Факултет инжењерских наука Универзитета у Крагујевцу

Јасмина С. Радичевић

Java програмирање кроз примере

завршни рад

Крагујевац, 2014.

Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Машинско инжењерство

Ниво студија: Основне академске студије

Модул: Информатика у инжењерству

Предмет: Програмски језици

Број индекса: 711/2013

Јасмина С. Радичевић

Java програмирање кроз примере

завршни рад

Комисија за преглед и одбрану:

1. **др Ненад Грујовић, проф.**
2. _____ Датум одбране:_____
3. _____ Оцена:_____

У оквиру овог завршног рада кандидат треба да проучи препоручену, као и ширу литратуру која се односи на програмски језик JAVA. На бази стеченог знања потребно је да на низу одабраних примера покаже практичне кораке у примени развојног окружења JAVA програмског језика.

Препоручена литература:

[1] Ken Arnold, James Gosling and David Holmes. *The java Programming Language*. Prentice Hall, fourth edition 2006.

[2] David J. Eck. *Introduction to Programming using Java*. Free version at http://math.hws.edu/javanoted, fifth edition, 2006.

[3] Jonathan Knudsen and Patrick Niemeyer. *Learning Java*, O'Reilly, third edition, 2005.

[4] Codolini P., Davolini F., Marescotti G., Maryni P. (2007), *Developiong a distance learning system using java applets.*

Ментор:

Крагујевац, Октобар 2014. др Ненад Грујовић, проф.

Резиме рада: У раду је описан програмски језик Java, његова употреба и његове перформансе. Представљени су Java аплети и конзолне апликације кроз примере. Сви примери су одрађени у програмском језику JDK 7 (Java Development Kit, version 7). Коришћено је развојно окружење NetBeans IDE 8.0, где је описано коришћење овог алата. Рад треба да омогући лако разумевање програмског језика Java.

Кључне речи: Програмски језик, Java, објектно-орјентисано програмирање, конзолне апликације, аплети, код програма, форме, методе, петље, променљиве, програмске структуре.

Summary: This paper describes the Java programming language, it's use and it's performance. Represented as Java applets and console applications through exapmples. All exapmples were carried out in programming language JDK 7 (Java Development Kit, version 7). The development enviroment used here is NetBeans IDE 8.0, which describe the use of this tool. This paper should enable easy understanding of the Java programming language.

Keywords: Programming languages, Java, object-oriented programming, console application, applet, code program, forms, methods, loop, variables, program structure.

САДРЖАЈ

У В О Д

Изумом и доступношћу нових информационих технологија (IT), јављају се нове могућности. Данас, скоро немогуће водити било који посао без информационих технологија. Информационе технологије су нашле велику примену у овом раду.

Рад омогућава упознавање са програмским језиком Java. Тачније, омогућава детаљније упознавање са конзолним апликацијама и аплетима. Кроз примере је показана примена конзолних апликација и аплета.

За потребе рада коришћено је развојно окружење NetBeans IDE 8.0 и софтвер JDK 7 (Java Development Kit, version 7).

У првом делу овог рада је описан настанак програмског језика Java. Такође, садржи информације о синтаксним правилима овог програмског језика, типовима података који се користе, операторима, наредбама језика Java и дефинисање класе и објекта.

У другом делу рада су објашњене конзолне апликације и аплети, чему служе и каква је њихова примена.

Трећи део рада је посвећен развојном окружењу NetBeans IDE 8.0. Описане су основне могућности овог алата за израду аплета и конзолних апликација.

У четвртом делу рада је кроз примере детаљно објашњена примена аплета и конзолних апликација. Примери су урађени из области: линијске програмске структуре, разгранате програмске структуре, структуре података-низови, рад са реалним величинама двоструке тачности, уношење и издавање података-рад са датотекама и рад са знаковним величинама.

1
УВОД У ПРОГРАМСКИ ЈЕЗИК JAVA

Настанак језика Java везује се за 1991. годину. Почетком 90-тих година јавила се потреба за програмима који су независни од платформе. Због тога је група програмера компаније Sun Microsystems, одлучила да направи нов програм, који би се могао користити за прављење софтвера намењеног за потребе индустријске електронике за управљање уређајима у домаћинству. Овај језик је добио име „Oaka“, али је 1995. године оно промењено у „Java“.

Програмски језик Java је виши програмски језик опште намене који у потпуности подржава парадигму објектно орјентисаног програмирања (OOP – object-oriented programming). Програмски језик Java је већину својих особина наследио од језика C и C++.

Јава се користи за израду апликација (самостални програми) и аплета (мале програме који су уметнути у Web страну). Такође, Java програмски језик је независан од платформе на којој ради, односно могу се користити у различитим развојним окружењима [1].

1.1. ОСНОВНА СИНТАКСА

Синтаксна правила код програмског језика Java су следећа: мора почети словом (великим или малим), може бити праћено било којим бројем слова и цифара, карактери ($) и (_) су прихваћена слова, идентификатор је осетљив на величину слова и не сме бити кључна реч.

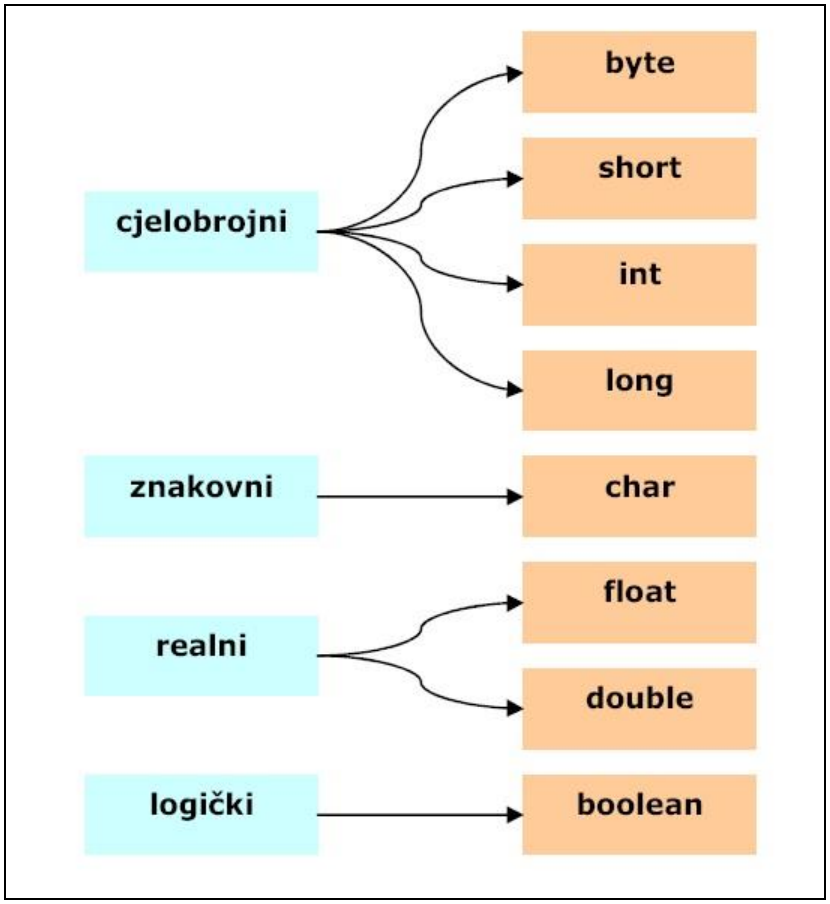
Идентификатори (engl. identifiers) служе да именује променљиве, методе, класе, типове података или неке друге објекте у програму. Идентификатори се састоје од слова и цифара, али први знак мора бити слово. У идентификаторима се поред слова и цифара може појавити и знак подвучено (_) и знак ($). Дужина идентификатора је произвољна.

Кључне речи (engl. keywords) језика Java су резервисане речи и не могу се користити као идентификатори. Кључне речи су:

abstract	assert	boolen	break	byte
case	catch	char	class	continue
const	default	do	double	else
enum	extends	false	final	finally
float	for	goto	if	implements
import	instanceof	int	interface	long
native	new	null	package	private
protected	public	return	short	static
strictfp	super	switch	synchronized	this
throw	throws	transient	true	try
void	volatile	while		

1.2. ТИПОВИ ПОДАТАКА

Типови података (data type) су посебно важни у Javi. Сви оператори морају имати дефинисан тип. Дефинисање типа омогућава откривање грешака и повећава сигурност програма. У основне типове података спадају (слика 1): integer, floating, boolean и character.



Слика 1 - Основни типови података

У Javi су дефинисана четири типа целобројног податка (integer): byte, short, int и long. Типови података у покретном зарезу (floating) су: float и double. Тип float је 32-битни и опсега је од 1.4e-045 до 3.4e+038. Тип double је 64-битни и опсега је од 4.9e-324 до 1.8e+308.

У Javi, карактери нису 8-битни као што су код већине програмских језика. Уместо тога, Java користи unicode. Unicode дефинише скуп карактера који представља све карактере који се налазе у свим природним језицима. У Javi char је unsigned 16-битни тип опсега од 0 до 65536. Стандардни 8-битни ASCII скуп карактра је подскуп unicode и опсега је од 0 до 127. Логички тип података (boolean) представља true/false вредности [2], [3].

1.3. ОПЕРАТОРИ ЈЕЗИКА JAVA

Оператори представљају радње које се извршавају над операндима дајући при томе одређени резултат.

Разликујемо следеће врсте оператора:

- аритметички оператори,
- релациони оператори,

- оператори за рад са битовима,
- оператори декрементирање и инкрементирање,
- логички оператори,
- оператори за доделу вредности.

Аритметички оператори

-	одузимање
+	сабирање
*	множење
/	дељење
%	модуо (целобројни остатак при дељењу)

Пример 1.3. а:

```
int rezultat;
rezultat = 16%3;  /*rezultat je 1,  jer je 16/2 daje ostatak 1*/
```

Релациони оператори

<	мање
<=	мање или једнако
>	веће
>=	веће или једнако
==	једнако
!=	није једнако

Пример 1.3. б:

```
int rezultat;
float a=5.2; b=11.4;
rezultat = a >= b;   /*rezultat je=0*/
rezultat = a < b;    /*rezultat je=1*/
rezultat = a != b;   /*rezultat je=1*/
```

Оператори за рад са битовима

<<	померање улево
>>	померање удесно
&	логичко „и“ за низ битова
|	логичко „или“ за низ битова
^	искључиво логичко „или“ за низ битова

Пример 1.3. в:

```
char rezultat;
char a=0x3C;    /*00111100*/
char b=0xF0;    /*11110000*/

rezultat = a << 4;    /*11000000*/
rezultat = a >> b;    /*00001111*/
rezultat = a & b;     /*00110000*/
rezultat = a | b;     /*11111100*/
rezultat = a ^ b;    /*11000011*/
```

Оператори декрементирање и инкрементирање

++	унарно инкрементирање (увећање вредности за 1)
--	унарно декрементирање (умањење вредности за 1)

Пример 1.3. г:

```
int rezultat, a=3; b=3;
rezultat = a * --b;    /*rezultat je = 6 */
/*umesto prethodne linije mogu se pisati sledeće dve linije:
b = b – 1;
rezultat = a * b;
*/

b++;     /* b = 3 */

rezultat = a--* b;    /*rezultat je = 9 */
/*umesto prethodne linije mogu se pisati sledeće dve linije:
rezultat = a * b;
a = a – 1;
*/
```

Логички оператори

&&	логичко „и“
||	логичко „или“
!	логичка негација

Пример 1.3. д:

```
int rezultat;
flot a=5.2;
char b=’0’;
rezultat = (a == 5. || a > 5.1)&&(b != '\n');
/*rezultat = (0 || 1)&&(1)=(1) &&(1)= 1; */
```

Оператори за доделу вредности

=	елементарна додела вредности

-=	+=	*=	/=	%=	аритметичка операција са доделом врености
<<=	>>=	&=	|=	^=	операција над битовима са доделом вредности

Пример 1.3. ђ:

```
flot a=5.2; b=.2E-2;
a += b; /*a=a+b=5.2+0.02=5.22 */
```

Оператори +, -, * и / раде на исти начин у Javi као и у другим програмским језицима. Могу се примeнтити на било који нумерички тип података.

Оператори инкремент и декремент су ++ и --. Инкремент оператор додаје 1 свом операнду, а декремент оператор одузима 1. Према томе, x = x + 1; је исто што и x++; а x = x – 1; је исто што и --x;. Такође је исто x++; или ++ x;

Резултат релационих и логичких оператора је boolean вредност.

Оператор доделе вредности је једноструки знак једнакости =. Java омогућује посебне облике скраћеног писања оператора доделе којим се поједностављује код наредби доделе. Исказ доделе x = x + 10; може се написати x += 10;. Опрераторски пар += каже компајлеру да додели вредност x претходну вредност за x и константу 10. Исказ x = x – 100; је исто као x -= 100;. Оба исказа додељују променљивој x претходну вредност за x умањену за 100 [2].

1.4. НАРЕДБЕ ЈЕЗИКА JAVA

Наредба је једна команда коју извршава Java интерпретатор током извршавања Java програма. Подразумевани редослед извршавања нардби је онај у којем су наредбе написане у програму, мада у Javi постоје многе управљачке наредбе којима се се овај секвенционални редослед извршавања може променити.

Наредбе у Javi могу бити просте и сложене. Просте наредбе обухватају наредбу доделе и наредбу декларације променљиве. Сложене нардебе служе за компоновање простих и сложених наредби ради дефинисања компликованијих управљачких структура у програму. У сложене наредбе спадају блок наредбе, наредбе гранања (if, if-else и switch) и наредбе понављања (while, do-while и for).

Наредба доделе је основни начин да се вредност неког израза додели променљивој у програму. Општи облик наредбе доделе је: променљива = израз.

Наредбе декларације променљиве – Све променљиве се морају декларисати пре него што се могу користити у Javi. Општи облик наредбе декларације променљиве је: тип променљива; или тип променљива = израз;.

Блок наредби (или само блок) је низ од више наредби написаних између отворене и затворене витичасте заграде. Општи облик блока наредбе је:

{
naredba$_1$

naredba$_2$
.
.
.
naredba$_n$
}

Наредба if је најпростија управљачка наредба која омогућава условно извршавање низа од једне или више наредби. Свака наредба if се састоји од једног логичког израза и једне блок наредбе која се у општем облику пише на следећи начин:

if (logicki izraz)
naredba

Пример 1.4. а:

```
    //niz
    brojClanovaNiza = unosVrednosti.nextInt();
    int niz [] = new int[brojClanovaNiza];
    System.out.println("Uneti"+brojClanovaNiza+          "clanova
niza:");
    for (i = 0; i<brojClanovaNiza; i++)
    {
```

```
        System.out.println("Clan br."+i+":");
        clan = unosVrednosti.nextInt();
        niz[i] = clan;
    }
```

У случају када је вредност логичког израза if наредбе једнака нетачно, програмска логика често налаже да треба урадити нешто друго, а не само прескочити наредбу унутар if наредбе. Тада се користи наредба if-else чији је општи облик:

if (logicki izraz)
    naredba_1
else
    naredba_2

Извршавање if-else наредбе је слично извршавању if наредбе, осим што у случају када је вредност логичког израза у загради једнака нетачно, извршава се naredba_2 и прескаче naredba_1. У другом случају, када је вредност логичког израза једнака тачно, извршава се naredba_1 и прескаче naredba_2. Тиме се у оба случаја извршења if-else наредбе завршава, а извршење програма се даље наставља од наредбе која следи иза if-else наредбе.

Пример 1.4. б:

```
if (i>=0)
{
// naredba
a=b=100;
}
else {
// naredba
a=b=-100;
}
```

Наредна наредба гранања је switch. За switch се каже да обезбеђује вишеструко гранање. Иако постоји угњеждено if као облик вишеструке селекције, у неким случајевима је switch ефикаснији. Општи облик је:

switch (izraz) {
case konstanta_1 :
    blok iskaz
break;
case konstanta_2 :
    blok iskaz
break;
case konstanta_3 :
    blok iskaz
break;
.
.
.
default :
    blok iskaz
}

Switch израз мора бити типа char, byte, short или int. Често је, израз који контролише switch једноставна променљива. Две case константе не могу имати идентичне вредности.

