PANEVROPSKI UNIVERZITET APEIRON, BANJA LUKA
FAKULTET INFORMACIONIH TEHNOLOGIJA

Redovne studije
Smjer : "Inženjering Informacionih Tehnologija"

Predmet:
PRINCIPI PROGRAMIRANJA

ZADACI U PROGRAMSKOM JEZIKU JAVA

(Seminarski rad)

Predmetni nastavnik
Prof. dr Zoran Avramović

Student: Pavlović Ivan
Br. Indeksa : 92-20/RITP-S

Banja Luka, jan 2023.

SADRZAJ

UVOD

Kreativnost i umijetnost su jedne od osnovnih ljudskih vjestina i licno smatram da programiranje pripada toj grupi. Za programski jezik Java koji je sa nama jos od 1995-te koji je mnogo doprinjeo danasnjem svijetu, a takodjer i mene motivisao da zapocnem sa samim programiranjem i napretkom u karijeri softverskog inzenjera.

Java je programski jezik koji se koriste na milion softverski inzenjera. Java se pojavila kao tehnoloski gigant zbog svoje jedinstvene prenosivosti i sposobnosti da radi slicno na bilo kom hardveru ili operativnom sistemu, takodjer je bezbijedan, jednostavan i robustan.

1. JAVA

1.1. Karakteristike programskog jezika Java

„Programski jezik Java radi na principu jezika visokog nivoa (engl. high-level-language), kao sto su Python, C, C++, i Perl.

Kao sto mozemo zakljuciti iz naziva „jezika visokog nivoa“, da takodjer postoje i jezici niskog nivoa, koji se ponekad nazivaju masinskim jezikom ili asemblerskim jezikom. Racunari mogu da pokrecu samo programe napisane jezicima niskog nivoa. Prema tome, programi napisani na jeziku visokog nivoa moraju biti prevedeni prije nego sto se pokrenu. Ovo prevodjenje zahtijeva dosta vremena i spada u jednu manu programskih jezika visih nivoa.

Mnogo je lakse programirati na jeziku visokog nivoa jer je nasem programu potrebno manje vremena za pisanje i sama sintaksa je kraca i lakse se cita. Takodjer jezici visokog nivoa su prenosivi, sto znaci da mogu da rade na razlicitim vrstama racunara sa malo ili minimalnim modifikacijama dok programi niskog nivoa mogu da rade samo na jednoj vrsti racunara i njena sintaksa bi se morala manuelno prepisivati na drugi racunar kako bi inicijalizacija bila uspijesna.

Zbog ovih prednosti, skoro svi programi su napisani na jezicima visokog nivoa. Jezici niskog nivoa se koriste samo za nekoliko posebnih aplikacija.

Postoje dva nacina za prevodjenje programa i to su „tumacenje“ (engl. interpreting) i „sastavljanje“ (engl. compiling). Interpreter je program koji cita program visokog nivoa i radi sta pise tj. on prevodi program red po red, naizmjenicno citajuci linije koda i izvrsavajuci komande. Compiler je takodjer program koji cita program visokog nivoa i prevodi sve odjednom, prije pokretanja bilo koje komande.

Java se u isto vrijeme kompajlira i interpretira. Umijesto da prevodimo nas program u masinski kod, Java kompajler generise Bajt kod (engl. Byte Code). Byte kod je veoma lak i brz za interpretovat, slican masinskom jeziku, ali je takodjer prenosiv kao jezik visokog nivoa. Moguce

je sastaviti program na jednoj masini, zatim prenijeti Byte kod na drugu masinu i nakon interpretisanja na novoj masini nas program bi uspijesno radio. Ova sposobnost je prednost programskog jezika Jave u odnosu na druge programske jezike koji su osnovani na visim nivoima“ [1].

1.2. Definicija programa

„Program je sekvenca instrukcija koje specificiraju kako da se izvrsi „proracun“ (engl. computation). Izracunavanje moze biti nesto matematicko poput rijesavanja sistem jednacina ili pronalazenje korijena polinoma, a takodjer moze biti i za simbolicno izracunavanje, poput pretrazivanja i zamijene teksta u dokumentu ili cak za sastavljanje programa.

Uputstva, koja nazivamo „iskazima“ (engl. statements), razlicito izgledaju u razlicitim programskim jezicima, ali postoji nekoliko osnovnih operacija koja vecina programskih jezika koristi:

Input : Dobijanje podataka putem tastature, datoteke ili nekog drugog uredjaja

Output : Prikazivanje podataka na ekranu ili posilja podataka drugom uredjaju

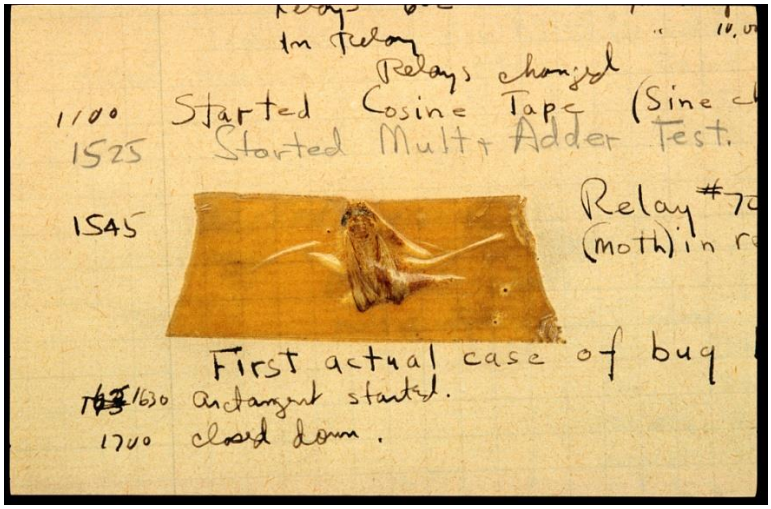
Math : Izvrsavanje osnovnih matematickih operacija kao sto su sabiranje ili mnozenje

Testing : Provjeravanje odredjenih uslova

Repetition : Izvrsavanje neke radnje n puta, obicno sa nekim varijacijama“ [1]

1.3. Otklanjanje gresaka

„Iz cudnih razloga, programske greske se nazivaju „bugs“ i proces njihovog pracenja i pronalazenja se naziva „debugging“. Postoje tri vrste gresaka koje se mogu pojaviti u programu i korisno ih je razlikovati da bih im lakse usli u trag“ [1].



Slika 1 - Prvi "Bug" u softveru

1.3.1. Sintaksne greske

„Kompajler moze da prevede program samo ako je sintaksa programa ispravna, u suprotnom, izvrsavanje kompajlera ce se zaustaviti i takodjer dolazi do prekida programa. Sintaksa se odnosi na strukturu programa i njenim pravilima o njenoj strukturi“ [1].

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
        Syntax error on tokens, delete these tokens
        Syntax error on token(s), misplaced construct(s)
        Syntax error on token ")", delete this token

        at iota_cafe3.ThreadAnimationEx.main(ThreadAnimationEx.java:27)
```

Slika 2 - Primjer greske u Sintaksi

1.3.2. Greske tokom izvodjenja

„Druga vrsta greske je greska tokom izvodjenja jer se greska nece pojaviti sve dok se program ne pokrene. U Javi, greske tokom izvodjenja se javljaju kad interpreter pokrece bajt i nesto podje naopako. Java ima tendenciju da bude bezbijedan jezik, sto znaci da kompajler hvata mnogo gresaka iako su greske u toku rada jako rijetku, pogotovo u jednostavnim programima“ [1].

```
run:
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds
for length 3
        at runtimeerrorsexample.RuntimeErrorsExample.main(RuntimeErrorsExample.java:8)
C:\Users\DELL\AppData\Local\NetBeans\Cache\12.6\executor-snippets\run.xml:111: The followin
g error occurred while executing this line:
C:\Users\DELL\AppData\Local\NetBeans\Cache\12.6\executor-snippets\run.xml:68: Java returned
: 1
BUILD FAILED (total time: 0 seconds)
```

Slika 3 - Primjer greske tokom izvodjenja

1.3.3. Logicke greske i semantika

„Treci tip greske je logicka ili semanticka greska. Ako postoji logicka greska u nasem programu, kompajler ce se pokrenuti bez generisanja poruke o gresci, ali to nece uciniti pravu stvar vec ce izvrsiti nesto drugo“ [1].

1.4. “Hello, world!“ program

Tradicionalno, prvi program koji ljudi pisu na novom programskom jeziku, jeste „Hello, World!“ zbog njegove jednostavnosti.

