Објектно орјентисано програмирање

Владимир Филиповић
vladaf@matf.bg.ac.rs
Александар Картељ
kartelj@matf.bg.ac.rs

Сложене конструкције програмског језика Јава

Владимир Филиповић
vladaf@matf.bg.ac.rs
Александар Картељ
kartelj@matf.bg.ac.rs

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

Променљиве

- **Променљива**- локација у меморији за запис неке вредности или референца на објекат.
- У Јави постоје 3 врсте променљивих:
  1. инстанцне (променљиве примерка односно објекта)
  2. класне
  3. и локалне.
- Сваку променљиву карактеришу: **име**, **тип** и **вредност**:
  1. име променљиве је идентификатор.
  2. тип променљиве је један од набројаних типова.
  3. променљива или садржи вредност или је референца на објекат.
- Ако променљива садржи вредност, онда је та вредност је литерал примитивног типа.

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

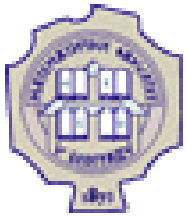
Променљиве (2)

- Свака променљива мора бити декларисана.
- Приликом декларисања одређује се тип променљиве.
- Може се доделити и почетна вредност (иницијализација):
  - Локалне променљиве морају добити почетну вредност пре коришћења (иначе се јавља грешка при превођењу).
  - Променљиве примерка и класне променљиве не морају експлицитно добити почетну вредност пре коришћења.
- Ако променљиве примерка нису експлицитном наредбом добиле почетну вредност, подразумеване вредности су:

  null - референца,
  0 - нумеричка,
  '\0' - знаковна,
  false - логичка

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs



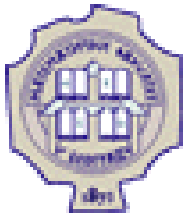
Променљиве (3)

Примери:

```
// Deklaracija jedne promenljive
int brojGodina;
String mojaNiska;
Knjiga x;

// Vise promenljivih istog tipa
float x, y, tezina;
Sting prva, druga, tvojaNiska;

// Sa inicijalizacijom
int i, k, n=32;
boolean ind = false;
String ime = "Dusan";
float a= 3.4f, b=5.8f, y=0.2f;
```

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs



Наредбе

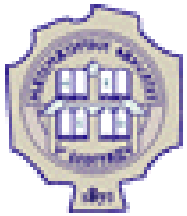
- Искористићемо Бекусову нотацију да дефинишемо наредбу:

```
<naredba> ::= <blok>|<obeležena naredba>
              |<prazna naredba>|<izraz-naredba>
              |<osnovna naredba dodele>|<naredba if>
              |<naredba switch>|<naredba ponavljanja>
              |<naredba break>|<naredba continue>
              |<naredba return>|<naredba throw>
              |<naredba try>|<naredba synchronized>
```

- Наредба је конструкција помоћу које се контролише ток програма и одвијање операција у Јави.
- Наредбе се завршавају знаком ; (тачком-запетом).

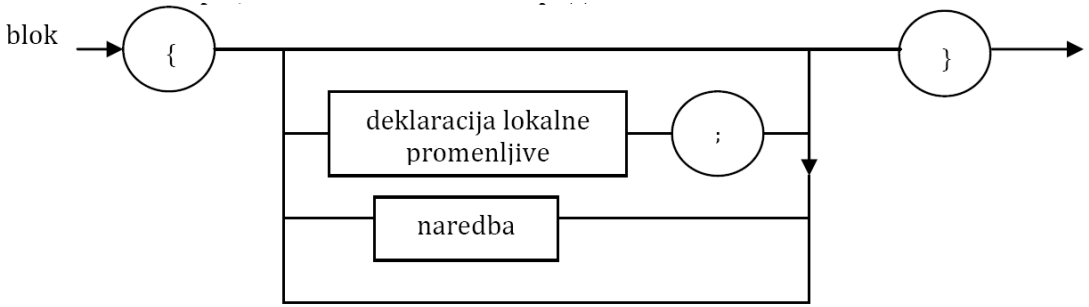
Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs



Блок

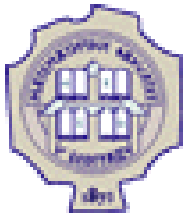
- Блок је секвенца од нула, једне или више наредби или декларација локалних променљивих ограђених витичастим заградама:



- Уочавамо да је претходна дефиниција блока рекурзивна:
  1. блок је дефинисан преко наредбе,
  2. а већ смо видели да наредба може бити блок.
- Тела класа, метода итд. су блокови.

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs



Блок (2)

- Блок може садржати друге блокове и било које друге наредбе које се извршавају једна за другом док се не наиђе на наредбу за промену тока управљања.
- У блоку могу бити декларисане променљиве.
- Оне су локалне за блок и видљиве су (могу се користити) само од места декларисања до краја блока.
- Локална променљива не може бити коришћена уколико јој није додељена почетна вредност.
- Блок не може садржавати више променљивих са истим именом.