Пример 1.4. в:

```
//Proračun različitih površina
int i;
double površina, ivica, širina, visina;
ivica=7; širina=13; visina=19;
switch(i)
{
    case1:
         površina=ivica*ivica;
         break;
    case2:
         površina=širina*visina;
         break;
    default:
          System.out.println(„odgovarajući slučaj ne postoji!“)
```

Наредба while је основна нардба за понављање извршавања неке наредбе (или блока наредби) више пута. Општи облик је:

while (izraz)
naredba

Извршавање while наредбе се изводи тако што се најпре израчунава вредност логичког израза у загради. Ако је добијена вредност израза једнака нетачно, одмах се прекида извршавање while наредбе. Уколико је израчунавање логичког израза тачно, најпре се израчунава наредба унутар while наредбе, а потом се поново израчунава логички израз у загради и понавља исти циклус.

Тело петље које чини наредба унутар while наредбе може бити, а обично и јесте, блок наредби, па while наредба има следећи облик:

while (izraz) {
niz - naredbi
}

Пример 1.4. г:

```
int i, imax, sum;
imax=10; i=1; sum=0;
while (i<=max) {
     sum=sum+i;
     i=i+1;
}
```

Код while наредбе се услов прекида петље проверава на почетку сваке интерације. У неким случајевима, међутим, природније је проверити тај услов на крају сваког извршавања тела петље. У таквим случајевима се може користити do-while наредба која је врло слична while наредби, осим што су реч while и услов прекида померени на крај, а на почетку се налази службена реч do. Општи облик do-while наредбе је:

do
naredba

```
while (izraz);
```

или, ако је наредба унутар do-while наредбе један блок наредби, тај облик је:

```
do {
    niz - naredbi
} while (izraz);
```

Пошто се логички израз се израчунава на крају интерације сваког циклуса, тело петље се код do-while наредбе извршава бар једанпут. То се разликује од while наредбе код које се тело не мора уопште извршити.

Наредна врста наредби у Javi је наредба for. Општи облик for наредбе је:

```
for (konro ln i - deo)
    naredba
```

У другом случају, ако се тело наредбе for састоји од блока наредби, њен облик је:

```
for (konrolni - deo) {
    niz - naredbi
}
```

Пример 1.4. д:

```
int imax, sum;
imax=10; sum=0;
for (int i=1; i<=imax; i++)  {
    sum=sum+i;
}
```

Могуће је изаћи из петље, прескачући неки део кода у телу петље, коришћењем два исказа: `break;` и `continue;` [3], [6].

1.5. КЛАСЕ И ОБЈЕКТИ

Класа (engl. classes) је у већини објектно-орјентисаних језика кориснички дефинисан тип података за креирање објеката. Класе прецизно описују како објекат може да се изгради и како ће се понашати.

Објекат (engl. objects) је принцип високог нивоа, концепт који нам омогућава да размишљамо о организовању софтвера. Објекат је део меморије у неком извршном програму који поштује групу кориснички дефинисаних правила.

Класа је шаблон којим се дефинише форма неког објекта. Она специфицира и податке и код који ће радити са тим подацима.

Објекти су инстанце класе. Методе и променљиве које чине класу зову се чланице класе. Подаци чланови се зову инстанце варијабли.

Када се дефинише класа, декларише се њен облик и природа. То се постиже специфицирањем инстанци променљивих које садржи и метода који се примењује над њима. Класа се креира коришћењем кључне речи class.

Основне особине класа и објеката су: наслеђивање, апстракција и полиморфизам.

Приступ подацима се може мењати и експлицитно дефинисати преко кључних речи: public, protected и private. Public подацима се може приступити из било које тачке програма унутар или изван класе у којој су дефинисани. Protected подацима може да се приступи само из класе којој припадају или из класе изведених из ове класе (подкласа). Private подацима не може се приступити из изведених класа.

Класа обухвата поља и методе који се једним именом називају чланови класе. Да би се приступило члану класе корити се оператор тачка (.). Оператор тачка повезује име објекта са именом члана. Општи облик је:

```
objekat.član
```

Ови чланови класе могу бити статички (static) или нестатички (објектни).

Да би се боље разумеле основе класе и објеката, биће објашњено на једноставном примеру.

```
class Korisnik {
static String ime;
static int id;
```

Класа Korisnik садржи статичка поља Korisnik.ime и Korisnik.id, па у програму који користи ову класу постоји само по један примерак променљивих Korisnik.ime и Korisnik.id. Тај програм може дакле моделирати ситуацију у којој постоји само један корисник у сваком тренутку, јер има меморијски простор за чување података о само једном кориснику. Класа Korisnik и њена два статичка поља Korisnik.ime и Korisnik.id постоје све време извршавања програма.

Следећа класа садржи скоро иста (нестатичка) поља:

```
class Klijent {
String ime;
int id;
```

У класи Klijent су дефинисана нестатичка (објектна) поља ime и id, па у овом случају не постоје променљиве Klijent.ime и Klijent.id. Ова класа не садржи ништа конкретно, осим шаблона за креирање објеката те класа. Ови примери не значе да класа може имати само статичке или само нестатичке чланове. У неким примерима постоји потреба за обе врсте чланова класе.

Општи облик дефинисања класе у Javi следећи облик:

```
modifikatori class ime-klase {
telo-klase;
}
```

Модификатори на почетку дефиниције класе нису обавезни, али се могу састављати од једне или више службених речи којима се одређују извесне карактеристике класе која се дефинише. Свака класа у Javi мора припадати неком пакету. Када се дефинише нова класа, уколико се другачије не наведе, класа аутоматски припада једном безименом пакету. Тај такозвани анонимни пакет садржи све класе које нису експлицитно додате неком именованом пакету. За додавање класе у именовани пакет користи се декларација `package` којом се одређује пакет коме припада класа. Ова декларација се наводи на почетку текста класе (чак пре декларације import). Модификатор приступа public указује на то да се класа може користити изван свог пакета. Иначе, без тог модификатора, класа се може користити само унутар свог пакета. Тело класе садржи дефиниције статичких или нестатичких чланова (поља и метода) класа. На пример, у програму за претстављање студената и њихових резултата на тестовима за један предмет може дефинисати класа на следећи начин:

```
public class Student {
public String ime;  // ime studenta
public int id;  // matični broj studenta
public double test1, test2, test3; // ocene na tri testa
public double prosek(){ // izračunati prosek ocena
return (test1 + test2 + test3)/3;
}
}
```

Ниједан од чланова класе Student нема модификатор static, што значи да та класа нема статичких чланова и да је зато има смисла користи само за креирање објеката. Сваки објекат који је инстанца класе Student садржи поља ime, test1, test2, test3, као и метод prosek(). Метод prosek() примењен за различите објекте класе Student даваће зато различите резултате, јер ће се за сваки објекат користити вредности његових поља.

Да би се илустровале класе направљен је пример класе која енкапсулира информације о возилима, као што су аутомобили, комби возила и камиони. Класа се зове Vozilo и садржи информације о возилу: број путника, брзину и време.

Класа Vozilo дефинише три инстанце променљивих: putnici, brzina i vreme.

```
class Vozilo {
    int putnici;   //broj putnika
    int brzina;    // brzina vozilo
    int vreme;     // vreme
}
```

Дефиниција класе креира нови тип података. У овом случају, нови тип података зове се Vozilo. Декларација класе је само опис типа и не креира објекат. Да би се креирао објекат Vozilo, користи се исказ облика:

```
Vozilo kombi = new Vozilo(); //kreira se objekat Vozilo sa nazivom kombi
```

Када се изврши овај исказ, kombi постаје инстанца класе Vozilo. Сваки пут кад се креира и објекат који садржи копију сваке инстанце променљиве дефинисане класом. Сваки објекат класе Vozilo садржаће копије инстанце пронменљивих: putnici, brzina i vreme. Да би се приступило овим променљивама, користи се оператор тачка (.). Оператор тачка повезује име објекта са именом члана.

Дакле објекат је специфициран лево, а члан десно.

```
kombi.brzina = 16;
```

У општем случају може се користити оператор тачка да се приступи и инстанцама променљивих и методима.

```
//Ova klasa deklariše objekat tipa Vozilo.
class VoziloDemo {
public static void main(String args[]) {
Vozilo kombi = new Vozilo();
int put;
//dodela vrednosti poljima u kombi
kombi. putnici = 7;
kombi. brzina = 16;
kombi. vreme = 21;
//izračunava predjeni put
put = kombi. brzina * kombi. vreme;
System.out.println("Kombi može prevesti"  + kombi. Putnici
+"putnika na putu od"  + put +"km. ");
}
}
```

Излазни резултат: Kombi može prevesti 7 putnika na putu od 336 km.

2
КОНЗОЛНЕ АПЛИКАЦИЈЕ И АПЛЕТИ

Java се може користити за креирање два типа програма:

- Аплета (engl. applet) и
- Конзолних апликација.

У наставку текста биће објашњени оба типа програма.

2.1. АПЛЕТИ

Аплет је посебна врста Java програма намењена дистрибуцији преко Интернета и аутоматском извршавању у читачима (engl. Web browsers) који подржавају Javu. Аплет је мала апликација у Java програмском језику која се извршава у Java виртуелној машини инсталираној на Web читачу.

Аплети се уграђују у Web страницу на сличан начин на који се уграђују слике, скрипт сегменти и слично. Java аплет се учитава у читач као саставни део HTML кода и интерпретира од стране Java интерпретатора који је уграђен у читач. Java аплети могу да имају исту функционалност као и било која класична апликација. Помоћу њих могу се Web апликацији додати анимирани графички елементи, реализују компијутерске игрице или графиконима прикажу резултати неке статистичке анализе.

Шта више, аплет се с мреже преузима на захтев, баш као слика, звучна или видео секвенца. Основна разлика је у томе што је аплет интелигентан програм, а не само анимација или мултимедијална датотека. Другим речима, аплет је програм који може да реагује на акцију корисника и да се динамички мења, а не само да понавља исту анимацију или звучну секвенцу.

Ма колико аплети били узбудљиви, они би остали само пуста жеља када Java не би могла да се ухвати у коштац са два основна проблема који уз њих иду: безбедношћу и преносивошћу.

Када се користи читач Web-а који подржава Javu, може се безбедно преузети Java аплети без бојазности од вируса или злих намера. Java обезбеђује ову заштиту тако што програм ограничава на окружењу за извршавање Java програма, не дозвољавајући му приступ другим деловима рачунара. Могућност преузимања аплета без страха од штете или бојазност да ће бити прекршена безбедносна правила, за многе представља највећу новину коју је Java донела.

Web компоненте омогућавају јасније и модуларније пројектоање апликације, јер обезбеђују начин да се раздвоји апликационо моделирање од пројектовања Web страна. Стога, особе укључене у пројектовање Web страна не

морају разумети синтаксу Java програмског језика да би радиле њихов посао [4], [5].

2.2. КОНЗОЛНЕ АПЛИКАЦИЈЕ

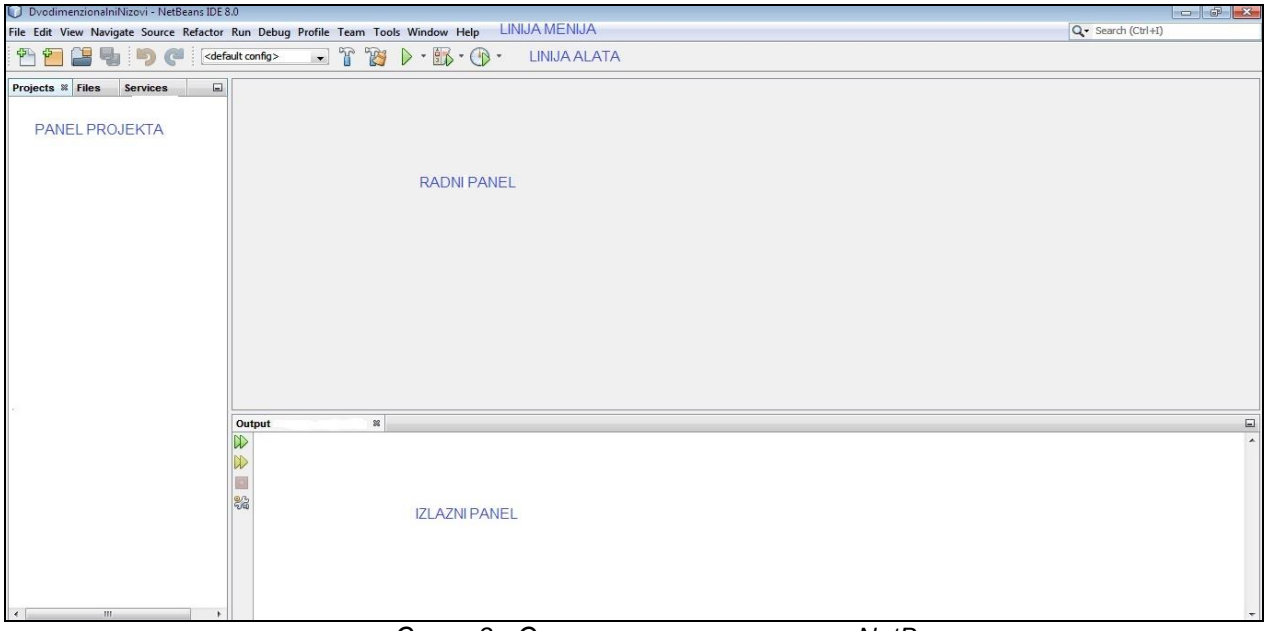
Апликација је самостални програм за решавање једног (или скупа) проблема, који садржи метод main и једну или више класа. Оне не морају да буду интегрисане у HTML коду. Апликација је везана за рад са прозорима. Излазни подаци се приказују у прозорима, а улазни подаци генеришу догађаје у прозорима. Апликације које имају само текстуални улаз и излаз називају се конзолним апликацијама, управљају целим екраном, односно прозором који симулира цео екран конзоле. Компоненте које се појављују на екрану називају се контроле. Те контроле су у облику: екрански тастери (button), радио-дугмад (radio-button), поља за потврде (checkbox), клизачи (scrollbar), поља за текст (text box), листе (list), падајуће листе (combo-box, choice) и многе друге контроле [4], [5].

3
УПОЗНАВАЊЕ СА РАЗВОЈНИМ ОКРУЖЕЊЕМ NetBeans IDE

NetBeans IDE (Integrated Developement Environment) представља интегрисано развојно окружење које је намењено за развој различитих врста програма. У овом поглављу су описане основне могућности алата за писање Java аплета (engl. applet) и конзолних апликација.

У раду се користи развојно окружење NetBeans IDE 8.0 и софтвер JDK 7 (Java Development Kit, version 7).

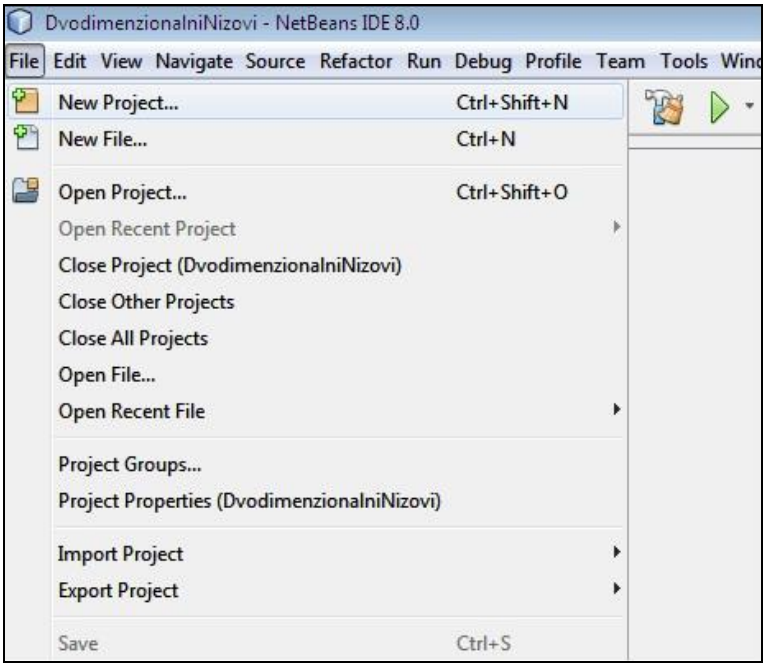
Главни прозор окружења NetBeans-а (слика 2) изгледа слично другим графичким окружењима за развој програма у другим програмским језицима. Састоји се од линије менија, линије алата на врху прозора, док преостали део прозора деле три панела у којима се приказују информације за актуелну апликацију која се развија.