```
class Hello {
        public static void Main (String[] args){
                System.out.println(„Hello, World!“);
        }
}
```

Slika 4 - Hello, World!

2. Varijable i tipovi podataka

„Jedna od najmocnijih karakteristika programskog jezika je sposobnost da manipulisemo promijenljivima. Promijenjiva tzv. Varijabla je imenovana lokacija koja cuva neku vrijednost. Vrijednosti su stvari koje se mogu stampati, cuvati i upravljati sa njima.

Da bi sacuvali vrijednost, moramo kreirati vrijednost. Ako zelimo da kreiramo „String“ koja ce sacuvati neku vrijednost dacemo mu tip potaka String i ime te varijable koju kasnije mozemo pozvati u nasem programu“ [1]:

```
String ime = 'Ivan';
```

```
[tip podatka] [ime varijable] = [vrijednost koju zelimo da sacuvamo];
```

Npr.

```
class Main {
	public static void Main (String [] args){
		String ime = 'Ivan';
		System.out.println(ime);
		System.out.println('Moje ime je ' + ime);
	}
}
```

OUTPUT:

```
Pero
Moje ime je Ivan
```

„Neki tipovi podataka pocinju sa velikim slovom, a neki sa malim pocetnim slovom. Npr. da bismo kreirali cijelobrojnu promijenljivu, nasa sintaksa bi bila int number. Takodjer jako je korisno da za imena promijenljivih stavimo tako da ukazuju na to sta planiramo da uradimo sa promijenljivom“ [1].

```
int broj = 230_000;
int kamatnaStopa = 3;
String imeBanke = 'ATOS';
String opcija = 'Kamata';
```

2.1. Varijable

„Varijable su kontejneri za cuvanje vrijednosti podataka. U Javi postoje razlicite vrste promijenljivih, naprimjer:

String – skladisti tekst
int – cuva cijele brojeve
float – cuva brojeve sa decimalnom vrijednosti
char – cuva pojedinacne znakove
boolean – cuva vrijednosti tacno ili netacno“ [1]

2.2. Predstavljanje varijabli

Metoda „println();“ se veoma cesto koristi za prikazivanje varijabli. Da bismo kombinovali tekst i varijablu, koristimo znak plusa „+“.

```
String ime = 'Pero';
System.out.println('Pozdrav, ' + ime + '!');
```

Output:

```
Pozdrav, Pero!
```

2.3. Identifikatori

„Sve varijable u Javi moraju biti identifikovane jedinstvenim imenima i ta jedinstvena imena se zovu identifikatori.

Identifikatori mogu biti kratka imena kao sto su („x“ i „y“) ili vise opisana imena (broj, starost, stanjeRacuna, kalkVrijednostMase, itd.).

Strogo se preporucuje da se koriste opisna imena kako bismo kreirali razumljiv i odrzan kod.

```
// Odlicno opisano ime varijable
int minutaPoSatu = 60;

// Lose opisano ime varijable za istu vrijednost jer ne poznajemo sta tacno 'm' znaci
int m = 60;
```

Opsta pravila za imenovanje varijabli su:

- Imena mogu da sadrze slova, cifre, donje crte i znakove dolara
- Imena moraju pocinjati sa slovom
- Imena moraju da pocinju malim slovom i ne mogu da sadrze razmak
- Imena mogu da pocinju sa '_' ili '$' ali to se koristi u ekstremnim situacijama
- Imena su case-sensitive (postoji velika razlika u 'mojBroj' i 'mojbroj')
- Ne mozemo da koristimo rezervisane rijeci programskog jezika Jave“ [1]

2.4. Tipovi podataka

„Tipovi podataka u programskom jeziku Java su podijeljeni u dve grupe:

- Primitivni tipovi podataka – (byte, short, int, long, float, double, boolean, char)
- Neprimitivni tipovi podataka – (String, Arrays, Classes)“ [1]

2.4.1. Primitivni tipovi podataka

„Primitivni tip podatka odredjuje velicinu i tip vrijednosti varijabli i nema dodatke metode. U Javi postoji osam primitivnih tipova podataka“ [2].

Tip podatka	**Velicina**	**Deskripcija**
byte	1 byte	Cijeli brojevi od -128 do 127
short	2 bytes	Cijeli brojevi od -32,768 do 32,767
int	4 bytes	Cijeli brojevi od -2,147,483,648 do 2,147,483,647
long	8 bytes	-//- -9,223,372,036,854,775,808 do 9,223,372,036,854,775,807
float	4 bytes	Cuvanje od 6 do 7 decimalnih cifri
double	8 bytes	Cuvanje do 15 decimalnih cifri
boolean	1 bit	Vrijednosti tacno ili netacno
char	2 bytes	Cuva jedan znak, slovo ili ASCII vrijednost

2.4.2. Neprimitivni tipovi podataka

„Neprimitivni tipovi podataka se nazivaju referentni tipovi jer se odnose na objekte. Glavna razlika izmedju primitivnih i neprimitivnih tipova podataka je:

- Primitivni tipovi su unaprijed definisani unutar Jave dok neprimitivne tipove kreira programer i Java ih ne definise (sem String-a).
- Neprimitivni tipovi se mogu koristiti za pozivanje metoda sa sprovodjene odredjenih operacija, dok primitivni tipovi ne mogu.
- Primitivni tip uvijek ima vrijednost dok neprimitivni tip moze biti nulti (null).
- Primitivni tip podatka pocinje sa malim slovom dok neprimitivni tip pocinje sa velikim slovom
- Velicina primitivnog tipa podatka zavisi od tipa podatka dok neprimitivni tipovi imaju istu vrijednost“ [2]

2.4.3. Brojevi

„Tipovi primitivnih brojeva se dijeli u dve grupe:

Integer types – tipovi podataka koji cuvaju cijele brojeve

Floating point – tipovi podataka koji cuvaju vrijednosti sa decimalnim zarezom

Byte

Tip podatka bajta moze da skladisti cijele brojeve od -128 do 127. Ovaj tip podatka se moze koristiti umjesto „int“ ili druge vrijednosti za cuvanje cijelog broja sa ustedu memorije kada smo sigurni da ce vrijednost biti unutar prethodno pomenutog ranga.

Short

Ovaj tip podatka moze da skladisti cijele brojeve od -32768 do 32767.

Int

Tip podatka Int (tzv. Integer) moze da skladisti cijele brojeve od -2147483648 do 2147483647, takodjer vazno je napomenuti da je ovo najcesci tip podatka u kom se cuvaju vrijednosti cijelih brojeva.

Long

Long je tip podatka koji moze da skladisti cijele brojeve od -9223372036854775808 do 9223372036854775807. Ovaj tip podatka se koristi kada tip podatka „int“ nije u stanju da precisno prikaze veci broj i vazno je napomenuti da na kraju broja kada koristi Long tip podatka moramo definisati „l“ na kraju:

```
long velikiBroj = 922_337_203_685_477_580_742L;
```

Floating point Datatypes

U slucaju da trebamo prikazati broj sa decimalnim zarezom kao sto je 9,99 ili 321,32 koristicemo tip podatka „float“ ili „double“. Tipovi podataka float i double mogu da cuvaju brojeve sa decimalnim zarezom i brojevi moraju da zavrsavaju sa 'f' za float i 'd' za double“ [1].

2.4.4. Boolean tip podatka

„Veoma cesto u programiranju nam bude veoma koristan tip podatka 'boolean' koji moze imati samo jednu od dve vrijednosti“ [1]:

ON / OFF
YES / NO
TRUE / FALSE
DA / NE
JA / NEIN

Npr.

```
boolean isJavaFun = true;
boolean isUsingDrugsFun = false;
System.out.println(isJavaFun); // tacna vrijednost
System.out.println(isUsingDrugsFun); // netacna vrijednost
```

2.4.5. Char tip podatka

„Tip podatka char se koristi za cuvanje jednog znaka. Znak definisan mora biti okruzen jednostrukim zarezima“ [2].

