.wikipedia.org/wiki/Device_driver/) , s interneta
- [8] SSL Certifikati - [https:// www.posluh.hr/novosti/zasto-je-dobro-posjedovati-ssl-certifikat /](https://www.posluh.hr/novosti/zasto-je-dobro-posjedovati-ssl-certifikat/) , s interneta
- [9] IBM-ova DB2 baza podataka - <https://www.ibm.com/analytics/products> , s interneta
- [10] Qt-creator - <https://www.qt.io/qt-features-libraries-apis-tools-and-ide/> , s interneta

## **POPIS OZNAKA I KRATICA**

JVM - Java Virtual Machine

DB2 - DataBase2

JRE – Java Runtime Environment

JDK – Java Development Kit

Java ME – Java Micro Edition

Java SE – Java Standard Edition

Java EE – Enterprise Edition

HNB – Hrvatska Narodna Banka

## SAŽETAK

Veliki poslovni informacijski sustavi uvijek su bili sastavni dio strukture svake osnovne institucije bez obzira je li u pitanju bio javni ili privatni sektor. Bez obzira na napredak i razvoj tehnologije veliki poslovni informacijski sustavi su ostali jedini sustavi koji mogu jamčiti visoku sigurnost privatnih podataka, kvalitetu usluge poput transakcija i slično, performanse, implementacije raznih zakonskih normi te njihovu dosljednu primjenu i skladištenje te čuvanje ogromnih količina podataka. Može se reći da je kao takav jedini bio u mogućnosti i biti će u budućnosti jamčiti visoke sigurnosne kriterije. Kao takav koristi se u javnom sektoru poput bolnica, visoko obrazovnih ustanova, administrativnih ustanova, banka i ostalih privatnih poduzeća sa sličnim zahtjevima. Java je uvijek bila zbog svojih karakteristika bazni programskih jezik velikih sustava, dok se C/C++ uglavnom koristi kao jedan dio implementacije sustava za unosni dio podataka ili posebnih programa prevodioca („drivera“) zbog bliskosti sa strojnim jezikom. Kao takav cilj ove aplikacije koja prikazuje jedan mali dio funkcionalnosti poslovnog sustava bio je naglasiti i istaknuti značajke i prednosti zbog kojeg su unatoč brzom razvoju tehnologije, veliki poslovni informacijski sustavi i dalje ostali u upotrebi. Izračun modula za obračun kamate pri ovršnom postupku je funkcionalnost čiji je glavni zadatak da se FINA naplati unaprijed, odnosno u pri samom početku prijave procesa ovrhe. Na taj način FINA će sa naplatiti bez obzira na to je li ovrha naplativa ili ne, čime je se izbjegla mogućnost da FINA ostane bez provizije za određenu pruženu uslugu.

## **KLJUČNE RJEČI**

JVM

Java Platforma

Konformna metoda

Proporcionalna metoda

API

JDK

## **TITLE**

Creating a module for interest calculation in Business Court

## SUMMARY

Large business information systems have always been an integral part of the structure of every basic institution, whether in the public or private sector. Regardless of the advancement and development of technology, large business information systems are the only systems that can guarantee the high security of private data, the quality of services such as transactions and the like, performance, implementation of various legal norms, their consistent application and storage, and the storage of huge amounts of data. It can be said that, as such, it was the only one that could and will guarantee high safety criteria in the future. As such, it is used in the public sector such as hospitals, higher education institutions, administrative institutions, banks and other private companies with similar requirements. Java has always been, because of its characteristics, the basic programming language of large systems, while C / C ++ is mainly used as part of a system implementation for input data or special driver programs due to its proximity to the machine language. As such, the goal of this application, which portrays a small part of the functionality of a business system, was to highlight and highlight the features and benefits that, despite the rapid development of technology, large business information systems remained in use. The calculation of the interest calculation module in the enforcement procedure is a functionality whose main task is to pay FINA in advance, that is, at the very beginning of reporting the foreclosure process. In this way, FINA will charge, regardless of whether the foreclosure is payable or not, thus avoiding the possibility of FINA leaving out of commission for the particular service provided.

## KEYWORDS

JVM

Java Platform

Convenient method

Proportional method

API

JDK