Слика 2 - Основни прозор окружења NetBeans

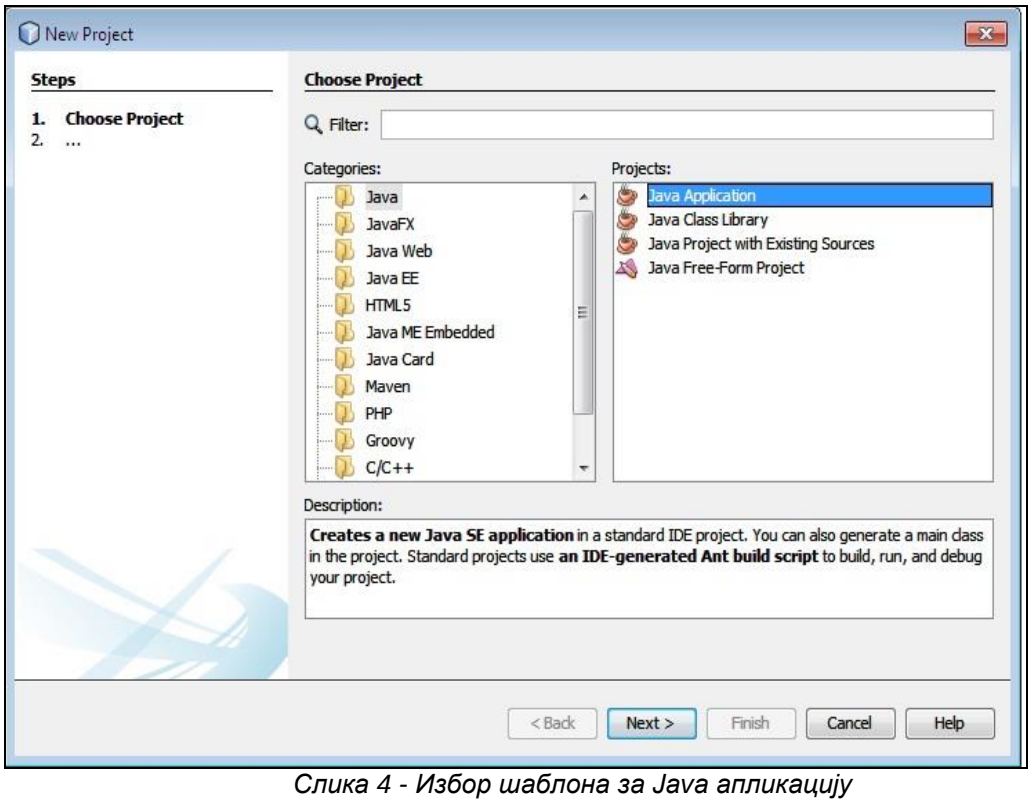
3.1. ПРАВЉЕЊЕ ПРОЈЕКТА

Основна јединица рада у окружењу NetBeans је пројекат. Да би се направио нов пројекат након покретања NetBeans-а, потребно је извршити следеће кораке: на линији менија изабере се команда File > New Project, као што је приказано на слици 3.



Слика 3 - Креирање новог пројекта

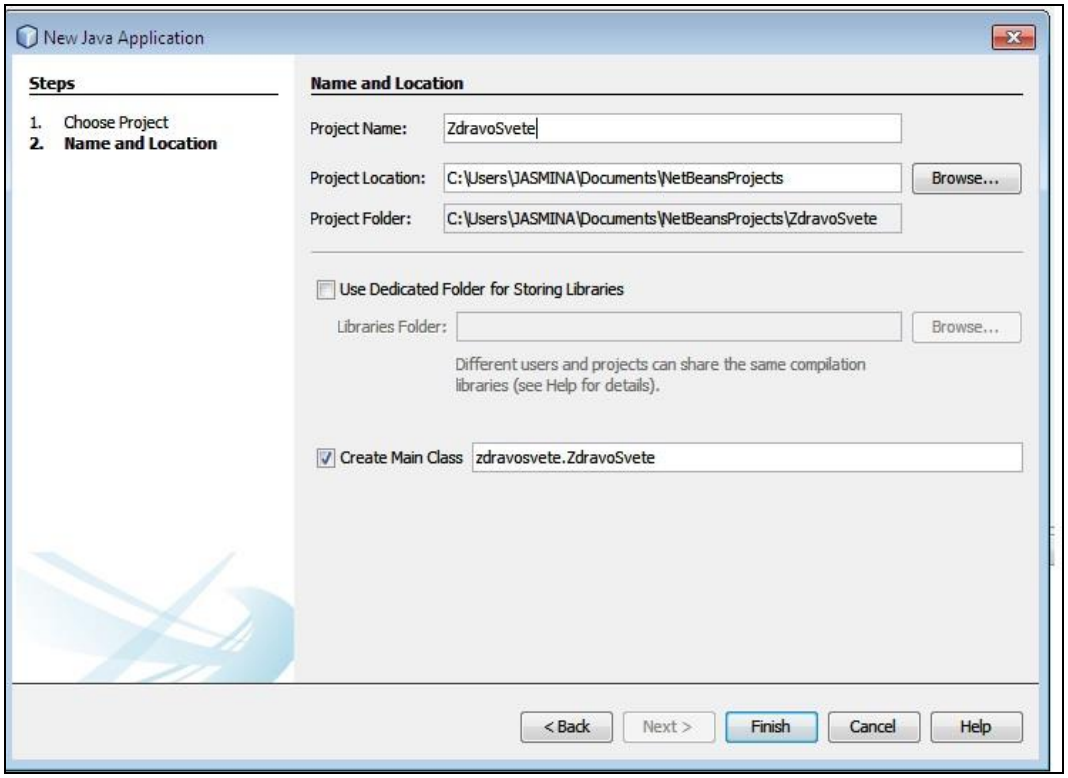
У отвореном прозору New Project одаберe се категорија Java, а затим се изабере опција Java Application, као што је приказано на наредној слици.



Слика 4 - Избор шаблона за Java апликацију

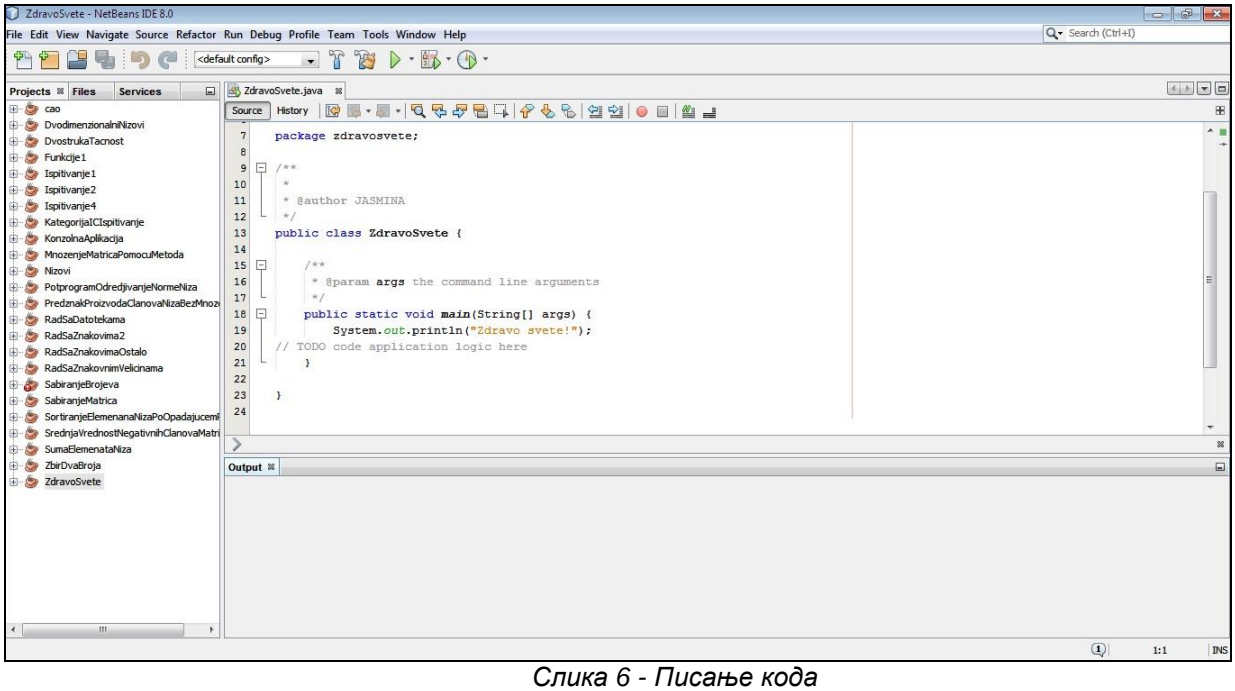
У отвореном прозору New Java Aplication потребно је уписати у поље Project Name име пројекта, у поље Project Location, дефинисати путању где ће се се снимити текући пројектат, додаје се име директоријума радног простора, у

поље Create Main Class потребно је уписати име главне класе апликације, као што је приказао на слици 5. Након тога, притисне се дугме Finish.



Слика 5 – Креирање пројекта

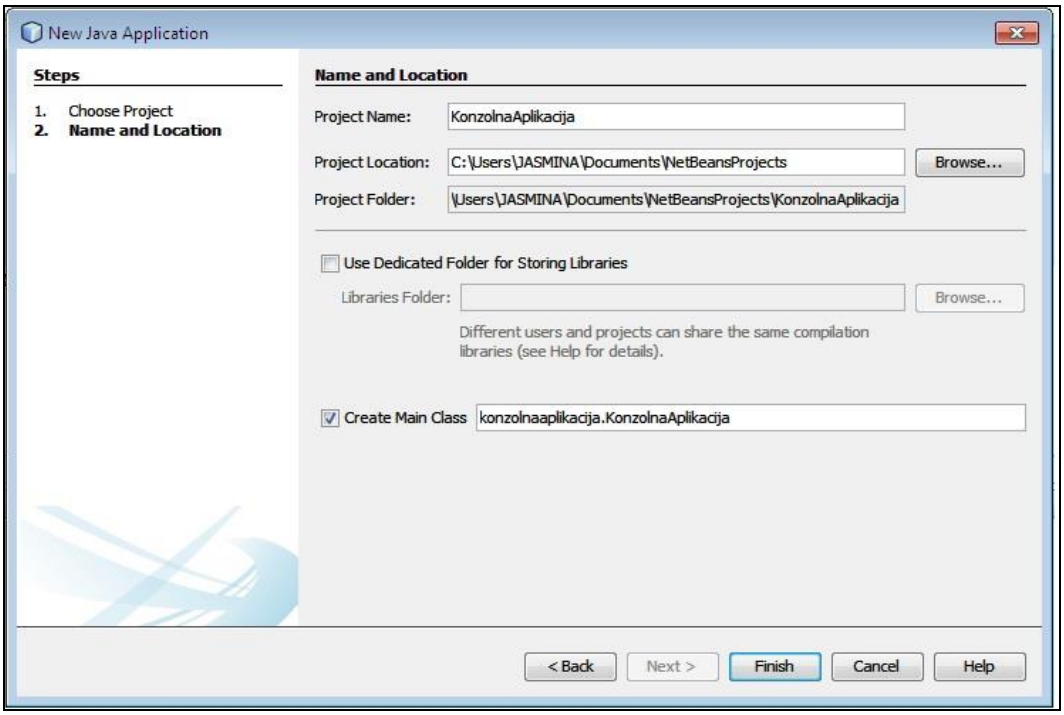
Након извршења ових корака креира се аплет, након чега је могуће писање кода као што приказано на наредној слици.



Слика 6 - Писање кода

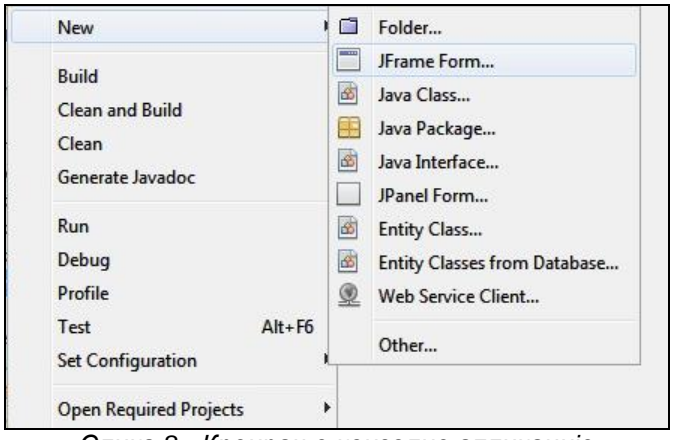
Уколико желимо да креирамо конзолну апликацију, поступак је сличан. Креирање конзолне апликације приказано је на примеру где ће се кликом на дугме иписивати: „Успешно сте кликнули!“.

Потребно је направити нови пројекат, File > New Project, након отварања тог прозора, потребо је изабрати поново категорију Java и затим опцију Java Aplication, а затим је потребно унети назив пројекта „KonzolnaAplikacija“, као што је приказано на слици 7.



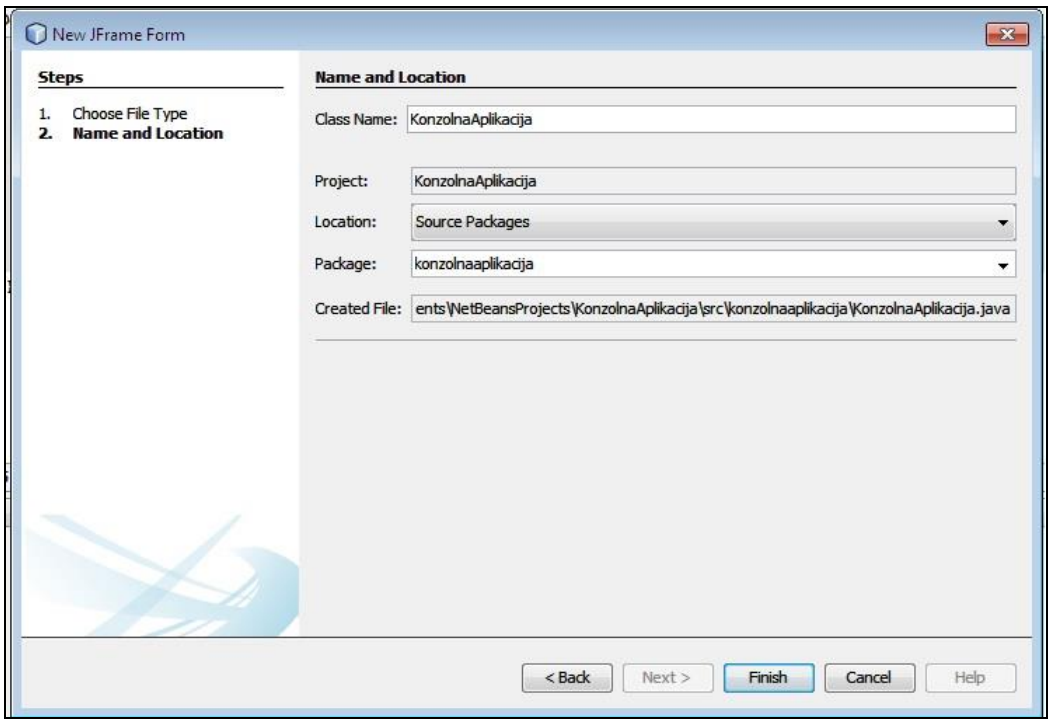
Слика 7 – Креирање пројекта

Након тога, у панелу пројекта у картици Project, појавиће се стабло и у њему име пројекта, кликом десним тастером миша на име пројекта појавиће се опција New где се одабере Jframe Form, као што је приказано на слици 8.



Слика 8 - Креирање конзолне апликације

Након овог корака, потребно је дати име главне класе.

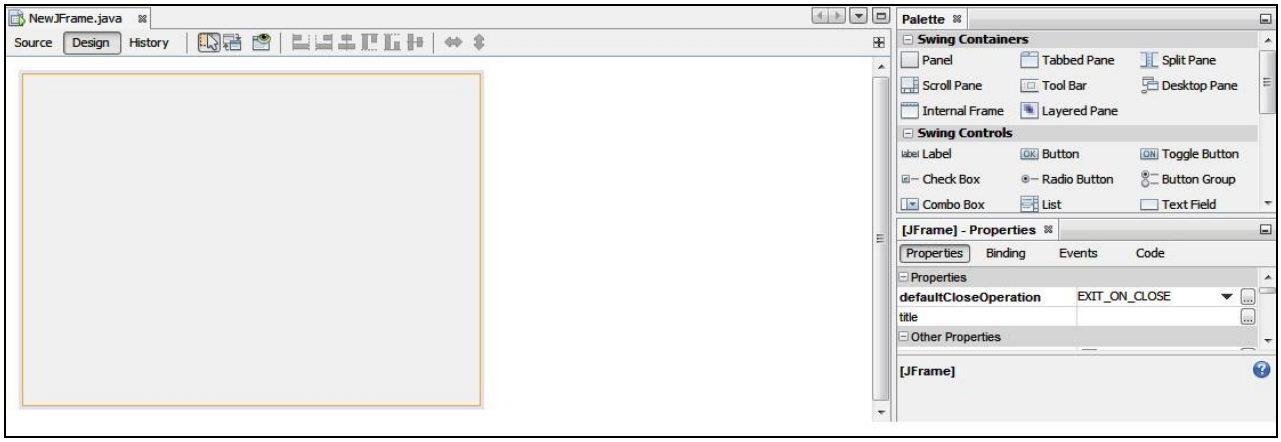


Слика 9 – Име класе

Креирана је главна класа под називом „KonzolnaApliacija“, отвара се прозор за дизајн форме и одговарајућа палета.