```
char prvoSlovoImena = 'I';
char zadnjeSlovoImena = 'n';
System.out.println(prvoSlovoImena + „ „ + zadnjeSlovoImena);
```

Output:

```
I n
```

3. Komentari

„Komentari se mogu koristiti za objasnjenje Java koda i za njegovu citljivost. Takodjer se moze koristiti za sprecavanje izvrsavanja prilikom testiranja alternativnog koda“ [2].

3.1. Jednoredni komentari

„Komentari u jednom redu pocinju sa dve kose crte (//).
Svaki tekst izmedju // i kraja reda Java ignorise tj. taj kod se nece izvrsiti“ [2].

Npr.
// Ovo je komentar ← komentar
// System.out.println('pozdrav'); ← komentar
System.out.println('pozdrav'); ← funkcija za ispis vrijednosti na ekranu
System.out.println('pozdrav'); // ispisi pozdrav ← funkcija + komentar na kraju

3.2. Viselinijski komentari

„Komentari sa vise redova pocinju sa /* i zavrsavaju se sa */. Svaki tekst izmedju /* i */ Java ce ignorisati“ [2].

Primjer viselinijskog komentara u Javi:

```
/* Ovo je jedan
veliki
komentar
koji
trenutno
nema
smisla
*/
```

4. Java uslovi

4.1. IF ... ELSE IF ... ELSE uslovi

„Programski jezik Java podrzava uobicajene logicke uslove iz matematike kao sto su: <, <= , >, >=, ==, !=.

Ove uslove mozemo koristiti za obavljanje razlicitih radnji za razlicite odluke. Uslovi izjava koje se koriste u programskom jeziku Java su“ [1]:

if – da navedemo blok koda koji treba da se izvrsi ako je odredjeni uslov tacan

else – da navedemo blok koda koji treba da se izvrsi ako je odredjeni uslov netacan

else if – da navedemo novi uslov za testiranje ako je prvi uslov netacan

switch - da navedemo mnoge alternativne blokove koda koji ce se izvrsiti

Primjer 'if'' uslova:

```
if (20 > 18) {
    System.out.println(„Broj 20 je veci od broja 18!“);
}
```

,takodjer mozemo testirati varijable:

```
int x = 20;
int y = 18;
if (x > y) {
    System.out.println(„Vrijednost x je veca od vrijednosti y!“);
}
```

Primjer 'else" uslova:

```
int vrijeme = 20;
if (vrijeme < 18) {
        System.out.println(„Dobar dan!“);
}
else {
        System.out.println(„Dobro vece!“);
}
```

Primjer 'else if" uslova:

```
int vrijeme = 22;
if (vrijeme < 10) {
        System.out.println(„Dobro jutro!“);
}
else if (vrijeme < 18) {
        System.out.println(„Dobar dan!“);
}
else {
        System.out.println(„Dobro vece!“);
}
```

„Postoji takodjer i skracenica za „if else“ uslov, koji je poznat kao ternarni (engl. ternary) operator jer se sastoji od tri operanda. Moze se koristiti za zamijenu vise redova koda jednom linijom, a najcesce se koristi za zamjenu jednostavnih if else izjava“ [1].

```
varijabla = (uslov) ? ekspresijaTacna  : ekspresijaNetacna;
```

4.2. Switch

„Umjesto pisanja mnogobrojnih if...else uslova, mozemo koristiti naredbu **switch**. Naredba switch bira jedan od mnogih definisanih blokova koji ce se izvrsiti“ [2].

```
switch ( ekspresija ) {
        case x:
                // kod koji zelimo da izvrsimo ako je odabrana vrijednost x
                break;
        case y:
                // kod koji zelimo da izvrsimo ako je odabrana vrijednost y
                break;
        case z:
                // kod koji zelimo da izvrsimo ako je odabrana vrijednost z
                break;
        default:
                // ispis koda u slucaju da ni jedan case nije bio odabran
                break;
}
```

4.3. While petlja

„Petlje mogu da izvrse kod sve dok se postigne odredjeni uslov. Prelje su koriste jer ustede dosta vremena, smanjuju velicinu koda i samim tim cine ga citljivim. While petlja prolazi kroz blok koda sve dok je odredjeni uslov istinit“ [2]:

```
int i = 0;
while (i < 5) {
        System.out.print(i);
        i++;
}
```

Output: 12345

4.4. For pretlja

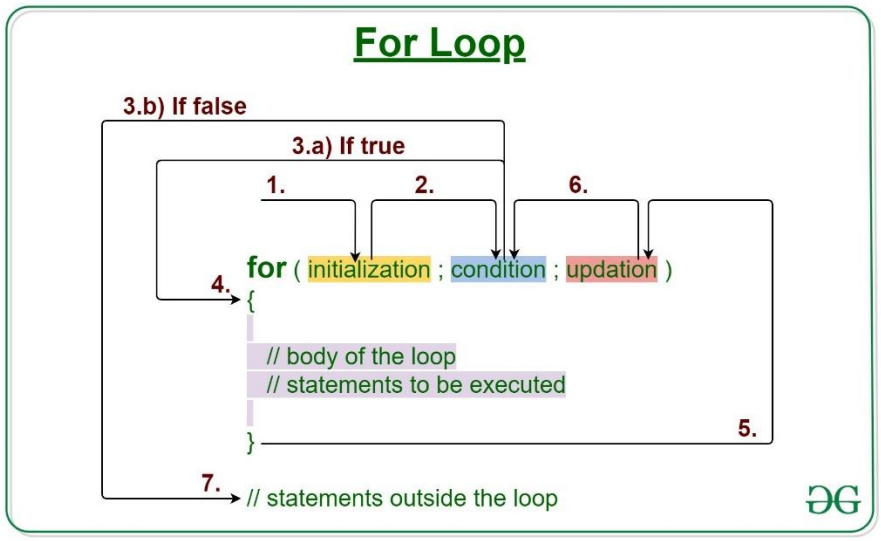
„Kada tacno znamo koliko puta zelimo da prodjemo kroz blok koda koristimo For petlju umjesto While petlje.

```
for (int i = 0; i < 5 ; i++) {
        System.out.println(i);
}
```

Prva vrijednost postavlja promijenljivu prije pocetka petlje – int i = 0

Druga vrijednost definise uslov za pokretanje pretlje i ako je uslov tacan, petlja ce poceti iznova, a ako je netacna petlja ce se zavrsiti – i < 5

Treca vrijednost povecava vrijednost definisane varijable „i“ (i++ ili i=i+1 ili i+=1) svaki put kada se izvrsi blok koda u petlji“ [2].



Slika 5 - Slikoviti prikaz For petlje

5. Nizovi (Arrays) u Javi

„Nizovi se koriste za cuvanje vise podataka u jednoj varijabli, umjesto da deklarisemo zasebne varijable za svaku vrijednost. Da bih smo deklarisali niz, definisemo tip promjenljive unutar uglastih zagrada“ [2].

Primjer niza koji sadrzi String vrijednosti:

```
String[] auta = {„BMW“, „Mercedes-Benz“, „Ford“, „Opel“};
```

Primjer niza koji sadrzi Integer vrijednosti:

```
int[] brojevi = {1, 2, 1342, 42, 420, 69};
```

5.1. Pristup elemenata niza

Elementu niza mozemo pristupiti pozivanjem njegovoj broja indeksa.

Primjer niza „auta“ gdje zelimo da selektujemo drugi element:

```
String[] auta = {„BMW“, „Mercedes-Benz“, „Ford“, „Opel“};
System.out.println([1]);
```

Output:

```
Mercedes-Benz
```

Primjer niza „brojevi“ gdje zelimo da selektujemo prvi element:

```
int[] brojevi = {1, 2, 1342, 42, 420, 69};
System.out.println(brojevi[0]);
```

Output:

```
1
```

5.2. Promjena vrijednosti elemenata u nizu

Da bismo promjenili vrijednost odredjenog elementa trebamo promjeniti odredjeni indeks.

```
String[] auta = {„BMW“, „Mercedes-Benz“, „Ford“, „Opel“};
auta[0] = „Citroen“;
System.out.println(auta[0]);
```

Output:

```
Citroen
```

5.3. Duzina niza

Kako bismo saznali koliko elemenata je prisutno u odredjenom nizu, koristimo metodu **lenght**. Ova metoda je veoma korisna kad zelimo inicijalizovati For petlju koja ce prolaziti kroz niz i umjesto da manuelno unosimo kolicinu elemenata to mozemo sprijeciti i ustediti na vremenu kad koristimo lenght metodu. Npr:

```
int[] brojevi = {1, 2, 3, 4};
for(int i = 0; i < brojevi.lenght; i++){
    .
    .
    .
```

6. Java metode

„Metoda je blok koda koji se pokrece samo kad je pozvan u koje mozemo proslijediti podatke, poznate kao parametre u metodu. Metode se koriste za obavljanje odredjenih radnji, a poznate su kao i funkcije. Metode se najcesce koriste iz razloga da ne ponavljamo nas kod u smislu da iznova i iznova pisemo istu funkciju“ [1].

6.1. Kreiranje metoda

„Metoda mora biti deklarisana unutar klase koju definisemo imenom metode, praceno zagradama. Programski jezik Java pruza neke unaprijed definisane metode, kao sto je System.out.println(), ali takodjer moze kreirati sopstvene metode za obavljanje odredjenih radnji“ [2].