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

Блок (3)

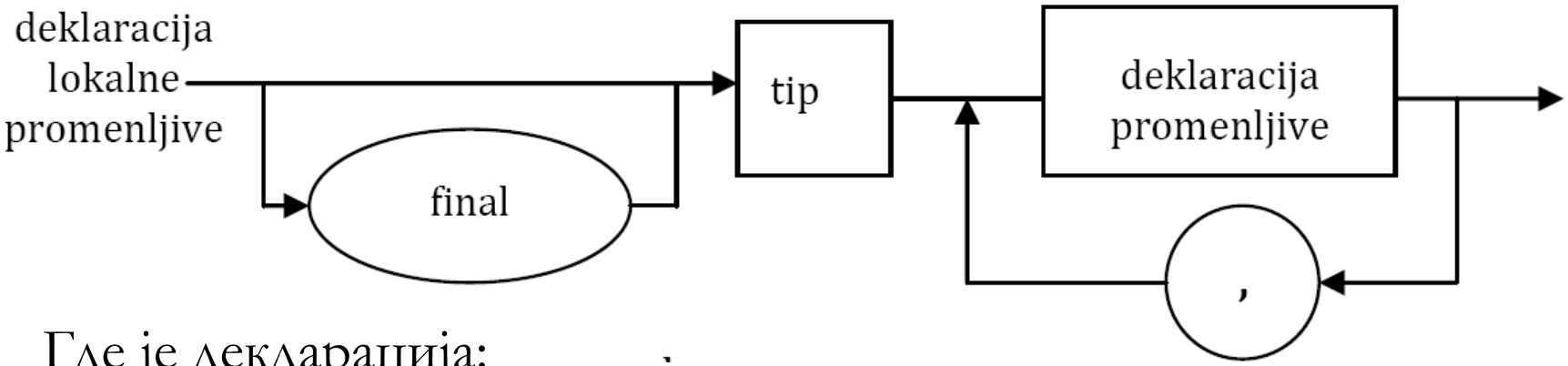
```
public class Test {
        // telo klase je blok
        public static void main(String[] args) {
                //telo metode je takodje blok
                System.out.println("Zdravo svete");
                {
                        //primer suvisnog bloka koji nema nikakvu namenu
                        int x; x=5;
                        if(x<3){
                                // blok za if granu
                                int y=x++;
                                System.out.println(y);
                        }
                        System.out.println(x);
                        //System.out.println(y);
                        // nemoguc ispis vrednosti y, jer je y
                        // deklarisana u if bloku i ovde se ne vidi
                }
        }
}
```

Математички факултет

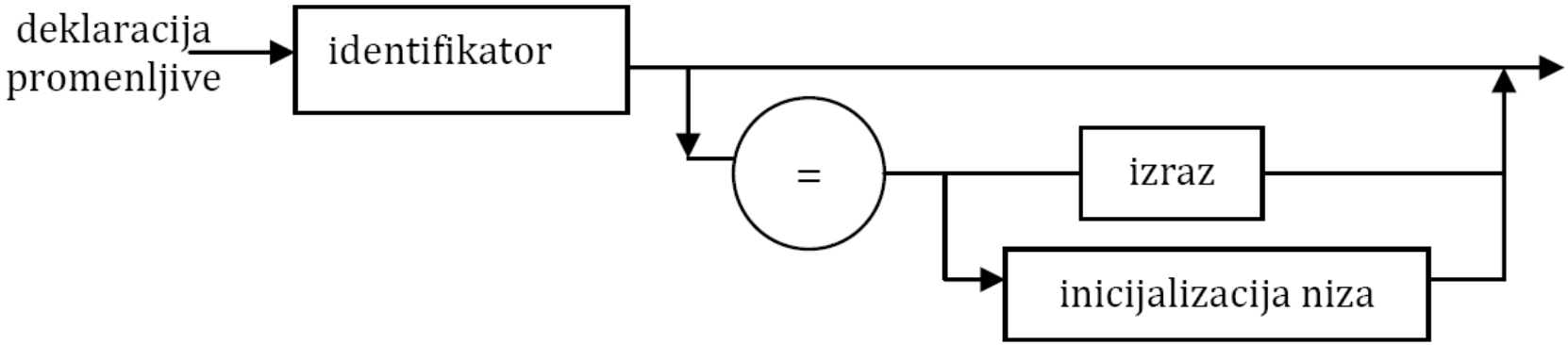
{vladaf, kartelj}@matf.bg.ac.rs

Наредба декларације

- Синтакса декларације локалне променљиве је следећа:



- Где је декларација:



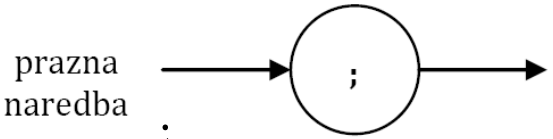
- Металингвистичку променљиву <inicijalizacija niza> ћемо дефинисати када будемо радили са низовима.