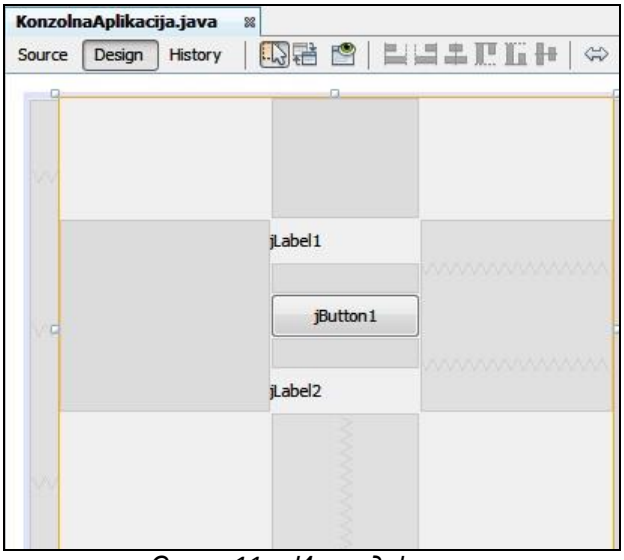
Захваљујући „Swing Containeres“ палети унутар NetBeans-a, могуће је једноставно превлачење компоненти на дијалог које су потребне за функционисање апликација, попут дугмића, панела, лабела, check box-a и слично.

Са леве стране са налази поље за форму на које се постављају контроле, изнад њега су дугмићи за форматирање компонената која се постављају на форму, кликом на дугмиће Source и Design могуће је прелазити између приказа кода и самог дизајна стране. На десној страни се налази палета алата, као што је приказано на слици 10.



Слика 10 - Прозор за креирање конзолне апликације

На главну форму потребно је додати један панел (JPanel), затим на форму додати две лабеле (JLabels) и једно дугме (JButtons).



Слика 11 – Изглед форме

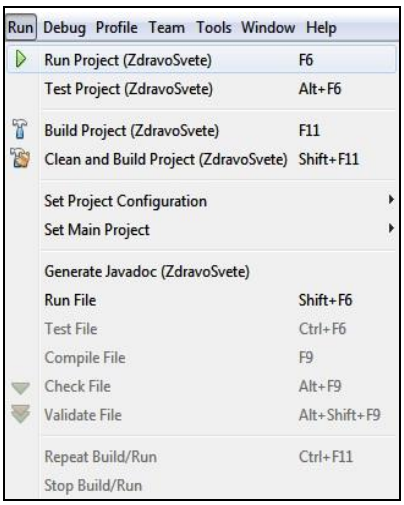
Преименовати компоненте тако да изгледају као на слици 12. Промена имена се врши на једноставан начин, селектовати дугме а потом кликом на десни тастер миша појављује се падајући мени на коме треба изабрати Edit Text.

Слика 12 – Изглед главне форме

У примеру је потребно додати функције дугмету. Десним кликом миша на дугме „Dugme“ појави се контексни мени и потребно је изабрати Events> Action> ActionPerformed. У методи коју NetBeans креира замени се ред: `// TODO add your handling code here` са `jLabel2.setText("USPEŠNO STE KLIKNULI!");`. На првој лабели је потребно заменити текст са „Klik na dugme“, а другој лабели је потребно обрисати текст. На крају је потребно снимити рад File> Save.

3.2. ПРЕВОЂЕЊЕ АПЛИКАЦЈЕ

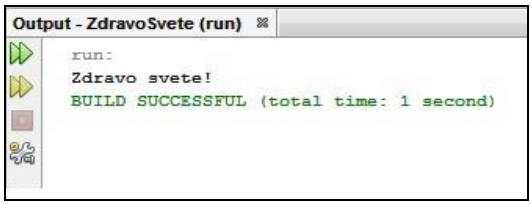
Након исписивања кода, потрено је превођење апликације. Да би се Java апликација превела, потребно је изабрати команду Run > Run Project из линије менија.



Слика 13 - Прозор за превођење програма

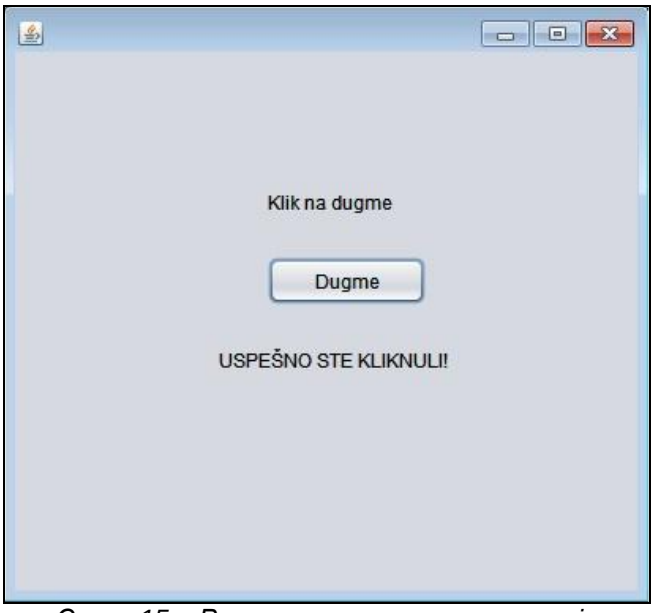
Ако се резултат превођења завршава поруком BUILD SUCCESSFUL, то значи да је превођење било успешно. У супротном, ако се резултат завршава поруком BUILD FAILED, код садржи грешке, које се морају исправити [7].

Резултат рада аплета је приказан на слици 14.



Слика 14 - Резултат рада аплета

Конзолна апликација након извршења је приказана на слици 15.



Слика 15 – Резултат конзолне апликације

4

ПРИМЕНА КОНЗОЛНИХ АПЛИКАЦИЈА И АПЛЕТА КРОЗ ПРИМЕРЕ

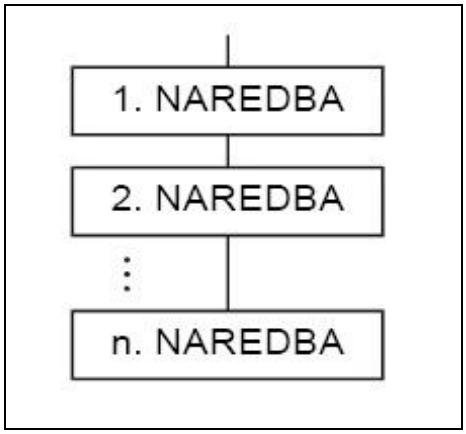
У овом делу је објашњена примена конзолних апликација и аплета кроз примере.

4.1. ПРОГРАМСКЕ СТРУКТУРЕ

У програмирању често постоји потреба да се редослед изршавања наредби услови претходно добивеним међурезултатима у току изршавања програма. На овај начин се могу образовати различити редоследи наредби при изршавању програма. Редослед наредби, у којем се програм може извршити, чини програмску структуру. По овој особини наредбе могу образовати линијску, разгранату и цикличну структуру.

4.1.1. ЛИНИЈСКЕ ПРОГРАМСКЕ СТРУКТУРЕ

Код линијске програмске структуре у Javi, као и у већини програмских језика, наредбе се извршавају редно, линија по линија, као што и сама реч каже. На следећем примеру ће бити демонстрирана једноставна линијска структура.



Слика 16 - Линијска структура

Задатак 1: Написати програм који за задате вредности x и y, израчунава вредност функције: $F(x,y) = e^{-2x}\sin(3y - x)$

У даљем тексту се налази опис кода за дати програм, овај задатак је рађен као аплет.

Овај код је написан у склопу апликације која се зове „primer1“. Креиран је такође пакет са истим називом.

```
package primer1;
```

Резервисана реч „Import“ служи за учитавање већ постојећих библиотека. У овом примеру се користи „Scanner“ класа. Ова постојећа класа у себи садржи више метода, али у примерима за конзолне апликације ће се користити само метода „nextInt“. Класа „Scanner“ служи за читање са тастатуре. Такође се користи и класа „math“, која је математичка класа, која у себи садржи математичке операције, функције и слично.

```
import java.util.Scanner;
import java.math.*;
```

Код испод представља коментар, и њега компајлер не види.

```
/**
 @author Jasmina
 */
```

Ова класа је „public“, што значи да јој је могуће приступити из било које друге класе.

```
public class Primer1 {
```

Унутар „main“ методе се налази код програма.

```
public static void main(String[] args) {
```

Креира се објекат „s“ класе „Scanner“, са параметрима „System.in“.

```
Scanner s = new Scanner(System.in);
```

Дефинишу се „double“ променљиве: „x,y, i rezultat“.

```
double x,y,rezultat;
```

Потом се исписује порука на екрану корисника од кога се захтева да се унесу вредности x и y, и то помоћу системске методе за испис текста на екрану „.out.println“, штампање линија по линија.

```
System.out.println("Unesite vrednost X a zatim vrednost
Y");
```

Сада променљива x и y добијају нове вредности, у зависности од уноса корисника. Дакле, x добија вредност тако што се позива објекат „s“ са методом „nextDouble“.

```
x = s.nextDouble();
y = s.nextDouble();
```

Пошто сада x и y имају своје вредности које су сачуване у меморији, променљива „rezultat“ је једнака F(x,y) из поставке задатка. Јасно се види да се користе математичке функције које су смештене у библиотеци „Math“, односно класи са методама „exp“ и „sin“.

```
rezultat = Math.exp(-2*x)*Math.sin(3*y-x);
```

Испис резултата:

```
System.out.println("Rezultat je:"+rezultat);
}
}
```

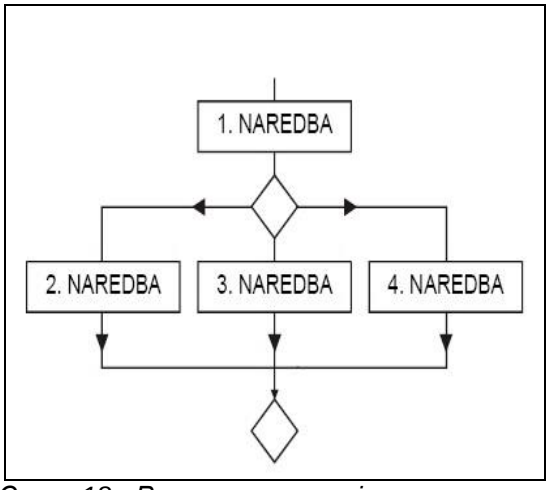
Резултат рада програма је приказан на слици 17.

```
Unesite vrednost X:
2
Unesite vrednost Y:
3
Rezultat F(x,y) =0.012033129296871049
```

Слика 17 - Резултат задатка 1

4.1.2. РАЗГРАНАТЕ ПРОГРАМСКЕ СТРУКТУРЕ

Често је потребно у програму предвидети различите токове извршавања наредби. Наредбе које омогућавају промену редоследа изршавања наредби програма зову се наредбе преласка, које се деле на: наредбе безусловног преласка и наредбе условног преласка, које омогућавају гранање у програму (разграната структура). Број грана зависи од наредбе преласка, којом се остварују гранање.



Слика 18 - Разгранате линијске структуре

4.1.2.1. НАРЕДБЕ УСЛОВНОГ ПРЕЛАСКА

У javi као и у већини програмских језика, условни прелазак представља одређени део кода који се извршава само у одређеним условима, где се користи петља „IF“.

Задатак 2: Написати програм за израчунавање функције:

$x > 0 \qquad y = x^2$

$x = 0 \qquad y = 2x$

$x < 0 \qquad y = x$

Овај пример је рађен као аплет. Пример ће показати како се користи петља if и else. Почетни део кода, до петње „if“ је већ познат из претходног примера. Дефинише се објекат класе „Scanner“, за читање вредности са тастатуре, информације кориснику програма и дефинисање променљивих које ће се користити.

```
public class Primer2 {
public static void main(String[] args) {
Scanner unosVrednosti = new Scanner(System.in);
System.out.println("Za x>0, y = x^2");
System.out.println("Za x=0, y = 2x");
System.out.println("Za x<0, y = x");
System.out.println("****************\n   Unesite   vrednost
X:");
double x = unosVrednosti.nextInt();
double y;
```

Петља „if“ почиње клучном речју „if“ и затим услов који се пише у наставку, у загради. У овом случају, ако је X веће од 0, извршава се део програма унутар витичастих заграда.

```
if (x>0)
{
y=(int) Math.pow(x, 2);
}
```

Метода „Pow“ служи за степеновање одређеног броја. У загради је потребно унети параметре, где се на првом месту уноси члан који се степенује, а на другом месту сам степен. Овде је број 2, јер је реч о квадратном броју.

Следећа петља је „else if“ која се извршава ако претходни услов није испуњен, а задовољавају се критеријуми унутар заграде. Уколико је x једнако 0, онда се извршава следећа наредба:

```
else if (x==0)
{
y=2*x ;
},
```

Уклолико ни један услов није испуњен, извршава се наредба након речи „else“.

```
else
{        y=x;    }    System.out.println("Y="+y);    }
```

Резултат програма је дат на слици 19.

Слика 19 - Резултат задатка 2

<u>Задатак 3</u>: Саставити програм који за задате вредности a и b, израчунава вредност c ако важи следећи услов:

a>b → c=a*b,

a<=b → c=a+b.

Следећи пример је сличан претходном:

```
public static void main(String[] args) {
Scanner unosVrednosti = new Scanner(System.in);
System.out.println("Unesite vrednost A a zatim vrednost
B");
double a,b,c = 0;
a = unosVrednosti.nextInt();
b = unosVrednosti.nextInt();
System.out.println("a>b => c = a*b \n a<=b => c = a+b \n");
if (a>b)
{
c=a*b;
}
else if (a<=b)
{
c=a+b;
```

```
}
System.out.println("C= "+c);
}
```

Резултат програма је дат на слици 20.

```
Unesite vrednost A a zatim vrednost B:
2
3
a>b => c = a*b
 a<=b => c = a+b

Rezultat:

C= 5.0
```

Слика 20 - Резултат задатка 3

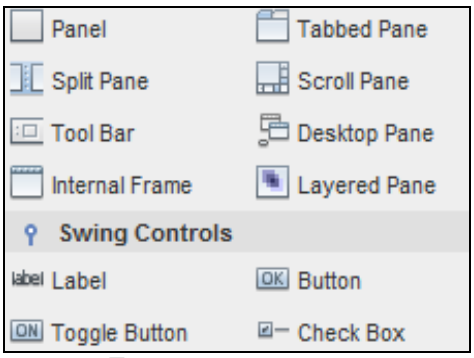
У Javi је могуће креирати аплете и конзолне апликације. На следећем примеру ће бити демонстрирано коришћене конзолних апликација, где ће бити објашњено како се креирају форме.

Код креирања дијалога односно „прозора“ потребно је на назив пакета кликнути десним тастером миша, а затим „new-jFrameForm“, након чега се креира аутоматски део кода, који се иницијализује приликом покретања апликације. Захваљујући „Swing Containeres“ палети унутар NetBeans-a, могуће је једноставно превлачење компоненти на дијалог односно „прозор“, које су потребне за функционисање апликације, попут дугмића, панела, лабела, check box-a и слично.