```
public class Main {
    static void mojaMetoda() {
        // blok koda
    }
}
```

mojaMetoda – ime metode
static – znaci da metoda pripada klasi Main, a ne objektu klase main
void – znaci da ovaj metod nema povratnu vrijednost

6.2. Pozivanje metoda

Kako bismo pozvali metodu u programskom jeziku Java, potrebno je napisati ime metoda sa dve zagrade i tackom-zarezom na kraju.

Primjer pozivanja mojaMetoda metoda:

```
public class Main {
	static void mojaMetoda() {
		System.out.println('Pozdrav brate');
	}

	Public static void Main (String[] args) {
		mojaMetoda();
	}
}

// OUTPUT: Pozdrav brate
```

Metode takodjer mozemo pozvati vise puta.

6.3. Paremetri metoda

„Informacije se mogu prenositi metodama kao parametar, a parametri se ponasaju kao promjenljive unutar metode. Parametri se definisu iza naziva metode unutar zagrada i mozemo vise parametara imati ali moramo ih razdvojiti zarezom“ [2].

```
public class Main {
	static void mojaMetoda(String ime) {
		System.out.println( ime + ' studira na Apeironu');
	}
	Public static void main (String[] args) {
		mojaMetoda('Ivan');
		mojaMetoda('Pavle');
	}
}
// Ivan studira na Apeironu
// Pavle studira na Apeironu
```

6.4. Java obim

U programskom jeziku Java, promijenljive su dostupne samo unutar regiona u kom su kreirane i to se zove obim (engl. „Scope“).

6.4.1. Obim metoda

Varijable koje su deklarisane direktno unutar metode, su dostupne bilo gdje u metodi nakon linije koja u kojoj su deklarisane.

```
public class Main {
	public static void main (String[] args) {
		// Kod ovdje ne moze da koristi vrijednost – x

		int x = 100;

		// Kod ovdje moze da koristi vrijednost – x
		System.out.println(x);
	}
}

// OUTPUT: 100;
```

6.4.2. Obim bloka

Obim bloka se odnosi na sav kod izmedju viticastih zagrada. Varijable deklarisane unutar blokova koda su dostupne samo kodom unutar viticastih zagrada, koji prati red u kome je varijabla deklarisana.

```
public class Main {
    public static void main(String[] args) {
        // Kod ovdje ne moze da koristi vrijednost – x

        { // Ovo je blok
            // Kod ovdje ne moze da koristi vrijednost – x

            int x = 100;

            // Kod ovdje moze da koristi vrijednost – x
            System.out.println(x);

        } // Zavrsetak bloka
        // Kod ovdje ne moze da koristi vrijednost – x
    }
}
```

Blok koda moze postojati sam ili moze pripadati if, while ili for uslovu. U slucaju For uslova, varijable deklarisane u samom iskazu su takodjer dostupne unutar opsega bloka.

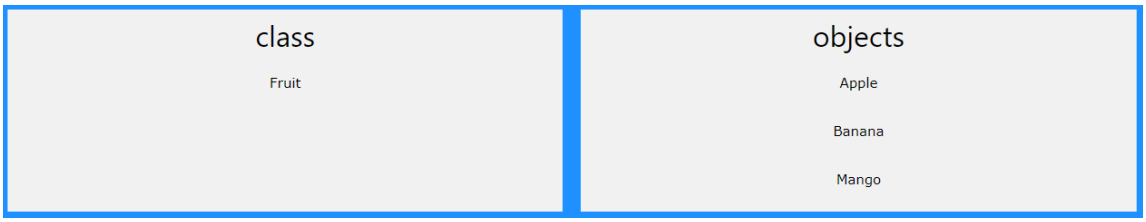
7. Java OOP

„OOP je skracenica za objektno orijentisano programiranje. Proceduralno programiranje se odnosi na pisanje procedura ili metoda koje izvode operacije nad podacima, dok je obijetkno orijentisano programiranje stvaranje objekata koji sadrze i podatke i metode.

Objetko orijentisano programiranje ima nekoliko prednosti u odnosu na proceduralno programiranje:

- OOP je brzi i laksi za izvrsavanje
- OOP pruza jasnu strukturu za programe
- OOP pomaze da se Java kod odrzi 'Suvim' (da se ne ponavlja), i cini kod laksim sa odrzavanje, modifikaciju i otklanjanje gresaka
- OOP omogucava kreiranje aplikacija za visekratnu upotrebu sa manje koda

Klase i Objekti su dva glavna aspekta objektno orijentisanog programiranja i u ilustraciji ispod mozemo da vidimo razliku izmedju njih“ [2].



Slika 6 - OPP Class / Objects

7.1. Klase i Objekti u Java OOP

„Sve u programskom jeziku Java je povezano klasama i objektima, zajedno sa njenim atributima i metodama. Klasa je poput konstruktora objekata ili 'nacrta-plana' za kreiranje objekata“ [1].

7.1.1. Kreiranje Klase

Kako bismo kreirali klasu u programskom jeziku Java koristimo kljucnu rijec 'class'.

```
public class Main {
    int x = 5;
}
```

7.1.2. Kreiranje Objekta

U programskom jeziku Java, objekti se kreiraju iz Klase. Da bismo kreirali objekat Main, moramo navesti ime klase, a zatim ime objekta i koristiti kljucnu rijec 'new'.

```
public class Main {
    int x = 5;

    public static void main (String[] args) {
        Main mojObjekat = new Main();
        System.out.println(mojObjekat.x);
    }
}
```

7.2. Nasljedje u Javi

„U programskom jeziku Java je moguce naslijediti atribute i metode iz jedne klase u drugu. Grupisemo 'koncept naslijedja' u dve kategodije:

- subclass (child) – klasa koja naslijedjuje drugu klasu
- superclass (parent) – klasa od koje se naslijedjuje

Da bismo naslijedili klasu, koristimo kljucnu rijec 'extends"“ [2].

7.3. Java paketi i API

„Paketi u Javi se koriste za grupisanje srodnih klasa. Koristimo pakete da bismo izbjegli sukobe izmedju imena i da bismo napisali kod koji se bolje i lakse odrzava. Paketi su podjeljeni u dve grupe:

- Built-in Packets (ugradjeni paketi iz Java API-a)
- User-defined Packets (korisnicki definisani paketi)“ [2]

7.3.1. Ugradjeni paketi Java API-a

„Java API je biblioteka unaprijed napisanih klasa, koje su besplatne za koristenje, ukljucene u Java razvojno okruzenje.

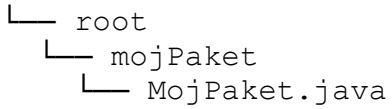
Biblioteka sadrzi komponente za upravljanje unosom, programiranje baze podataka i jos mnogo toga. Kompletna lista se moze naci na Oracle veb sajtu. Biblioteka je podijeljena na pakete i klase sto znaci da uvezemo jednu klasu (zajedno sa njenim metodama i atributima) ili cijeli paket koji sadrzi sve klase koje pripadaju navedenom paketu.

Da bismo koristili klasu ili paket iz biblioteke, protrebno je koristiti kljucnu rijec 'import'.

```
import package.name.Class;   // Importovanje jedne klase iz paketa
import package.name.*;       // Importovanje citavog paketa“ [2]
```

7.3.2. Korisnicki definisani paketi

Da bismo kreirali sopstveni paket, moramo da razumijemo da Java koristi direktorijum sistema datoteka za njihovo skladistenje, kao fascikle na nasim personalnim racunarima.



Kako bismo kreirali paket, moramo koristiti kljucnu rijec 'package'.

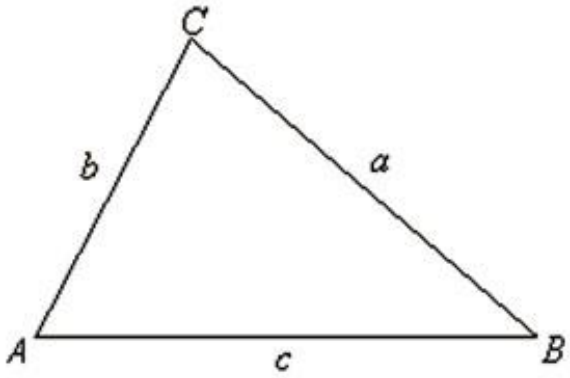
MojPaket.java

```
package mojPaket;

class MojPaket {
	public static void main (String[] args) {
		System.out.println('Ovo je moj paket');
	}
}
```

8. Zadaci u programskom jeziku Java

8.1. Izracunavanje povrsine trougla pomocu unesenih stranica

U programu 'PovrsinaTrougla' koristimo klasu Scanner kako bih dobili unos od korisnika, jer u ovom programu od korisnika se trazi da unese tri vrijednosti stranica trougla. Za ovaj program koristena je Heronova formula koja sluzi za izracunavanje povrsine trougla cije stranice imaju duzinu a, b, c.



$$P = \sqrt{s(s-a)(s-b)(s-c)}$$

$$s = \frac{a+b+c}{2}$$

Slika 7 - Herenova formula

Tekstualni prikaz programa za izracunavanja povrsine trougla

```
import java.util.Scanner;

public class PovrsinaTrouglav2 {

    public static void main(String[] args) {
        Scanner unos = new Scanner(System.in);

        System.out.println("Unesite duzinu prve stranice: ");
        double prvaStr = unos.nextDouble();

        System.out.println("Unesite duzinu druge stranice: ");
        double drugaStr = unos.nextDouble();

        System.out.println("Unesite duzinu trece stranice: ");
        double trecaStr = unos.nextDouble();

        double stranice = (prvaStr + drugaStr + trecaStr) / 2.0;
        double povrsina = Math.sqrt(stranice * (stranice - prvaStr) * (stranice - drugaStr) * (stranice
- trecaStr));

        System.out.println("Porvsina trougla je: " + povrsina);
    }
}
```

Prikaz programa za izracunavanja povrsine trougla u NetBeans IDE

```java
package com.mycompany.povrsinatrouglav2;


import java.util.Scanner;

public class PovrsinaTrouglav2 {

    public static void main(String[] args) {
        Scanner unos = new Scanner( source: System.in);

        System.out.println( x: "Unesite duzinu prve stranice: ");
        double prvaStr = unos.nextDouble();

        System.out.println( x: "Unesite duzinu druge stranice: ");
        double drugaStr = unos.nextDouble();

        System.out.println( x: "Unesite duzinu trece stranice: ");
        double trecaStr = unos.nextDouble();

        double stranice = (prvaStr + drugaStr + trecaStr) / 2.0;
        double povrsina = Math.sqrt(stranice * (stranice - prvaStr) * (stranice - drugaStr) * (stranice - trecaStr));

        System.out.println("Porvsina trougla je: " + povrsina);
    }
}
```

Slika 8 - Kod za izracunavanje povrsine trougla

```
Output - Run (PovrsinaTrouglav2)
Compiling 1 source file to C:\Users\Hasbulla\Documents\NetBeansProjects\PovrsinaTrouglav2\target\classes

--- exec-maven-plugin:3.0.0:exec (default-cli) @ PovrsinaTrouglav2 ---
Unesite duzinu prve stranice:
13
Unesite duzinu druge stranice:
12
Unesite duzinu trece stranice:
4
Porvsina trougla je: 23.894298483110987
------------------------------------------------------------------------
BUILD SUCCESS
```

Slika 9 - Rezultat izracunavanja povrsine trougla

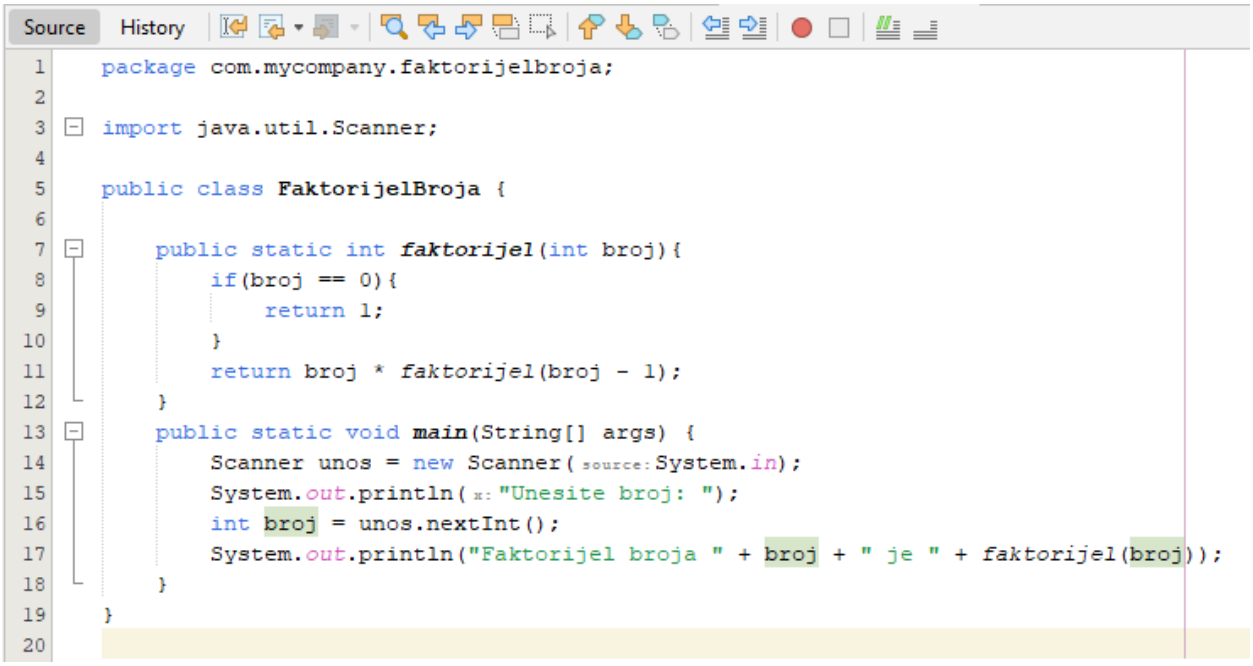
8.2. Izracunavanje Faktorijela unesenog broja

Ovaj program koristi klasu Scanner da procita unos korisnika, a zatim poziva rekurzivnu funkciju 'faktorijel (broj)' koja uzima cijeli broj kao unos i vraca njegovu faktorijel vrijednost. Ova funkcija provjerava da li je unos jednak 0 i ako jeste vraca 1, sto je osnovni slucaj rekurzije. Ako unos nije jednak nuli, funkcija poziva samu sebe, ali ovaj put propustajuci unos – 1, na ovaj nacin, funkcija nastavlja da poziva sebe smanjujuci unos za 1 svaki put dok ne dodje do osnovnog slucaja.

Tekstualni prikaz programa za izracunavanja faktorijela unesenog broja

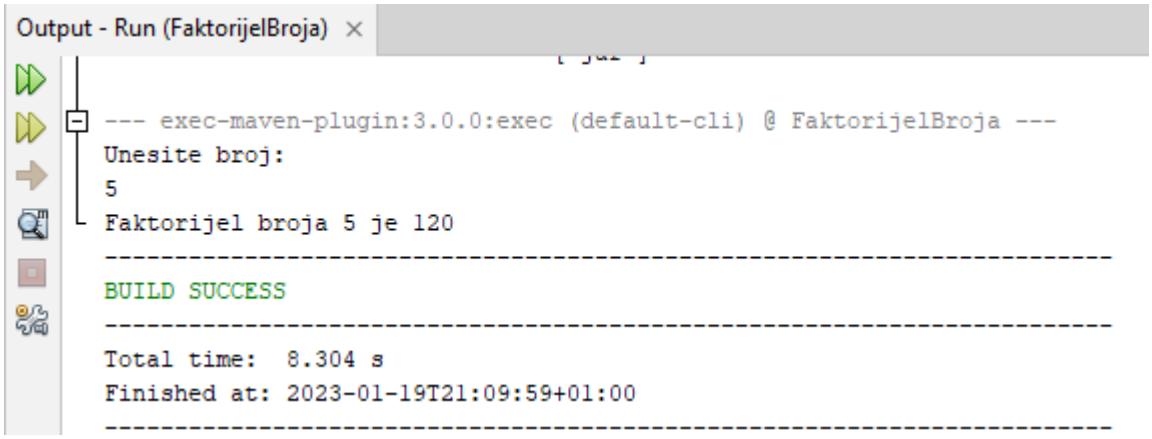
```
import java.util.Scanner;

public class FaktorijelBroja {

    public static int faktorijel(int broj){
        if(broj == 0){
            return 1;
        }
        return broj * faktorijel(broj - 1);
    }
    public static void main(String[] args) {
        Scanner unos = new Scanner(System.in);
        System.out.println("Unesite broj: ");
        int broj = unos.nextInt();
        System.out.println("Faktorijel broja " + broj + " je " + faktorijel(broj));
    }
}
```

Prikaz programa za izracunavanja faktorijela broja u NetBeans IDE



```
1  package com.mycompany.faktorijelbroja;
2
3  import java.util.Scanner;
4
5  public class FaktorijelBroja {
6
7      public static int faktorijel(int broj){
8          if(broj == 0){
9              return 1;
10         }
11         return broj * faktorijel(broj - 1);
12     }
13     public static void main(String[] args) {
14         Scanner unos = new Scanner( source: System.in);
15         System.out.println( x: "Unesite broj: ");
16         int broj = unos.nextInt();
17         System.out.println("Faktorijel broja " + broj + " je " + faktorijel(broj));
18     }
19 }
20
```

Slika 10 - Kod za izracunavanje faktorijela broja