Задатак 4: За задате вредности a и b, написати део програма који израчунава вредност c, према следећој формули:

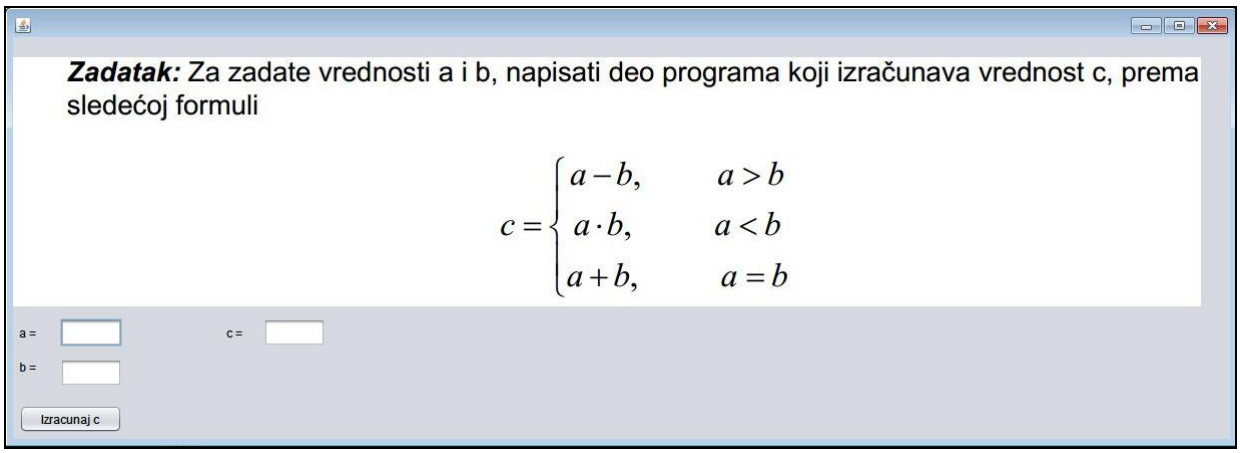
$$c = \begin{cases} a - b, & a > b \\ a \times b, & a < b \\ a + b, & a = b \end{cases}$$

На следећој слици 21. је један део палете алата, где се компоненте превлачењем додају на форме.



Слика 21 - Палета алата у NetBeans-у 8.0

Када се додају жељене компоненте као на слици испод, потребно је одредити шта ће програм заправо радити са текстуалним пољима, дугмићима и сличо. Креирано је три текстуална поља за унос текста, једна лабела којој је додељен задатак, односно слика задатка и једно дугме које заправо и садржи код задатка.



Слика 22 - Форма задатка 4

У задатку је потребно израчунати вредност c у зависности од унетих вредности a и b. Овде се за разлику од претходних примера не користи класа „Scanner“, јер је овде унос у другом формату преко форми, што ће бити објашњено у коду испод.

Креира се класа „Primer5“ која наслеђује особине „Jframe-a“.

```
public class Primer5 extends javax.swing.JFrame {
```

Додаје се конструктор, са истим називом који садржи методу за иницијализацију прозора и свих компоненти. Овај код је аутоматски генерисан, приликом креирања „Jframe“ форме.

```
public Primer5() {
initComponents();
}
```

Следећи код односно метода се аутоматски додаје, након што се одабере акција која ће се извршити након клика на дугме за израчунавање.

```
privatevoidjButton1ActionPerformed(java.awt.event.
ActionEvent evt) {
```

Следећи корак је дефинисање променљивих, a и b. Променљива a као и b је заправо број који се унесе у текстуално поље „JtextField1“. Потребно је да тај број програм препозна као „double“ (двострука прецизност), па се користи метода „parseDouble“ где је за параметар потребно унети „String“ (текстуална вредност), а то је заправо „string“ које се унесе у поље 1.

```
double a = Double.parseDouble(jTextField1.getText());
double b = Double.parseDouble(jTextField2.getText());
double c = 0;
```

Наредни код је већ познат из претходног примера.

```
if (a>b)
{
c = a-b;
}
else if (a==b)
{
c = a+b;
}
else if (a<b)
{
c = a*b;
}
```

Испис резултата се врши уносом места где желимо да испишемо резултат, затим метода „.setText“ са параметром односно променљивом која представља резултат и она се додаје тако што се унесе испред променљиве знак „+“

```
jTextField3.setText(""+c);
}
```

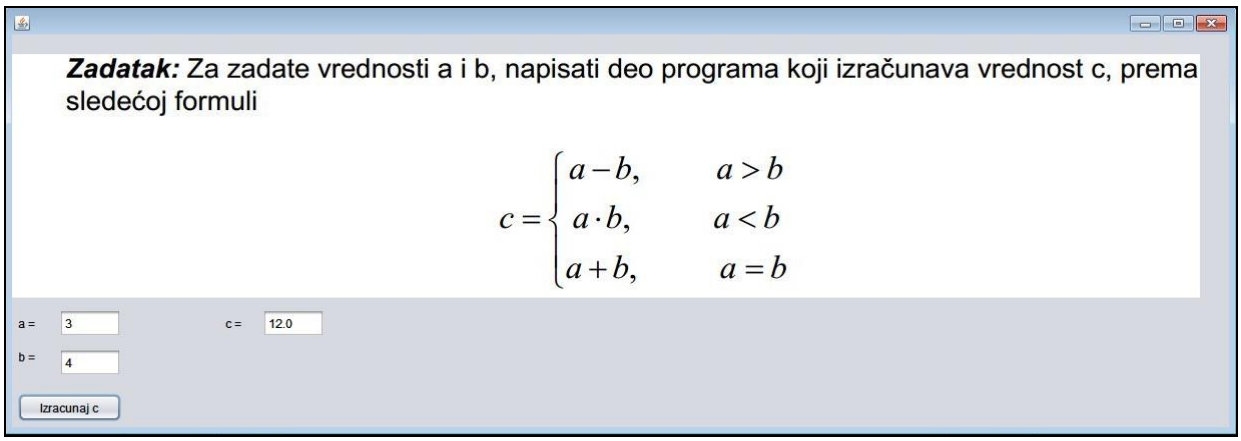
И код оваквих програма као и код сваког Java програма мора да постоји „main“ метода, али се овде ова метода аутоматски иницијализује након креирања примера, па није потребно додатно разматрање.

```
public static void main(String args[]) {
/* Set the Nimbus look and feel */
java.awt.EventQueue.invokeLater(new Runnable() {
public void run() {
new Primer5().setVisible(true);
}
});
```

Аутоматско дефинисање дугмића, лабела и текстуалних поља је објашњено у следећем коду.

```
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JTextField jTextField1;
```

Испис резултата је дат на слици испод:

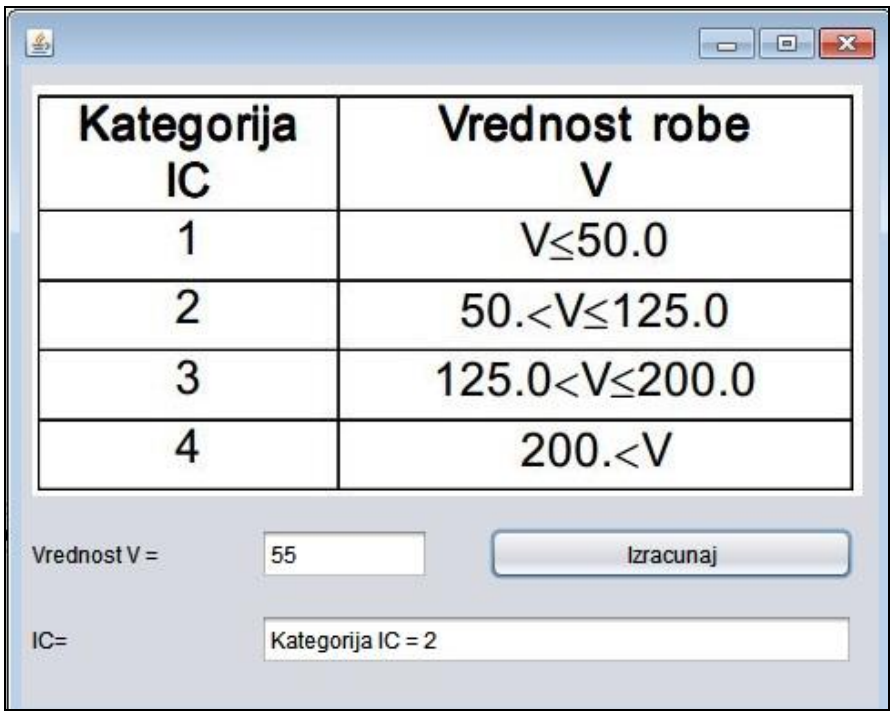


Слика 23 - Резултат задатка 2

<u>Задатак 5:</u> Користећи наредбе блоковског условног преласка дефинисати категорију вредности робе према приложеној табели:

Категорија IC	Вредност робе V
1	$V \leq 50.0$
2	$50.0 < V \leq 125.0$
3	$125.0 < V \leq 200.0$
4	$200.0 < V$

Овај пример је такође рађен као конзолна апликација.



Слика 24 - Форма задатка 5

Код је сличан коду претходног примера, па ће бити наведен део кода само у оквиру методе за клик на дугме „Izračunaj“:

```
privatevoidjButton1ActionPerformed(java.awt.event.
ActionEventevt) {
```

Дефинише се променљива V која се добија након уноса вредности у текстуално поље.

```
double V = Double.parseDouble(jTextField1.getText());
int ic = 0;
if (V<=50)
{
ic = 1;
}
```

За овај део кода ваља напоменути да “&&“ представља логички склоп који представља у нашем језику везник „i“, што значи у овом примеру да је потребно да буду испуњена два услова, да је променљива „V“ већа од 50 и мања од 125 да би се извршила наредба унутар витичастих заграда које се налазе унутар петље [8].

```
else if (V>50 && V<=125)
{
ic = 2;
}
else if(V>125 && V<=200)
{
ic = 3;
}
else if (V>200)
ic = 4;
jTextField2.setText("Kategorija IC = "+ic);
}
```

4.2. СТРУКТУРЕ ПОДАТАКА – НИЗОВИ

Низ је колекција променљивих истог типа одређен заједничким именом. У Javi, низови могу имати једну или више димензија. За разлику од других програмских језика у Javi су низови имплементирани као објекти.

Декларација низа у Javi се врши на два начина и то:

tip_elemenata[] a;

tip_elemenata a[];.

Променљива „a" је само локација у меморији која може да буде референца на низ бројева одговарајућег типа. Променљива „a" садржи адресу низа у меморији.

Пример креирања низа:

a = new int [10]

Оператором „new" креира се низ који може да чува 10 вредности типа „int", а локација почетног елемената низа у меморији смешта се у променљиву „a".

4.2.1. ЈЕДНОДИМЕНЗИОНАЛНИ НИЗОВИ

Једнодимензионални низ је листа сродних променљивих.

Задатак 6: Саставити програм за креирање и приступ члановима низа.

Кроз овај пример је показано креирање аплета, сви следећи примери везани за низове су аплети.

```
public class Nizovi {
```

Дефинисање променљивих које ће се користити у програму.

```
private static int brojClanovaNiza;
private static int clan;
private static int i;
private static int clanZaPristup;
public static void main(String[] args) {
Scanner unosVrednosti = new Scanner(System.in);
System.out.println("Uneti broj clanova niza.");
brojClanovaNiza = unosVrednosti.nextInt();
```

Креирање променљиве низ, која у себе смешта онолико чланова низа колико се зада преко тастатуре и смести у променљиву „brojClanovaNiza".

```
int niz [] = new int[brojClanovaNiza];
System.out.println("Uneti"+brojClanovaNiza+"clanovaniza:");
```

„For" петља представља део кода који ће се извршавати све док је задовољен услов у загради, и најчешће садржи променљиву односно бројач који креће од 0, извршава се док је испуњен одговарајући услов и увећава за дату вредност (i++ - након завршене петље, уколико се поново улази у петљу, променљива „i" се увећава за 1).

```
for (i = 0; i<brojClanovaNiza; i++)
{
```

При уласку у петљу, i=0, а од корисника се захтева да унесе све вредности чланова низа у зависности од броја тих чланова.

```
System.out.println("Clan br."+i+":");
```

Променљива „član" добија прву вредност, и смешта је у niz[0], након поновног уласка у петљу, i=1, па ће нова вредност члана бити смештена на позицију 1 у низу и тако редом у зависности од броја чланова тог низа.

```
clan = unosVrednosti.nextInt();
niz[i] = clan;
}
```

Испис низа се врши позивањем методе „.toString“ класе „Arrays“.

```
System.out.println("Niz je: "+Arrays.toString(niz)+"\n");
System.out.println("Kom clanu niza zelite da pristupite?
\n");
```

Променљивој „clanZaPristup“ тренутно није додељена ни једна вредност, али се од корисника захтева да унесе тај број (тај број не сме да буде већи од броја чланова низа).

```
clanZaPristup = unosVrednosti.nextInt();
if (clanZaPristup <= brojClanovaNiza)
{
```

У делу испод се врши приступ члану низа..

```
System.out.println("Clan niza je: "+niz[clanZaPristup]);
}
else System.err.println("Greska!");
return;
}
```

Резултат извршавања програма је приказан на слици 25.

```
Uneti broj clanova niza.
3
Uneti3clanova niza:
Clan br.0:
1
Clan br.1:
2
Clan br.2:
3
Niz je: [1, 2, 3]

Kom clanu niza zelite da pristupite?

2
Clan niza je: 3
```

Слика 25 - Резултат задатка 6

Задатак 7: Написати програм који исписује низ и рачуна суму елемената низа.

И у овом задатку је потребно дефинисати променљиве које ће се користити:

```
private static int brojClanovaNiza;
private static int clan;
private static int i;
private static int suma;
```

Наредни код је сличан коду као у претходном примеру.

```
public static void main(String[] args) {
Scanner unosVrednosti = new Scanner(System.in);
System.out.println("Uneti broj clanova niza.");
brojClanovaNiza = unosVrednosti.nextInt();
int niz [] = new int[brojClanovaNiza];
System.out.println("Uneti"+brojClanovaNiza+"clanovaniza:");
```

Део за одређивање суме чланова низа. У овом делу се покреће петља која ће се извршити онолико пута колико има чланова низа.

```
for (i = 0; i<brojClanovaNiza; i++)
{
```

Почетна вредност променљиве „suma“ је нула, а врши се читање првог члана низа. Па је сума једнака нула плус члан који се тренутно чита, у следећем уласком у петљу, сума ће бити први члан плус други, и тако редом.

```
System.out.println("Clan br."+i+":");
clan = unosVrednosti.nextInt();
niz[i] = clan;
suma = suma+clan;
}
System.out.println("Niz je: "+Arrays.toString(niz)+"\n");
System.out.println("Suma elemenata niza je: "+suma);    }
```

На слици 26 је приказан резултат рада програма.

```
Uneti broj clanova niza.
3
Uneti3clanova niza:
Clan br.0:
1
Clan br.1:
2
Clan br.2:
55
Niz je: [1, 2, 55]

Suma elemenata niza je: 58
```

Слика 26 - Резултат задатка 7

4.2.2. ДВОДИМЕНЗИОНАЛНИ НИЗОВИ

Дводимензионални низови је листа једнодимензионалних низова.

Задатак 8: Написати програм за сабирање матрица A и B, које су дефинисане на следећи начин:

$$A=\begin{bmatrix}1 & 2 & -1 & 0\\ 4 & 0 & 2 & 1\\ 2 & -5 & 1 & 2\end{bmatrix} \qquad B=\begin{bmatrix}3 & -4 & 1 & 2\\ 1 & 5 & 0 & 3\\ 2 & -2 & 3 & -1\end{bmatrix}$$

Одштампати елементе резултујуће матрице C= A+B.

Матрице представљају дводимензионалне низове, и у овом задатку се ради са матрицом 3x4. Дефинисање дводимензионалних низова је слично као и код једнодимензионалних. Прве две линије кода представљају статичне променљиве, односно низове.

```
int[][] a = {{1, 2, -1, 0}, {4, 0, 2, 1}, {2, -5, 1, 2}};
int[][] b = {{3, -4, 1, 2}, {1, 5, 0, 3}, {2, -2, 3, -1}};
```

Креира се дводимензионални низ 3x4, где ће бити смештен резултат.

```
int[][] c = new int[3][4];
```

Променљива „i“ представља број врста.

```
for(int i = 0; i < 3; i++)
```

Променљива „j“ је број колона.

```
for(int j = 0; j < 4; j++)
```

Прва вредност „i“ је нула, и прва вреност „j“ је нула, па се сабирају вредности матрице „a“ са врстом 0 и колоном 0, са матрицом „b“ такође са врстом и колоном са вредностима 0.

```
c[i][j] = a[i][j] + b[i][j];
System.out.println("Dobijena matrica c = a + b je ");
```

Уписивање вредности у матрицу „c“.

```
for(int i = 0; i < 3; i++) {
for(int j = 0; j < 4; j++)
System.out.print(c[i][j] + " ");
System.out.println();
}
}
```

На слици 27 је приказан резултат задатка 8.

```
Dobijena matrica c = a + b je
4 -2 0 2
5 5 2 4
4 -7 4 1
```

Слика 27 - Резултат задатка 8

Задатак 9: Написати програм који израчунава множење матрица коришћењем метода односно функција.