```
Output - Run (FaktorijelBroja)

--- exec-maven-plugin:3.0.0:exec (default-cli) @ FaktorijelBroja ---
Unesite broj:
5
Faktorijel broja 5 je 120
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  8.304 s
Finished at: 2023-01-19T21:09:59+01:00
------------------------------------------------------------------------
```

Slika 11 - Rezultat izracunavanja faktorijela broja

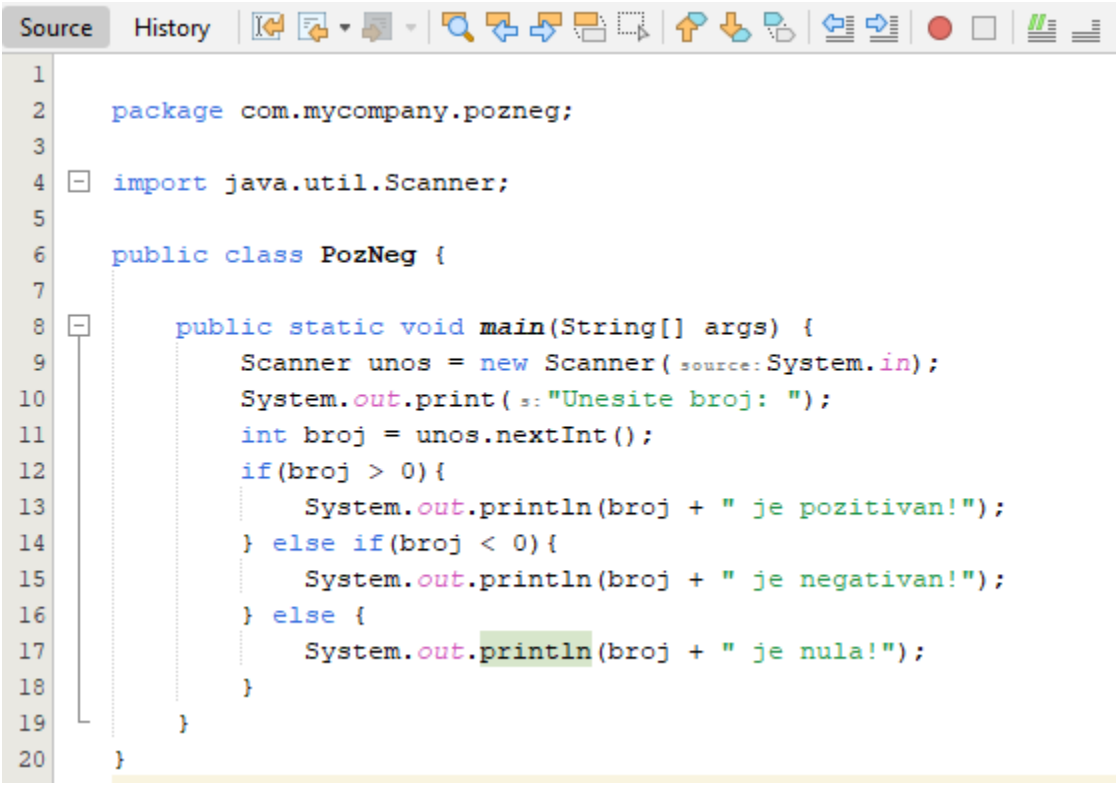
8.3. Provjera unesenog broja da li je pozitivan, negativan ili nula

Program koristi Scanner klasu da procita broj od korisnika, zatim provjerava da li je broj veci od 0, ako je uslov tacan na ekranu ispusuje '[unos] je pozitivan'. Ako je uneseni broj manji od nule na ekranu se ispisuje '[unos] je negativan' i ako je uneseni broj jednak nuli na ekranu se ispisuje da je broj nula.

Tekstualni prikaz programa za provjeru da li je broj pozitivan, negativan ili nula

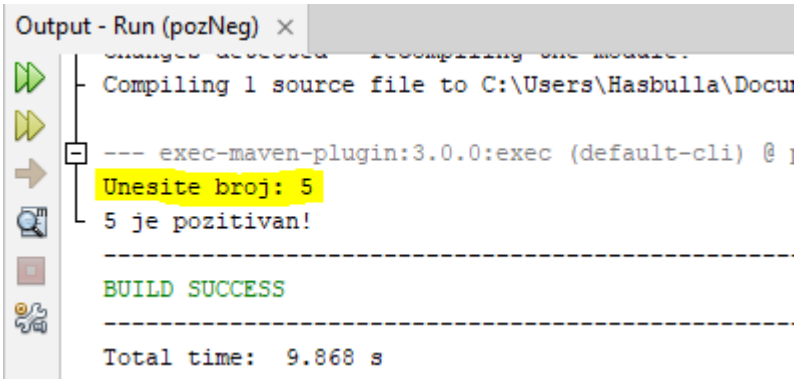
```
import java.util.Scanner;

public class PozNeg {

    public static void main(String[] args) {
        Scanner unos = new Scanner(System.in);
        System.out.print("Unesite broj: ");
        int broj = unos.nextInt();
        if(broj > 0){
            System.out.println(broj + " je pozitivan!");
        } else if(broj < 0){
            System.out.println(broj + " je negativan!");
        } else {
            System.out.println(broj + " je nula!");
        }
    }
}
```

Prikaz programa za provjeru broja da li je pozitivan, negativan ili nula u NetBeans IDE



```
1
2   package com.mycompany.pozneg;
3
4   import java.util.Scanner;
5
6   public class PozNeg {
7
8       public static void main(String[] args) {
9           Scanner unos = new Scanner( source: System.in);
10          System.out.print( s: "Unesite broj: ");
11          int broj = unos.nextInt();
12          if(broj > 0){
13              System.out.println(broj + " je pozitivan!");
14          } else if(broj < 0){
15              System.out.println(broj + " je negativan!");
16          } else {
17              System.out.println(broj + " je nula!");
18          }
19      }
20  }
```

Slika 12 - Kod za provjeru broja da li je pozitivan, negativan ili nula



```
Output - Run (pozNeg)
Compiling 1 source file to C:\Users\Hasbulla\Docu
--- exec-maven-plugin:3.0.0:exec (default-cli) @
Unesite broj: 5
5 je pozitivan!
------------------------------------------------
BUILD SUCCESS
------------------------------------------------
Total time:  9.868 s
```

Slika 13 - Rezultat provjere broja da li je pozitivan, negativan ili nula

8.4. Provjera unesenog broja da li je paran ili neparan

Ovaj program koristi takodjer klasu Scanner da dobije korisnicki unos, a zatim koristi operator % da provjeri da li je uneseni broj paran ili neparan. Ako je rezultat 'broj % 2 == 0' onda je paran, u suprotnom je neparan.

Tekstualni prikaz programa za provjeru da li je uneseni broj paran ili neparan

```
public class EvenOdd {

    public static void main(String[] args) {
        Scanner unos = new Scanner(System.in);
        System.out.print("Unesite broj: ");
        int broj = unos.nextInt();
        if (broj % 2 == 0) {
            System.out.println(broj + " je paran.");
        } else {
            System.out.println(broj + " je neparan.");
        }
    }
}
```

Prikaz programa za provjeru unesenog broja da li je paran ili neparan u NetBeans IDE

```java
package com.mycompany.evenodd;

import java.util.Scanner;

public class EvenOdd {

    public static void main(String[] args) {
        Scanner unos = new Scanner( source: System.in);
        System.out.print( s: "Unesite broj: ");
        int broj = unos.nextInt();
        if (broj % 2 == 0) {
            System.out.println(broj + " je paran.");
        } else {
            System.out.println(broj + " je neparan.");
        }
    }
}
```

Slika 14 - Kod za provjeru da li je uneseni broj paran ili neparan