У овом примеру су коришћене методе, односно делови кода програма који у себи садржи неки задатак. Коришћене су методе са параметрима које ће бити објашњене.

У коду испод све је познато из претходног примера.

```
public static void main(String[] args) {
Scanner unosVrednosti = new Scanner(System.in);
System.out.print("Uneti broj vrsta matrice A: ");
int vrstaA = unosVrednosti.nextInt();
System.out.print("Uneti broj kolona matrice A / vrsta u
matrici B: ");
int kolonaA = unosVrednosti.nextInt();
System.out.print("Uneti broj kolona matrice B: ");
int kolonaB = unosVrednosti.nextInt();
```

Креирање матрице A и B и додељивање вредности. Познато је да је множење матрица могуће само ако матрица A има исти број врста колико има колона матрица B.

```
int[][] a = new int[vrstaA][kolonaA];
int[][] b = new int[kolonaA][kolonaB];
System.out.println("Uneti vrednosti matrice A");
for (int i = 0; i < a.length; i++) {
for (int j = 0; j < a[0].length; j++) {
a[i][j] = unosVrednosti.nextInt();
}
}
System.out.println("Uneti vrednosti matrice B");
for (int i = 0; i < b.length; i++) {
for (int j = 0; j < b[0].length; j++) {
b[i][j] = unosVrednosti.nextInt();
}
}
```

Позивање методе за множење и придруживање резултата множења променњивој „c“. Метода множења захтева 2 параметра, односно две матрице и прослеђују се A и B.

```
int[][] c = mnozenje(a, b);
```

Након позивања методе прелази се на методу ради извршења задатка, па се враћа на код за испис резултата.

```
System.out.println("Rezultat mnozenja matrice A i matrice
B");
for (int i = 0; i < c.length; i++) {
for (int j = 0; j < c[0].length; j++) {
System.out.print(c[i][j] + " ");
}
System.out.println();
}
}
```

Метода за множење је типа матрице и захтева 2 параметра односно две матрице.

```
public static int[][] mnozenje(int[][] a, int[][] b) {
int vrstaA = a.length;
int kolonaA = a[0].length; //
int kolonaB = b[0].length;
int[][] c = new int[vrstaA][kolonaB];
for (int i = 0; i < vrstaA; i++) {
for (int j = 0; j < kolonaB; j++) {
for (int k = 0; k < kolonaA; k++) {
c[i][j] = c[i][j] + a[i][k] * b[k][j];
}} }
```

Након извршења множења, главном програму се враћа вредност „C“, која се уписује у претходно поменуту вредност, односно резултат.

```
return c;
}}
```

На наредној слици је приказан резултат рада програма.

```
Uneti broj vrsta matrice A:
2
Uneti broj kolona matrice A / vrsta u matrici B: 1
Uneti broj kolona matrice B: 3
Uneti vrednosti matrice A
1
2
Uneti vrednosti matrice B
2
3
4
Rezultat mnozenja matrice A i matrice B
2 3 4
4 6 8
```

Слика 28 - Резултат задатка 9

Задатак 10: Написати програм који помоћу функција рачуна збир четири броја, унета са тастатуре.

Још један пример ради бољег појашњења функција односно метода у Javi. Прво је потребно дефинисати променљиве.

```
public static void main(String[] args) {
Scanner unosVrednosti = new Scanner(System.in);
System.out.println("Uneti 4 broja: \n");
double a = unosVrednosti.nextDouble();
double b = unosVrednosti.nextDouble();
double c = unosVrednosti.nextDouble();
double d = unosVrednosti.nextDouble();
```

Променљива „zbir“ ће добити вредност након што се позове метода „suma“. Метода за сумирање садржи четири параметра, па је самим тим потребно тој методи која се позива проследити тих четири параметра, односно вредности: a,b,c и d.

```
double zbir = suma(a,b,c,d);
System.out.println("Suma datih brojeva je: "+suma);
}
```

Метода за сумирање је „private“, то значи да је доступна само овој класи, статична (на нивоу класе) и типа „double“. Методе могу бити и „void“, оне не враћају никакву вредност, само извршавају одређену радњу. Имајући у виду да је ово „double“ метода, потребно је да та иста врати неку вредност, односно променљиву делу одакле се позива, односно враћа се у „main“ методу и додељује вредности „double zbir“.

```
private static double suma(double a, double b, double c,
double d) {
suma = a + b + c + d;
return suma;
}
```

Резултат рада програма је приказан на слици 29.

```
Uneti 4 broja:

11.5
12
1008
10.884
Suma datih brojeva je: 1042.384
```

Слика 29 - Резултат задатка 10

Задатак 11: Написати програм који рачуна производ чланова низа и одређује предзнак производу.

Прво су дефинисане променљиве.

```
public class PredznakProizvodaClanovaNiza {
private static int proizvod = 1;
private static String predznak;
public static void main(String[] args) {
Scanner unosVrednosti = new Scanner(System.in);
System.out.println("Uneti 5 clanova niza. \n");
int niz [] = new int[5];
```

Пролазак кроз петљу, односно свих пет чланова низа и врши се множење свих чланова низа.

```
{
int elementNiza = unosVrednosti.nextInt();
niz [i] = elementNiza;
proizvod = proizvod * niz[i];
}
```

Ако је производ већи од нуле, предзнак је плус, а у супротном је минус.

```
if (proizvod > 0)
{
predznak = "Proizvod je pozitivan. Predznak je '+'";
}
else if (proizvod == 0)
{
```

```
predznak = "Proizvod je 0.";
}
else
{
predznak = "Proizvod je negativan. Predznak je '-'";
}
System.out.println("NIZ:" +Arrays.toString(niz));
System.out.println("Proizvod datih clanova je:" +proizvod);
System.out.println(predznak);
}}
```

Резултат рада програма је приказан на слици 30.

```
Uneti 5 clanova niza.

1
2
3
4
5
NIZ:[1, 2, 3, 4, 5]
Proizvod datih clanova je:120
Proizvod je pozitivan. Predznak je '+'
```

Слика 30 - Резултат задатка 11

Задатак 12: Написати програм за одређивање норме низа:

$$|a| = \left(a_1^2 + a_2^2 + \ldots + a_n^2\right)^{1/2}$$

И у овом задатку је потребно дефинисати променљиве.

```
private static int brojClanovaNiza;
private static int clanNiza;
private static double normaNiza = 0;
public static void main(String[] args) {
```

Позивање метода за одређивање норме.

```
Norma();
}
public static void Norma ()
{
Scanner unosVrednosti = new Scanner(System.in);
System.out.println("Uneti broj clanova niza:");
brojClanovaNiza = unosVrednosti.nextInt();
int niz[] = new int[brojClanovaNiza];
System.out.println("\n Uneti clanove niza: \n");
for (int i = 0; i<brojClanovaNiza; i++)
{
clanNiza = unosVrednosti.nextInt();
niz[i] = clanNiza;
normaNiza = normaNiza+Math.pow(niz[i], 2);
}
normaNiza = Math.sqrt(normaNiza);
System.out.println("Niz je: "+Arrays.toString(niz));
```

```
System.out.println("Norma niza (|niz| =
(clan1^2+clan2^2+...+clanN^2)^1/2) ="+normaNiza);
// odredjivanje norme
}
```

Резултат рада програма је приказан на слици 31.

```
Uneti broj clanova niza:
2

 Uneti clanove niza:

5
9
Niz je: [5, 9]
Norma niza (|niz| = (clan1^2+clan2^2+...+clanN^2)^1/2) = 10.295630140987
```

Слика 31 - Резултат задатка 12

Задатак 13: Написати програм за сортирање чланова низа по опадајућем редоследу, од највећег до најмањег.

Дефинисање променљивих:

```
private static int brojClanovaNiza;
private static int clanNiza;
private static int iMin;
private static int privremeno;
private static int i ,j;
```

Креирање низа и упис броја чланова низа:

```
public static void main(String[] args) {
Scanner unosVrednosti = new Scanner(System.in);
System.out.println("Uneti broj clanova niza:");
brojClanovaNiza = unosVrednosti.nextInt();
int niz[] = new int[brojClanovaNiza];
System.out.println("Uneti clanove niza:");
for (int i = 0; i<brojClanovaNiza; i++)
{
clanNiza = unosVrednosti.nextInt();
niz[i] = clanNiza;
}
```

Позивање методе за сортирање, где се прослеђује два параметра: низ и број чланова низа.

```
sortiranje(niz, brojClanovaNiza);
}
```

Метода за сортирање је типа „void“, што значи да главном програму не враћа никакву вредност.

```
private static void sortiranje (int niz[],int brojClanova)
{
System.out.println("Uneti niz:"+Arrays.toString(niz));
```

Постоје два бројача, јер ова метода за сортирање ради на принципу упоређивања два суседна броја, и уколико је следећи суседни број већи од претходног, врши се замена места, дакле упоређују се први и други број, ако је други већи него први, сада је други на првом месту, а први на другом, па се врши упоређивање другог и трећег, па када се заврши тај процес, бројач се поново увећава за један и на тај начин се врши провера свих комбинација кроз дати низ.

```
for (i=0;i<brojClanova;i++)
```

```
{
for (j=i+1;j<brojClanova;j++)
{
```

Ако је низ од „j“ већи од низа од „i“, креира се нова променљива „privremeno“ која само тренутно бележи дати број, сада низ од „i“ је у ствари низ од „j“,а низ од „j“ преузима вредностод променљиве „privremeno“.

```
if (niz[j]>niz[i])
{
privremeno = niz[i];
niz[i]=niz[j];
niz[j]=privremeno;
}
}
}
System.out.println("Sortirani niz je:
"+Arrays.toString(niz));
}}
```

Резултат рада програма је следећи:

```
Uneti broj clanova niza:
4
Uneti clanove niza:
22
1
18
11545
Uneti niz:[22, 1, 18, 11545]
Sortirani niz je: [11545, 22, 18, 1]
```

Слика 32 - Резултат задатка 13

Задатак 14: Саставити програм који одређује средњу вредност негативних чланова матрице.

Дефинисање променљивих је приказано у коду испод.

```
private static int i, j;
private static double suma=0;
private static double srednjaVrednost =0;
private static int brojVrsta = 3;
private static int brojKolona = 4;
private static int brojNegativnihClanova =0;
private static double proba = 0;
public static void main(String[] args) {
```

Креирање матрице 4x3, која садржи и негативне чланове:

```
int [][] matrica = {{-2,5,6,-11},{22,-14,7,252},{-144,-
2,24,11}};
System.out.println("Uneta matrica je:
"+Arrays.deepToString(matrica));
```

Позива се метода за одређивање средње вредности негативнх чланова, и тој методи се прослеђује сама матрица, број врста и број колона.

```
srednjaVrednost= sredVrd(matrica,brojVrsta,brojKolona);
System.out.println("Srednja vrednost negativnih clanova
matrice je: "+srednjaVrednost);
}
```

Најпре се врши провера матрице [0][0], ако је тај број на тој позицији мањи од нуле, сума добија ту вредност, на почетку је сума једнака нули. Сваки наредни

број који је негативан се сабира са претходном сумом негативних бројева, а самим тим и број негативних чланова се увећава.

```
public static double sredVrd(int [][]matrica,int brojVrsta,
int BrojKolona) {
for (i=0;i<brojVrsta;i++)
{
for (j=0;j<BrojKolona;j++)
{
if (matrica[i][j]<0)
{
suma = suma+matrica[i][j];
brojNegativnihClanova++;
}
}
}
```

Након изласка из петље, средња вредност је једнака „suma/broj negativnih članova“. Након тога, главном програму се враћа вредност „srednjaVrednost“.

```
srednjaVrednost = suma/brojNegativnihClanova;
return srednjaVrednost;
}}
```

На наредној слици је приказан резултат програма.

```
Uneta matrica je:[[-2, 5, 6, -11], [22, -14, 7, 252], [-144, -2, 24, 11]]
Srednja vrednost negativnih clanova matrice je: -34.6
```

Слика 33 - Резултат задатка 14

4.3. РАД СА РЕАЛНИМ ВЕЛИЧИНАМА ДВОСТРУКЕ ТАЧНОСТИ

Реалне променљиве се декларишу са речју „float“.

Задатак 15: Написати програм који рачуна тригонометријске функције синус, косинус и тангенс, где је могуће резултат приказати типа double, float, short.

У Javi, највећу прецизност има променљива која је типа „double“, затим „float“, „short“, „int“ и тако даље. Овај задатак омогућава израчунавање тригонометријских функција синус, косинус и тангенс. У овом програму је презентовано како се прецизност одражава на резултат.

Овај задатак је одређен као конзолна апликација, имајући у виду да је већ објашњено како се креирају форме код конзолних апликација, тај део у опису ће бити изостављен.

Пошто у форми постоји „combo box“, као и текстуална поља, потребно је те вредности преузети и помоћу математичких метода конвертовати у потребан формат.

```
String funkcija = jComboBox1.getSelectedItem().toString();
String preciznost =
jComboBox2.getSelectedItem().toString();
double ugao = Double.parseDouble(jTextField1.getText());
double radijan = Math.toRadians(ugao);
```

Провера одабране прецизности из „combo box-a“.

```
if (preciznost == "double")
{
```

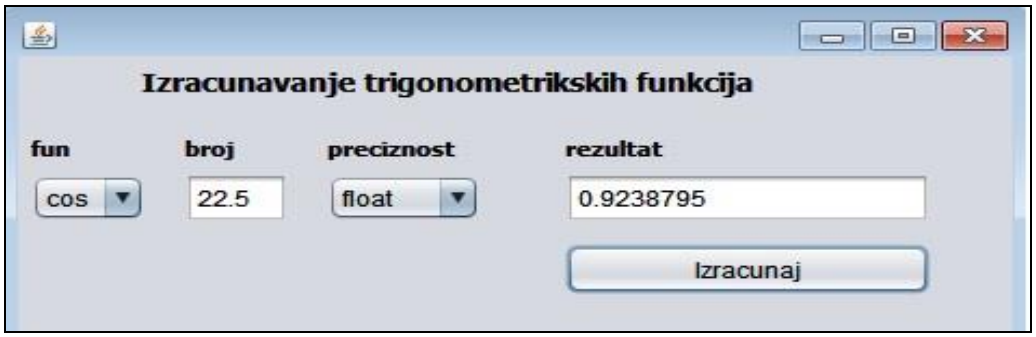
Пролазак кроз петљу је јасан, где постоје све комбинације, у зависности која функција је одабрана, односно која прецизност.

```
if (funkcija == "sin")
```

```
{
double rezultat = Math.sin(radijan);
jTextField2.setText(""+rezultat);
}
else if (funkcija == "cos")
{
double rezultat = Math.cos(radijan);
jTextField2.setText(""+rezultat);
}
else if (funkcija == "tan")
{
double rezultat = Math.tan(ugao);
jTextField2.setText(""+rezultat);
}
}
else if (preciznost == "float")
{
if (funkcija == "sin")
{
float rezultat = (float) Math.sin(radijan);
jTextField2.setText(""+rezultat);
}
else if (funkcija == "cos")
{
float rezultat = (float) Math.cos(radijan);
jTextField2.setText(""+rezultat);
}
else if (funkcija == "tan")
{
float rezultat = (float) Math.tan(ugao);
jTextField2.setText(""+rezultat);
}
}
if (preciznost == "short")
{
if (funkcija == "sin")
{
short rezultat = (short)Math.sin(radijan);
jTextField2.setText(""+rezultat);
}
else if (funkcija == "cos")
{
short rezultat = (short) Math.cos(radijan);
jTextField2.setText(""+rezultat);
}
else if (funkcija == "tan")
{
short rezultat = (short) Math.tan(ugao);
jTextField2.setText(""+rezultat);
}
}
}
```

Резултат рада програма је приказан на наредним сликама.