```
Output - Run (evenOdd)
Compiling 1 source file to C:\Users\Hasbulla\

--- exec-maven-plugin:3.0.0:exec (default-cli
Unesite broj: 5
5 je neparan.
------------------------------------------------
BUILD SUCCESS
------------------------------------------------
```

Slika 15 - Rezultat provjere broja da li je paran ili neparan

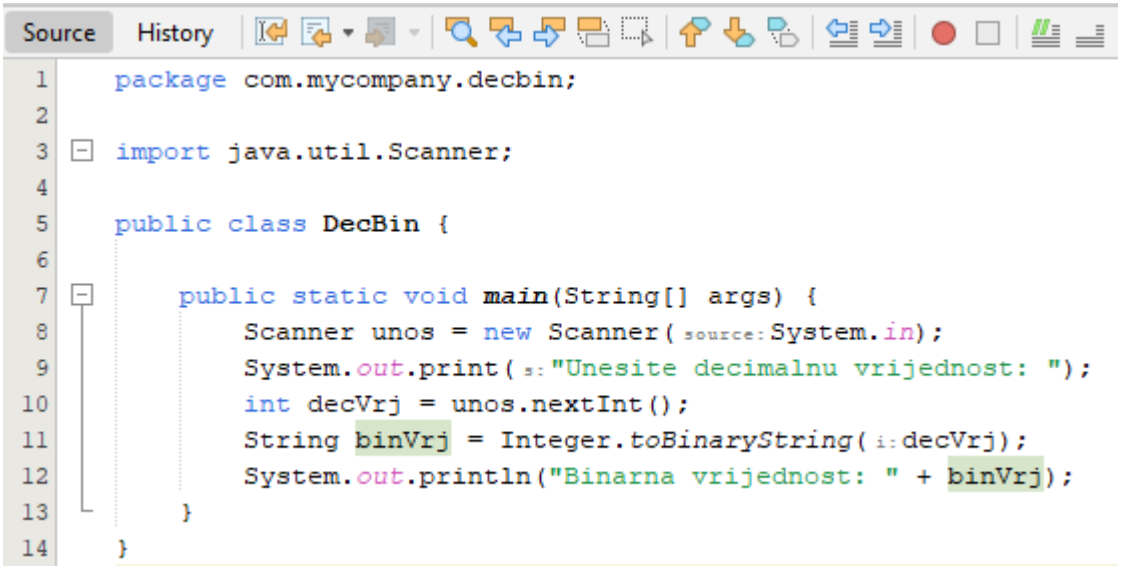
8.5. Konverter decimalne vrijednosti u binarnu

Ovaj program koristi klasu Scanner da dobije korisnicki unos, a zatim koristi metodu 'Integer.toBinaryString' da konvertuje decimalni broj u binarni. Rezultat se cuva u String promijenjlivoj i zatim se stampa na ekranu.

Tekstualni prikaz programa za konverziju decimalne vrijednosti u binarnu

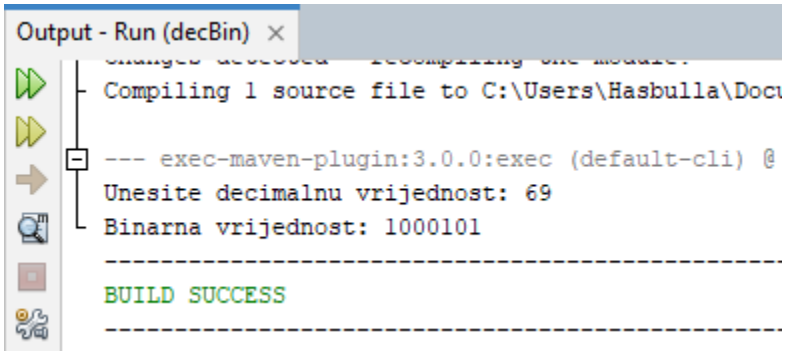
```
import java.util.Scanner;

public class DecBin {

    public static void main(String[] args) {
        Scanner unos = new Scanner(System.in);
        System.out.print("Unesite decimalnu vrijednost: ");
        int decVrj = unos.nextInt();
        String binVrj = Integer.toBinaryString(decVrj);
        System.out.println("Binarna vrijednost: " + binVrj);
    }
}
```

Prikaz programa za konvertovanje decimalne vrijednosti u binarnu u NetBeans IDE



```
1   package com.mycompany.decbin;
2
3   import java.util.Scanner;
4
5   public class DecBin {
6
7       public static void main(String[] args) {
8           Scanner unos = new Scanner( source: System.in);
9           System.out.print( s: "Unesite decimalnu vrijednost: ");
10          int decVrj = unos.nextInt();
11          String binVrj = Integer.toBinaryString( i: decVrj);
12          System.out.println("Binarna vrijednost: " + binVrj);
13      }
14  }
```

Slika 16 - Kod za konvertovanje decimalnog broja u binarni



```
Output - Run (decBin)
Compiling 1 source file to C:\Users\Hasbulla\Docu

--- exec-maven-plugin:3.0.0:exec (default-cli) @
Unesite decimalnu vrijednost: 69
Binarna vrijednost: 1000101
------------------------------------------------
BUILD SUCCESS
------------------------------------------------
```

Slika 17 - Rezultat konvertovanja decimalnog broja u binarni

8.6. Kalkulator

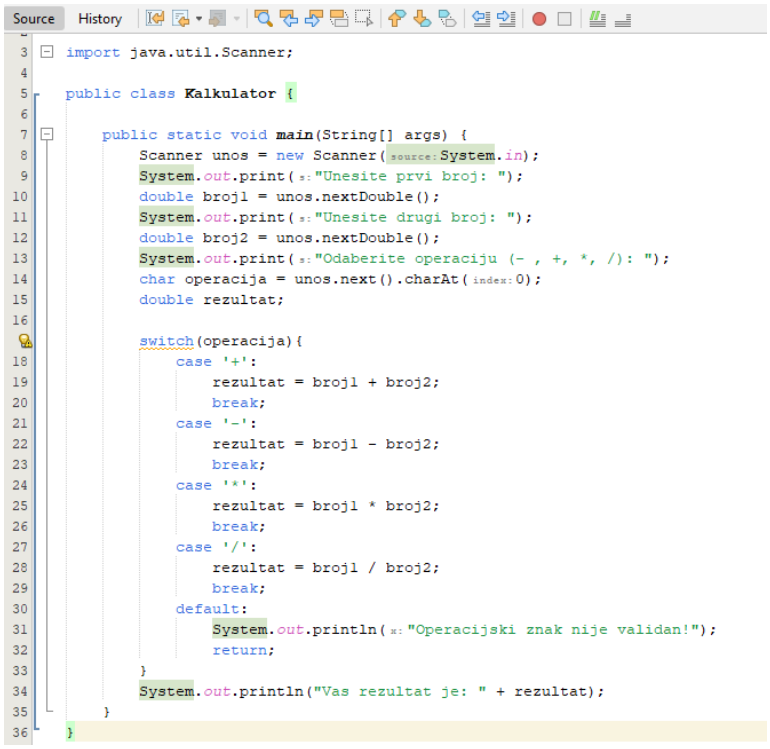
Program koristi klasu Scanner za korisnicki unos i koristi naredbu switch da odredi operaciju koju treba izvrsiti na osnovu operatora koji je unijeo korisnik. Zatim izvrsava operaciju i stampa rezultat na ekranu. U programu se koristi tip podatka Double kako bih uspijesno obradjivali decimalne brojeve, takodjer provjerava da li je operator koji je korisnik unijeo ispravan ili ne.

Tekstualni prikaz Kalkulatora

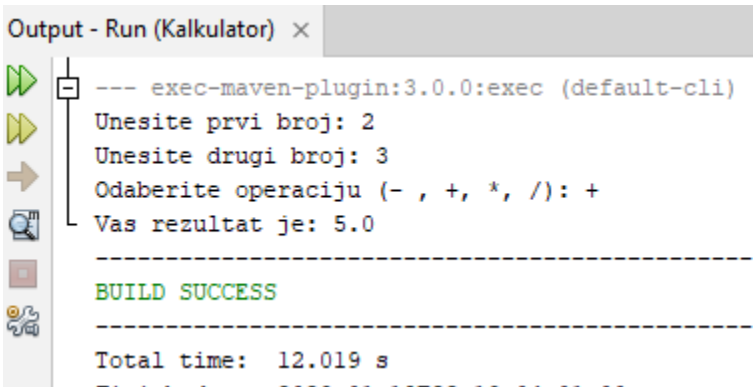
```
import java.util.Scanner;
public class Kalkulator {

    public static void main(String[] args) {
        Scanner unos = new Scanner(System.in);
        System.out.print("Unesite prvi broj: ");
        double broj1 = unos.nextDouble();
        System.out.print("Unesite drugi broj: ");
        double broj2 = unos.nextDouble();
        System.out.print("Odaberite operaciju (- , +, *, /): ");
        char operacija = unos.next().charAt(0);
        double rezultat;

        switch(operacija){
            case '+':
                rezultat = broj1 + broj2;
                break;
            case '-':
                rezultat = broj1 - broj2;
                break;
            case '*':
                rezultat = broj1 * broj2;
```

```
            break;
        case '/':
            rezultat = broj1 / broj2;
            break;
        default:
            System.out.println("Operacijski znak nije validan!");
            return;
    }
    System.out.println("Vas rezultat je: " + rezultat);
  }
}
```

Prikaz Kalkulatora u NetBeans IDE



Slika 18 - Kod kalkulatora



Slika 19 - Rezultat Kalkulatora

8.7. Generator HTML oznaka

Program koristi klasu Scanner kako bih dobio korisnicki unos za naziv oznake i sadrzaj oznake, a zatim ih spaja da generise HTML oznaku i generisana oznaka se zatim prikazuje na ekranu.

Tekstualni prikaz programa za generisanje HTML oznaka