Слика 34 - Резултат задатка 15 „Променљива double“



Слика 35 - Резултат задатка 15 „Променљива float“

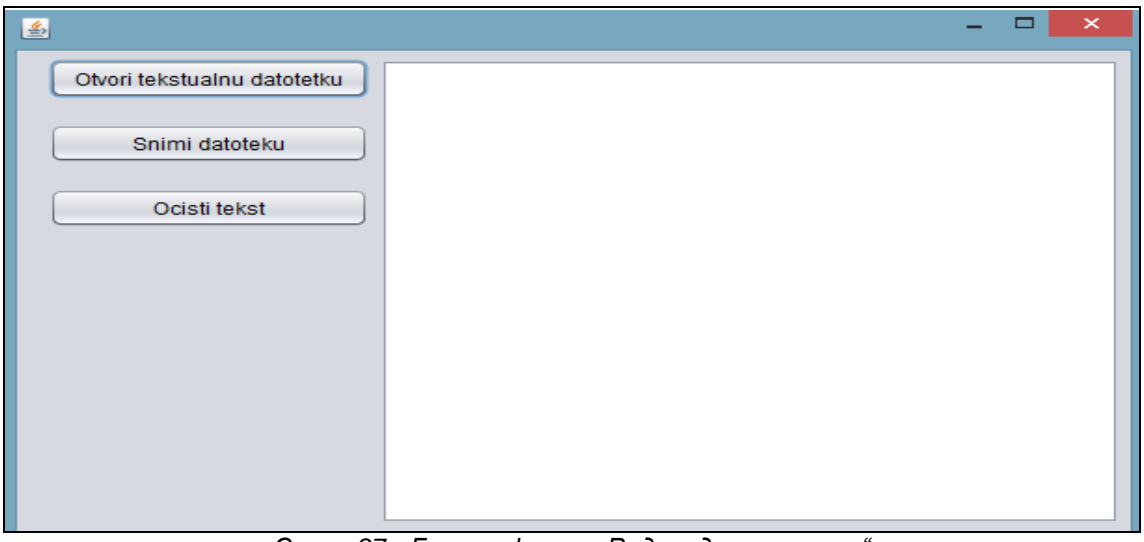
Слика 36 - Резултат задатка 15 „Променљива short“

4.4. УНОШЕЊЕ И ИЗДАВАЊЕ ПОДАТАКА – РАД СА ДАТОТЕКАМА

Задатак 16: Саставити програм који омогућава упис текста у датотеку и читање текста из датотеке.

Постоје два начина за упис и читање података у текстуални фајл, али ће бити приказан само начин преко новијих метода, које укључују визуелизацију, једноставност, прегледност чувања односно читања фајла, и то преко дијалога.

Рад са дотекама је приказан на следећем примеру. Овај задатак садржи три опције: за учитавање, снимање и брисање текста. На овом примеру је демонстиран рад конзолне апликације. На слици 37. приказана је главна форма.



Слика 37 - Главна форма „Рад са датотекама“

- Снимање текстуалног фајла:

Снимање односно чување текстуалног фајла на диск се врши најпре читањем унетог теста у текстуално поље са десне стране прозора, и његовим чувањем на хард диск преко дијалога за снимање. Најпре се преузима текст из текстуалног поља и чување у променљиву „tekst“.

```
String tekst = jTextArea1.getText();
```

Креира се објекат типа „JfileChosser“ (класа из java библиотеке).

```
JFileChooser chooser = new JFileChooser();
```

Подешавање почетног директоријума за чување фајла.

```
chooser.setCurrentDirectory(newFile("/home/me/Documents"));
```

Креира се променљива „interval“, која из Jave повлачи две вредности 0 и 1. Отвара се дијалог за чување, и уколико је потврда од корисника за чување, креира се објекат „fw“, класе „FileWriter“ (класа за писање фајлова), преузима текст од променљиве „tekst“, пише датотеку у формату „.txt“ и затвара је. Ова операција је могућа ако се користи „try“ и „cath“, односно такозвани изузеци.

```
int retrival = chooser.showSaveDialog(null);
if (retrival == JFileChooser.APPROVE_OPTION) {
try {
FileWriterfw=newFileWriter(chooser.getSelectedFile()+".txt;
fw.write(tekst.toString()); //pisane datoteka
fw.close(); // zatvaranje datoteka
JOptionPane.showMessageDialog(rootPane, "Uspesno ste
sacuvali datoteku");
} catch (Exception ex) {
ex.printStackTrace();
}}
}
```

- Читање датотеке:

Читање се врши на сличан начин као и писање:

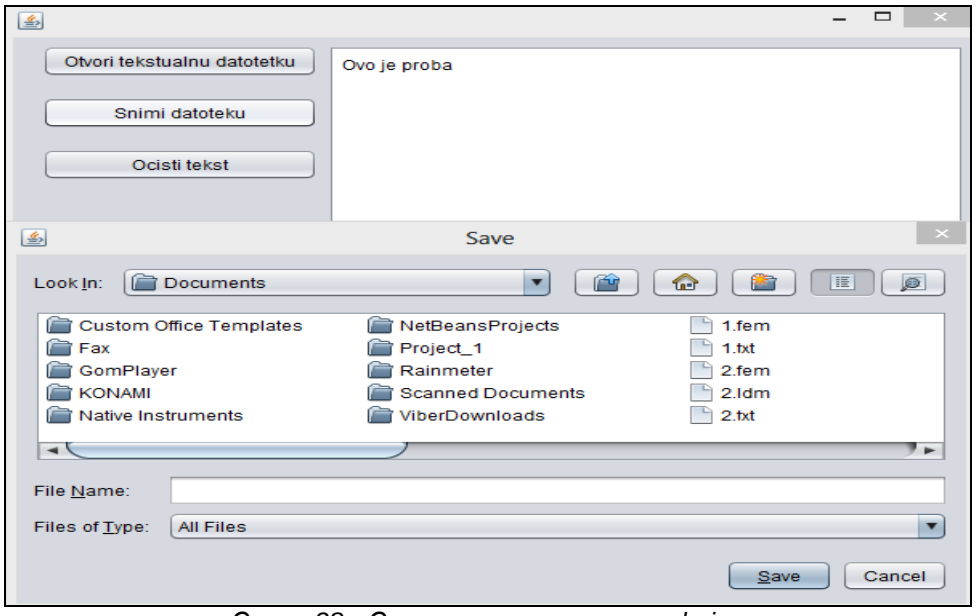
```
JFileChooser chooser = new JFileChooser();
```

„int returnVal“ добија вредност након што се отвори дијалог за отварање датотеке. Након тога се врши креирање објекта „f“, где се врши селектовање фајла. Након тога се креира објекат „br“, који служи за читање тог фајла и као што

је дефинисано чита се линија по линија, тај текст се преноси на текстуално поље унутар форме.

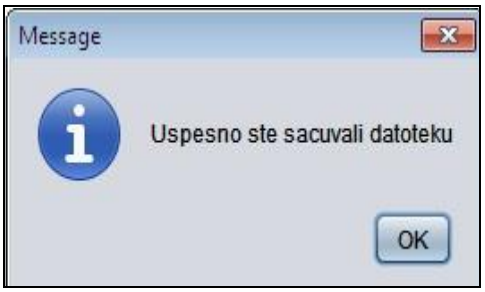
```
int returnVal = chooser.showOpenDialog(null);
if (returnVal == JFileChooser.APPROVE_OPTION)
{
File f = chooser.getSelectedFile();
try {
BufferedReader br = new BufferedReader(new FileReader(f));
String st = "";
while ((st=br.readLine())!=null)
{
jTextArea1.setText(jTextArea1.getText()+st+"\n");
}}catch(FileNotFoundExceptionrex)                                    {
Logger.getLogger(Main.class.getName()).log(Level.SEVERE,
null, ex);...
```

На наредној слици је приказано снимање текстуалног фајла. Текст је снимљен под називом „1.txt“.



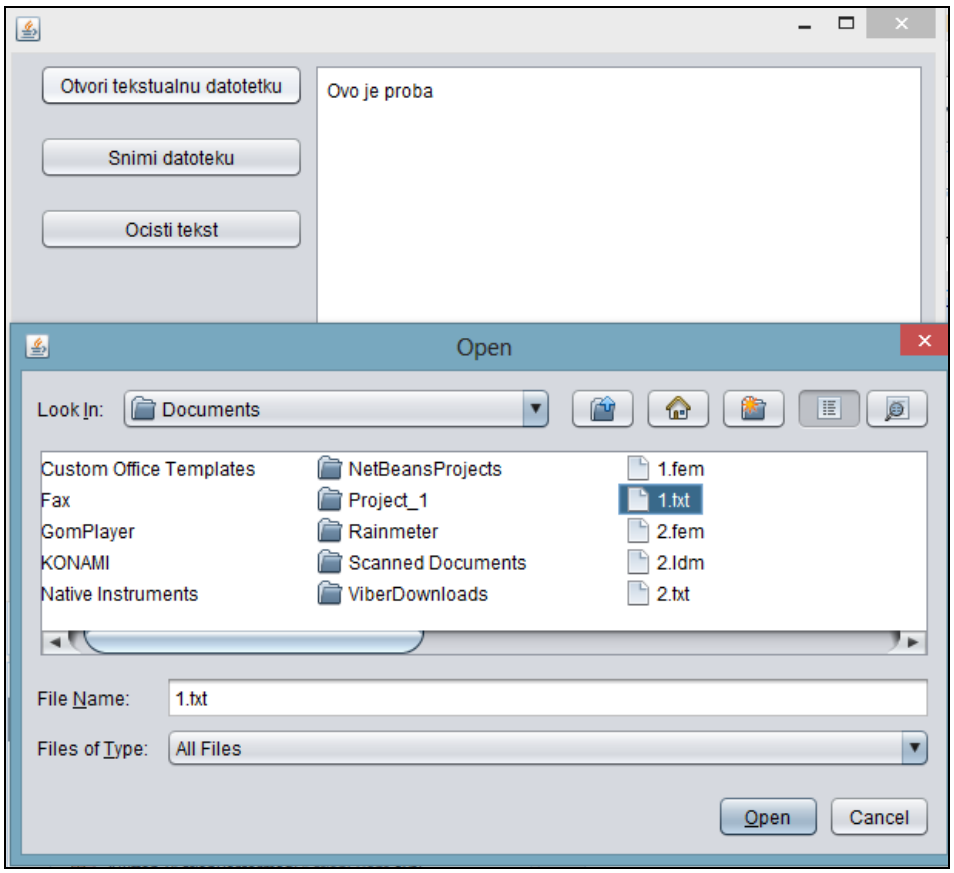
Слика 38 - Снимање текстуалног фајла

Уколико је датотека снимљена успешно, на екрану ће се појавити порука следећег садржаја: „Успешно сте сачували датотеку“, као што је и приказано на слици 39.



Слика39 - Порука

Слика 40. приказује отварање текстуалног фајла, под називом „1.txt“.



Слика 40 - Отварање текстуалног фајла

4.5. РАД СА ЗНАКОВНИМ ВЕЛИЧИНАМА

Службена реч која се користи за декларацију знаковних променљиви јесте „char“ (karakter). Кроз следећи пример биће објашњен рад са знаковним величинама. Наредна два примера су урађена као аплети.

Задатак 17: Саставити програм који исписује карактере по ASCII.

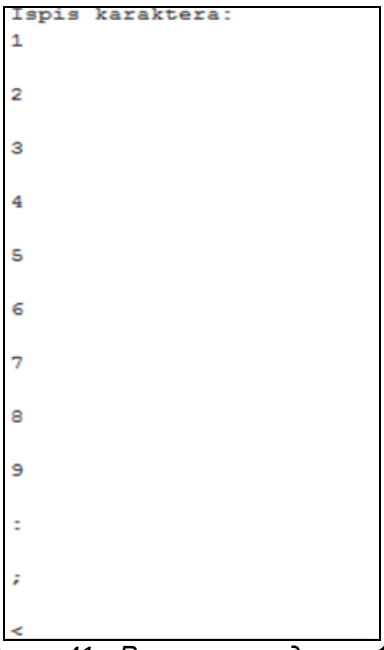
Ова класа је „public“, што значи да је могуће приступити из било које друге класе.

```
public class RadSaZnakovnimVelicinama {
/**
* @param args the command line arguments
*/
public static void main(String[] args) {
// ispis karaktera
```

Дефинисање променљиве „char“ (karakter).

```
char karakter = '0';
System.out.println("Ispis karaktera:");
for (int i = 0; i<Character.MAX_VALUE;i++)
{
karakter = (char) (karakter+1);
System.out.println(""+karakter+"\n");
}    }
}
```

Резултат рада програма је приказан на слици 41.



Слика 41 - Резултат задатка 17

Задатак 18: Саставити програм који врши креирање стрингова, карактера и низова.

Први начин креирања јесте једноставно дефинисање променљиве типа „String“, која бележи целе речи, бројеве у текстуалном формату.

```
String x = "abc";
String x1 = new String ("abc");
```

Други начин јесте креирање низа карактера, који се могу сабрати, то јест спојити у једну реч.

```
char [] ch = {'a','b','c'};
String x2 = new String(ch);
System.out.println("Prvi nacin:String x = \"abc\": "+x+"\n");
System.out.println("Drugi nacin:String x1 = new String (\"abc\"): "+x1+"\n");
System.out.println("Treci nacin:char [] ch = {'a','b','c'};\n"+"String x2=new String(ch):"+x2+"\n");
}
```

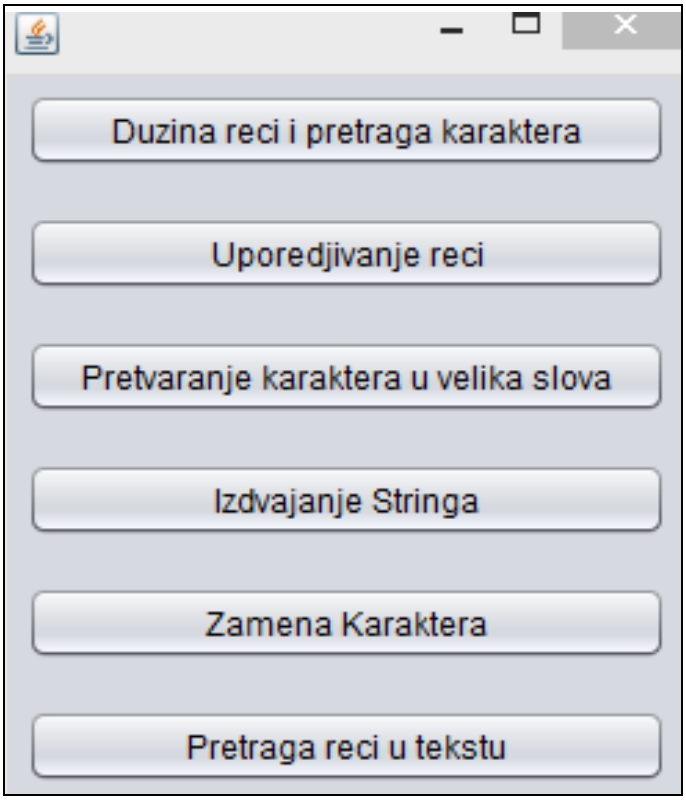
На слици је приказан резултат задатка 42.

```
Prvi nacin:String x = "abc": abc

Drugi nacin:String x1 = new String ("abc"): abc

Treci nacin:char [] ch = {'a','b','c'};
        String x2 = new String(ch): abc
```

Слика 42 - Резултат задатка 14

Следећи пример илуструје још неке од операција са стринговима и карактерима, односно знаковима у Javi.

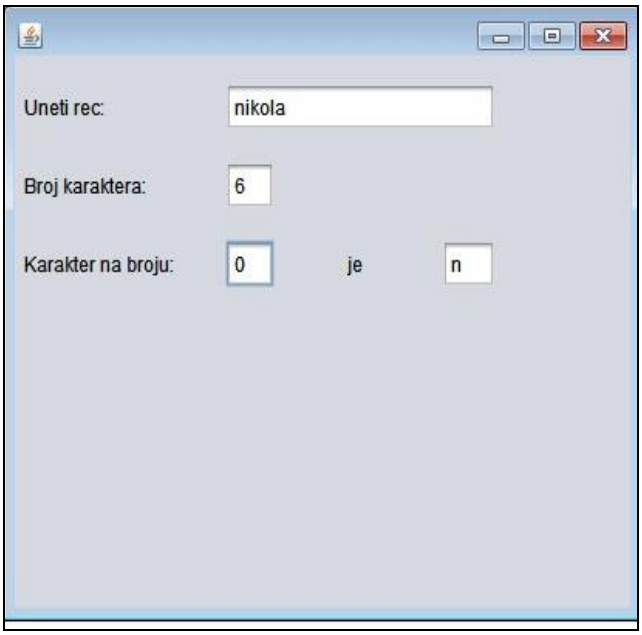
Задатак 19: Саставити програм који врши следеће функције: дужина речи и претрага карактера, упоређивање речи, претварање карактера у велика слова, издвајање стринга, замена карактера и претрага речи у тексту.



Слика 43 – Главна форма „Рад са карактерима“

- Дужина речи и претрага карактера

Дијалог на слици испод омогућава унос речи, провера њене дужине, односно од колико се карактера састоји унета реч и издвајање карактера са неке позиције.



Слика 44 – Форма „Дужина речи и претрага карактера“

Овај пример је одрађен као конзолна апликација, у наставку текста је исписан код.

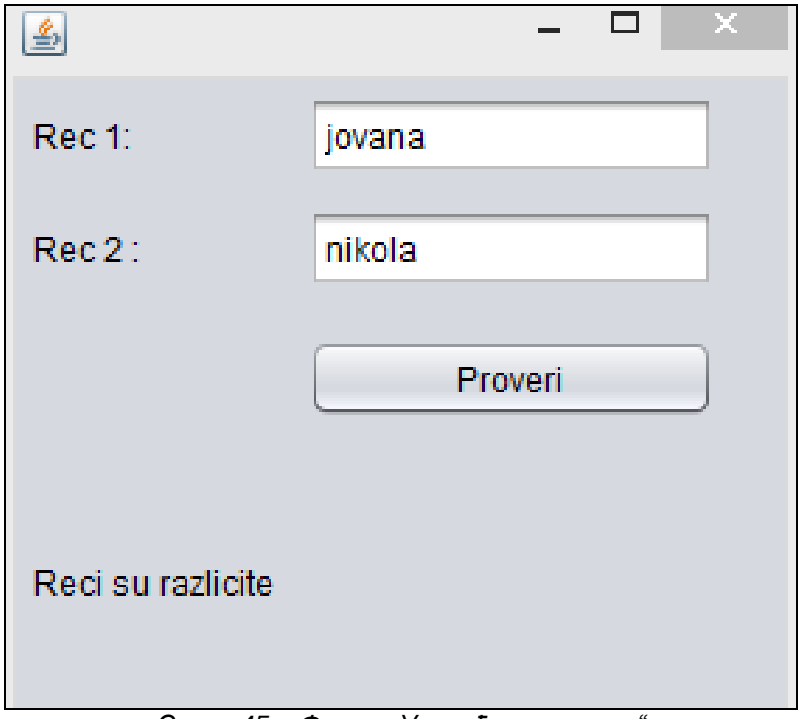
```
private void jTextField1ActionPerformed(java.awt.event.
ActionEventevt) {
rec = jTextField1.getText();
brojKaraktera = rec.length();
jTextField2.setText(""+brojKaraktera);
}
private void jTextField3ActionPerformed(java.awt.event.
ActionEventevt) {
int broj =Integer.parseInt(jTextField3.getText());
```

Метода за проверу карактера на одређеној позицији.

```
char k = rec.charAt(broj);
jTextField4.setText(""+k);
}
```

- Упоређивање речи

Наредна форма нам омогућава да упоредимо две речи. Прво је потребно унети прву реч, потом другу реч и након тога потребно је притиснути дугме „Proveri“, као што је и приказано на наредној слици. Након што се речи упореде, програм ће избацити коментар, у коме ће писати да ли су речи исте, односно различите.



Слика 45 – Форма „Упоређивање речи“

Дефинисана је променљива типа „String“, која бележи целе речи.

```
private void jButton1ActionPerformed(java.awt.event.
ActionEventevt) {
String rec1 = jTextField1.getText();
String rec2 = jTextField2.getText();
```

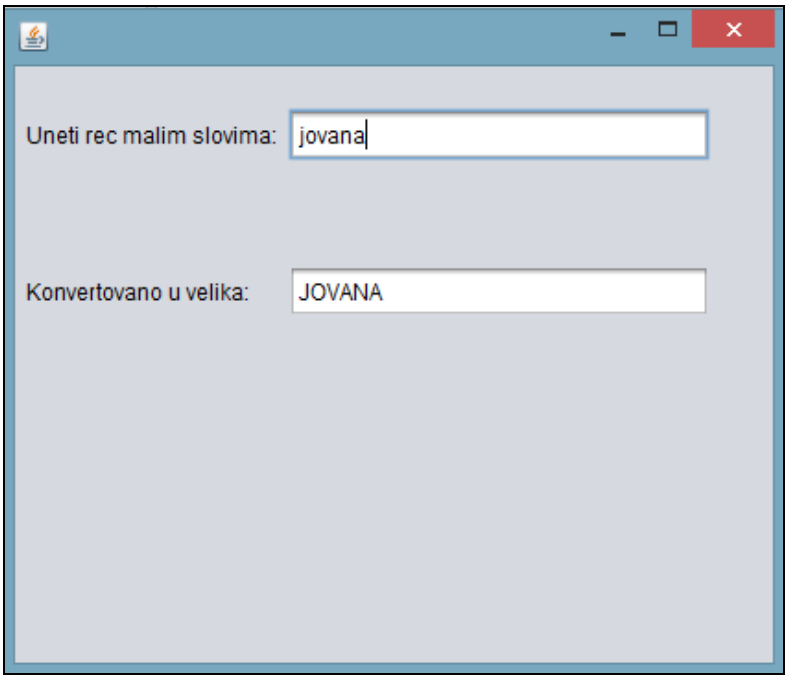
Метода „equals“ служи за упоређивање две речи, те у загради односно као параметар захтева другу реч са којом се прва реч упоређује.

```
if (rec1.equals(rec2))
{
jLabel3.setText("Reci su iste");
```

```
}else jLabel3.setText("Reci su razlicite");
}
```

- Претварање карактера у велика слова

Дијалог на слици 46. омогућава да мала слова конвертује у велика слова. Када се унесе реч малим словима у поље за текст, притиском на дугме „Enter“, аутоматски се слова конвертују у велика.



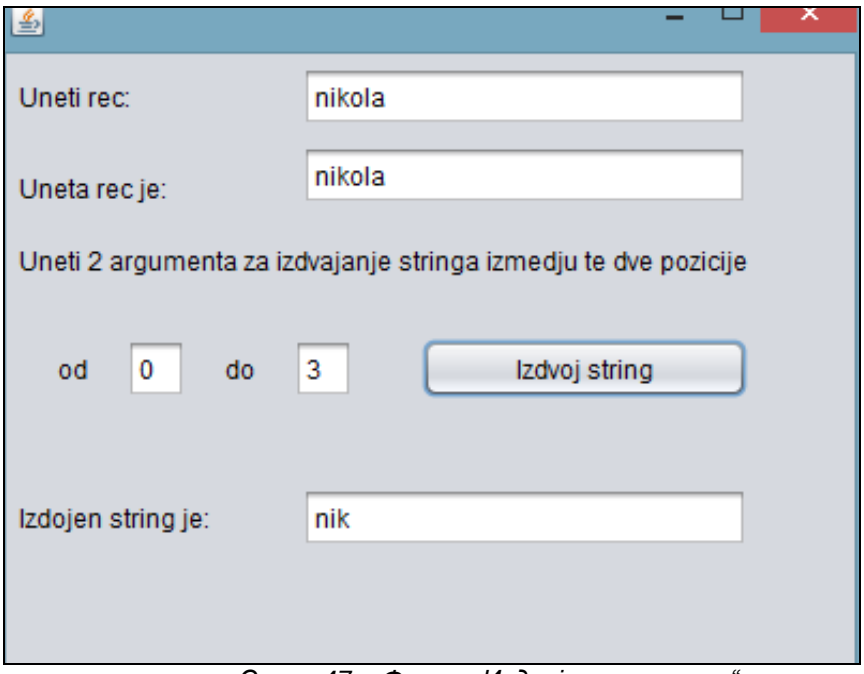
Слика 46 – Форма „Претварање карактера у велика слова“

Метода за претварање малих у велика слова.

```
String konvertovano = rec.toUpperCase();
jTextField2.setText(""+konvertovano);
```

- Издвајање стринга

Ова форма омогућава издвајање стринга из унете речи.



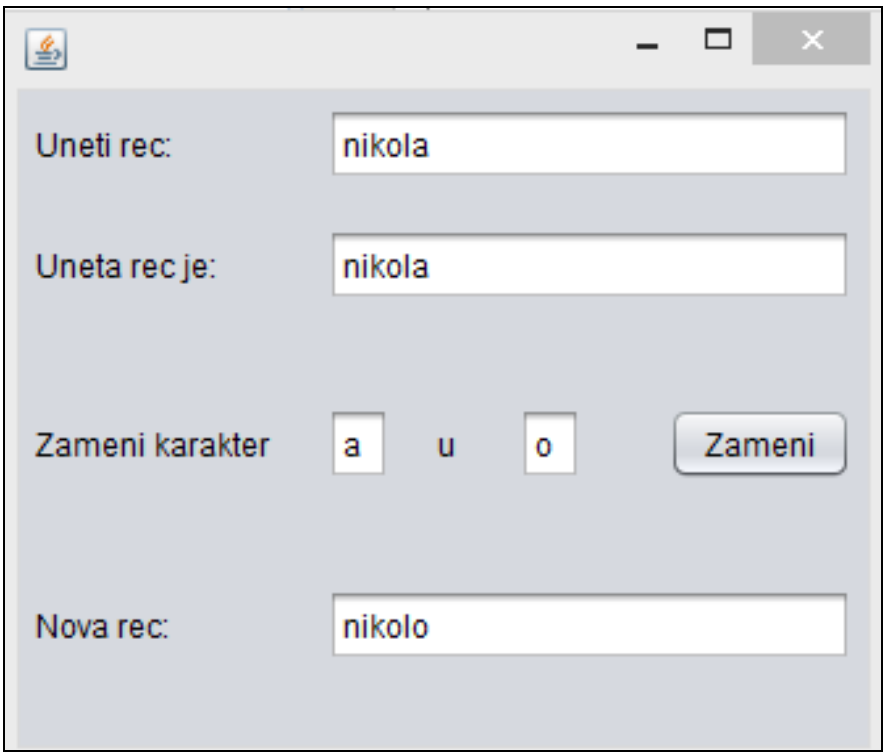
Слика 47 – Форма „Издвајање стринга“

Метода „substring“ служи за издвајање стринга, односно карактера. Захтева два параметра, први и други број.

```
String novaRec = rec.substring(par1, par2);
jTextField4.setText(""+novaRec);
```

- Замена карактера

Ова форма служи за замену карактера, као на овом примеру замена слова „a“ у „o“. Притиском на дугме „Zameni“, добијамо нову реч.



Слика 48 – Форма „Замена карактера“

Наредни код омогућава замену карактера.

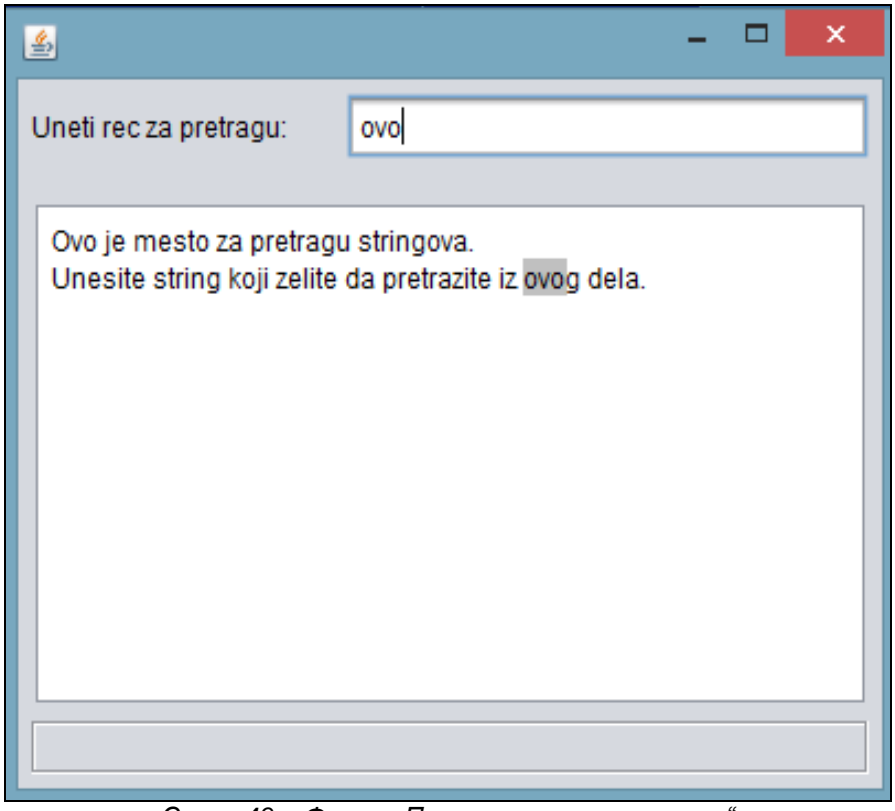
```
private void jTextField1ActionPerformed(java.awt.event.
ActionEventevt) {
rec = jTextField1.getText();
jTextField2.setText(rec);
}
private void jButton1ActionPerformed(java.awt.event.
ActionEventevt) {
String karakter1 = jTextField3.getText();
String karakter2 = jTextField4.getText();
```

Метода за замену речи, где је потребно унети карактер који се замењује, а други параметар је каракер који мења први карактер.

```
String novaRec = rec.replace(karakter1, karakter2);
jTextField5.setText(novaRec);
```

- Претрага речи у тексту

Овај дијалог служи да претражи одређену реч у задатом тексту. На наредној слици је приказана претрага речи у тексту [9].



Слика 49 – Форма „Претрага речи у тексту“

Наведени код служи за претрагу речи у тексту.

```
String sadrzaj = jTextArea1.getText();
Highlighter hilit;
hilit = new DefaultHighlighter();
Highlighter.HighlightPainter painter;
painter = new DefaultHighlighter.DefaultHighlightPainter
(Color.LIGHT_GRAY);
```

```
jTextArea1.setHighlighter(hilit);
hilit.removeAllHighlights();
String recZaPretragu = jTextField1.getText();
int broj  = sadrzaj.indexOf(recZaPretragu,0);
if (broj >= 0)
{
try
{
int kraj = broj +recZaPretragu.length();
hilit.addHighlight(broj, kraj, painter);
jTextArea1.setCaretPosition(kraj);
}
catch(Exceptione)
{}
```

5

З А К Љ У Ч А К

Пословна предузећа, друге врсте организација и појединци у савременом друштву зависе од информационих технологија. Због велике конкуренције и сталне иновације потребно је користити модерне технологије.

Употреба програмских језика и само програмирање је постало неизоставни део сакодневнице. Постоје различите врсте програмских језика које нуде различите опције и функционалности и крећу се од оних које имају мале, до оних које са собом носе велике могућности за реализовање идеја сваког корисника, као што је Java.

Java је објектно-орјентисан програмски језик и његове предности су многобројне. Почевши од тога што Java програми могу да се изршавају на скоро сваком типу рачунара, бесплатни су, једноставни за коришћење, независни од платформе на којој се изршавају, до тога што кориснику омогућавају реализацију идеја и креирање жељених апликација. Java програмски језик и његова употреба данас, у такозваном савременом добу, има велики утицај и олакшање реализације различитих видова апликација, од оних лакших, до изузетно обимних апликација чија је употреба неизоставна.

Великом предношћу Јаве сматра се и то што је, заправо, врло једноставан и лак програмски језик за учење, али без обзира на једноставност, програмски језик Java је изузетно моћан језик помоћу чијих функција и могућности корисник лакше може доћи до остварења својих идеја за креирање сопствених апликација.

6

ЛИТЕРАТУРА

[1] Ken Arnold, James Gosling and David Holmes. *The java Programming Language*. Prentice Hall, fourth edition 2006.

[2] David J. Eck. *Introduction to Programming using Java*. Free version at http://math.hws.edu/javanoted, fifth edition, 2006.

[3] Jonathan Knudsen and Patrick Niemeyer. *Learning Java*, O'Reilly, third edition, 2005.

[4] Codolini P., Davolini F., Marescotti G., Maryni P. (2007), *Developiong a distance learning system using java applets.*

[5] Healy, Berger, Michael R., Dale E. (2004), *Evaluating java applets for teaching on the Internet, Java Applets in Edcation.*

[6] Kamin N., Mickunos M. D., Reingold E. M., *An introduction to computer since using java*, McGraw-Hill, 1998.

[7] http://wikipedia.org/wiki/NetBeans.

[8] http://moodle.mfkg.rs/mod/resource/view.php?id=81

[9] http://moodle.mfkg.rs/mod/resource/view.php?id=82