```
import java.util.Scanner;
public class HTMLGen {
    public static void main(String[] args) {
        Scanner unos = new Scanner(System.in);
        System.out.print("Unesite HTML oznaku: ");
        String imeOznake = unos.next();
        System.out.print("Unesite vas sadrzaj: ");
        String sadrzaj = unos.next();

        String htmlOznaka = "<" + imeOznake + ">" + sadrzaj + "</" + imeOznake + ">";
        System.out.println(htmlOznaka);
    }
}
```

Prikaz programa za generisanje HTML oznaka u NetBeans IDE

```java
package com.mycompany.htmlgen;

import java.util.Scanner;

public class HTMLGen {
    public static void main(String[] args) {
        Scanner unos = new Scanner( source: System.in);
        System.out.print( s: "Unesite HTML oznaku: ");
        String imeOznake = unos.next();
        System.out.print( s: "Unesite vas sadrzaj: ");
        String sadrzaj = unos.next();

        String htmlOznaka = "<" + imeOznake + ">" + sadrzaj + "</" + imeOznake + ">";
        System.out.println( x: htmlOznaka);
    }
}
```

Slika 20 - Kod za generisanje HTML tagova

```
Output - Run (HTMLGen)
--- exec-maven-plugin:3.0.0:exec (default-cli
Unesite HTML oznaku: p
Unesite vas sadrzaj: pozdrav
<p>pozdrav</p>
------------------------------------------------
BUILD SUCCESS
------------------------------------------------
Total time:  9.242 s
```

Slika 21 - Rezultat generisanja HTML tagova

8.8. Program za prikaz Specifikacija racunara

Ovaj program koristi klasu 'OperatingSystemMXBean' iz paketa 'com.sun.management' da dobije informacije o operativnom sistemu, njegovom nazivu, verziji, arhitekturi i broju dostupnih procesora. Program takodjer preuzima prosijek opterecenja sistema za poslijednji minut.

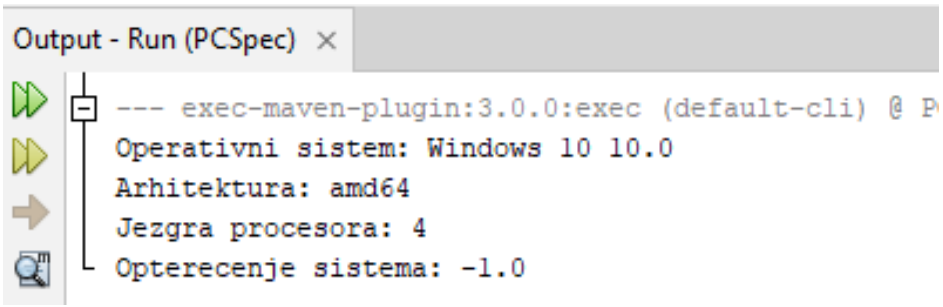
Tekstualni prikaz programa za ispis specifikacija racunara

```
import java.lang.management.ManagementFactory;
import java.lang.management.OperatingSystemMXBean;

public class PCSpec {
    public static void main(String[] args) {
        OperatingSystemMXBean osBean = ManagementFactory.getOperatingSystemMXBean();
        System.out.println("Operativni sistem: " + osBean.getName() + " " + osBean.getVersion());
        System.out.println("Arhitektura: " + osBean.getArch());
        System.out.println("Jezgra procesora: " + osBean.getAvailableProcessors());
        System.out.println("Opterecenje sistema: " + osBean.getSystemLoadAverage());
    }
}
```

Source History

```
1  package com.mycompany.pcspec;
2
3  import java.lang.management.ManagementFactory;
4  import java.lang.management.OperatingSystemMXBean;
5
6  public class PCSpec {
7      public static void main(String[] args) {
8          OperatingSystemMXBean osBean = ManagementFactory.getOperatingSystemMXBean();
9          System.out.println("Operativni sistem: " + osBean.getName() + " " + osBean.getVersion());
10         System.out.println("Arhitektura: " + osBean.getArch());
11         System.out.println("Jezgra procesora: " + osBean.getAvailableProcessors());
12         System.out.println("Opterecenje sistema: " + osBean.getSystemLoadAverage());
13     }
14 }
```

Slika 22 - Kod za ispis specifikacija



Slika 23 - Rezultat ispisa specifikacija

8.9. Super Mario Bros prepreke pomocu For petlje

Ovaj program koristi dve ugnjezdjene For petlje na ponavljanje kroz redove i kolone tj. prepreke. Sposljna petlja kontrolise redove, a unutrasnja kolone. Program koristi IF ELSE naredbu unutar unutrasnje petlje za provjeru trenutnog indeksa reda i kolone, ako je trenutni indeks u prvom ili poslijednjem redu ili koloni, stampace '#', a u obrnutom slucaju stampace ' ' tj. prazan string.

```
public class MarioPrepreke {

    public static void main(String[] args) {
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 4; j++) {
                if (i == 0 || i == 4 || j == 0 || j == 4) {
                    System.out.print("#");
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}
```

```java
package com.mycompany.marioprepreke;

public class MarioPrepreke {

    public static void main(String[] args) {
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 4; j++) {
                if (i == 0 || i == 4 || j == 0 || j == 4) {
                    System.out.print("#");
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}
```

Slika 24 - Super Mario prepreke pomocu For petlje

```
Output - Run (marioPrepreke)
####
#
#
#
####
------------------------------------
BUILD SUCCESS
------------------------------------
```

Slika 25 - Rezultat Super Mario prepreke pomocu For petlje

8.10. Program za izracunavanje povrsine pravougaonika

Ovaj program koristi klasu Scanner da dobije korisnicki unos za sirinu i visinu pravougaonika, zatim te vrijednosti koristi za izracunavanje povrsine pravougaonika mnozenjem visine i sirine. Rezultat se cuva u promjenljivoj i zatim se ispisuje na ekranu.

```
import java.util.Scanner;
public class PovrsinaPrav {

    public static void main(String[] args) {
        Scanner unos = new Scanner(System.in);
        System.out.print("Unesite sirinu pravougaonika: ");
        double sirina = unos.nextDouble();
        System.out.print("Unesite visinu pravougaonika: ");
        double visina = unos.nextDouble();
        double povrsina = sirina * visina;
        System.out.println("Povrsina trougla je: " + povrsina);
    }
}
```

```java
package com.mycompany.povrsinaprav;

import java.util.Scanner;

public class PovrsinaPrav {

    public static void main(String[] args) {
        Scanner unos = new Scanner( source: System.in);
        System.out.print( s: "Unesite sirinu pravougaonika: ");
        double sirina = unos.nextDouble();
        System.out.print( s: "Unesite visinu pravougaonika: ");
        double visina = unos.nextDouble();
        double povrsina = sirina * visina;
        System.out.println("Povrsina trougla je: " + povrsina);
    }
}
```

Slika 26 - Kod za izracunavanje povrsine pravougaonika

```
Output - Run (povrsinaPrav)
Changes detected - recompiling the m
Compiling 1 source file to C:\Users\

--- exec-maven-plugin:3.0.0:exec (de
Unesite sirinu pravougaonika: 10
Unesite visinu pravougaonika: 4
Povrsina trougla je: 40.0
------------------------------------
```

Slika 27 - Rezultat programa za izracunavanje povrsine pravougaonika

ZAKLJUCAK

Zahtijevi za kvalitetom, stabilnoscu i kvalitetnim radom su sve cesci i programeri moraju na takve zahtijeve odgovarati kvalitetnim odabirom alatki koji mu nece stajati na putu ka kreiranju tih zahtijeva.

Java predstavlja jednu od tih alatki koju programer moze da ukljuci u svoj razvojni proces. Kvalitet Java koda je integrisan u sam dizajn jezika i sa time se ostvaruje optimalan rad. Sigurnost je danas vaznije nego ikad prije, ali sa razvojim procesom jezika koji se temelji na OpenSource principima, to se veoma lako postize.

Java je programski jezik koji je tu sa nama duzi niz godina i koji ce se nastaviti razvijati i imace veliki uticaj na buduce programere u svakom obliku

POPIS SLIKA

CITATNI IZVORI

[1] A. B. Downey, Think Java, O'REILLY, 2012.

[2] W3Schools, »W3Schools,« 8 Jan 2023. [Mrežno]. Available: https://www.w3schools.com/java/default.asp.