Математички факултет

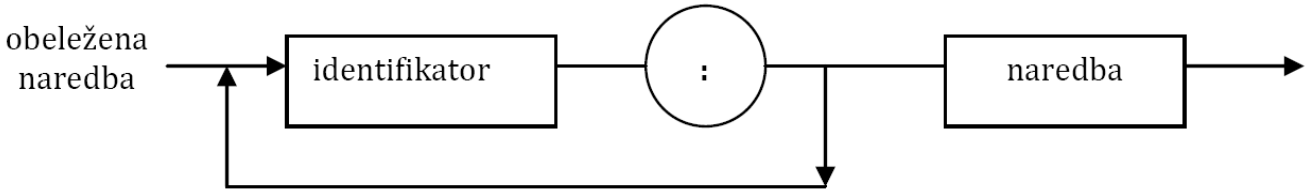
{vladaf, kartelj}@matf.bg.ac.rs

Празна и обележена наредба

- Празна наредба је наредба без дејства.
- Користи се у оним деловима програма где нема никаквих акција. Њена синтакса се може овако изразити:



- Наредба у програмском језику може имати једно или више обележја (лабела) и онда се назива обележеном наредбом:



- Обележена се користи се у комбинацији са наредбама break и continue за безусловни пренос управљања на одређено место у програму.

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

Обележена наредба

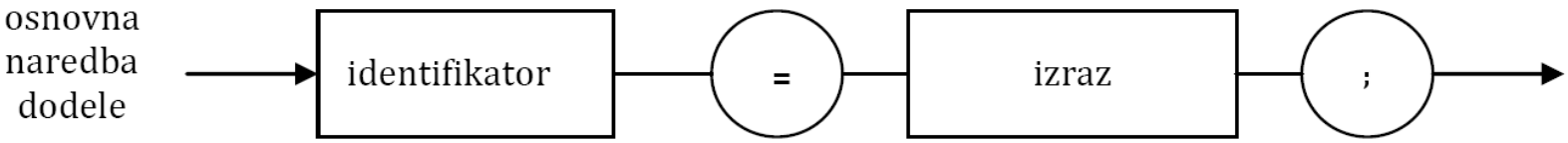
```
public class PrekidSaObelezjem {

        public static void main(String[] args) {
                int[][] niz = { { 32, 87, 3, 589 },
                                { 12, 1076, 2000, 8 },
                                { 622, 127, 77, 955 } };
                int traziSe = 12;
                boolean pronadjen = false;

                obelezje1: //obelezje
                for (int i = 0; i < niz.length; i++) {
                        for (int j = 0; j < niz[i].length; j++) {
                                if (niz[i][j] == traziSe) {
                                        pronadjen = true;
                                        break obelezje1;
                                        //da je samo break, bez obelezja,
                                        //onda bi se iskocilo samo iz
                                        //unutrasnje petlje
                                }
                        }
                }
                System.out.println("Pronadjen "+pronadjen);
        }
}
```

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

Наредба доделе

- Додељивање вредности променљивој у Јави може да се изврши на више начина, али главни начин је **основна наредба доделе**.
- Она омогућава да се променљивима доделе вредности до којих се дошло након одређене обраде.



- У синтаксној дефиницији основне наредбе доделе појављује се појам израза.

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

Изрази

- Изрази у Јави се користе да донесу, израчунају и сместе неку вредност.
- Приоритет оператора у изразу одређује редослед израчунавања вредности.
- У један израз могу бити укључени: операнди, оператори и сепаратори.
- Операнд у изразу може да буде: константа, текућа вредност променљиве, резултат позива метода и др.

```
<izraz> ::= <primarni izraz>| < izraz dodele>
        | <aritmetički izraz> | <relacioni izraz>
        | <logički izraz>| <izrazi sa operatorima po bitovima>
        | <uslovni izraz>| <instancni izraz>
        | <izraz kastovanja>| <unarni izraz>
```

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

Изрази (2)

- Примарни израз је једна од најкомплекснијих компоненти појма израз. Синтаксу примарног израза дефинишемо овако:

```
<primarni izraz>        ::=<primarni prefiks>{<primarni sufiks>}
<primarni prefiks>      ::= <literal> | <ime>
                        | this | super.<identifikator>
                        | (<izraz>) | <alokacioni izraz>
<primarni sufiks>       ::= .<identifikator> | .this
                        | .class | .<alokacioni izraz>
                        | [<izraz>] | <argumenti>
```

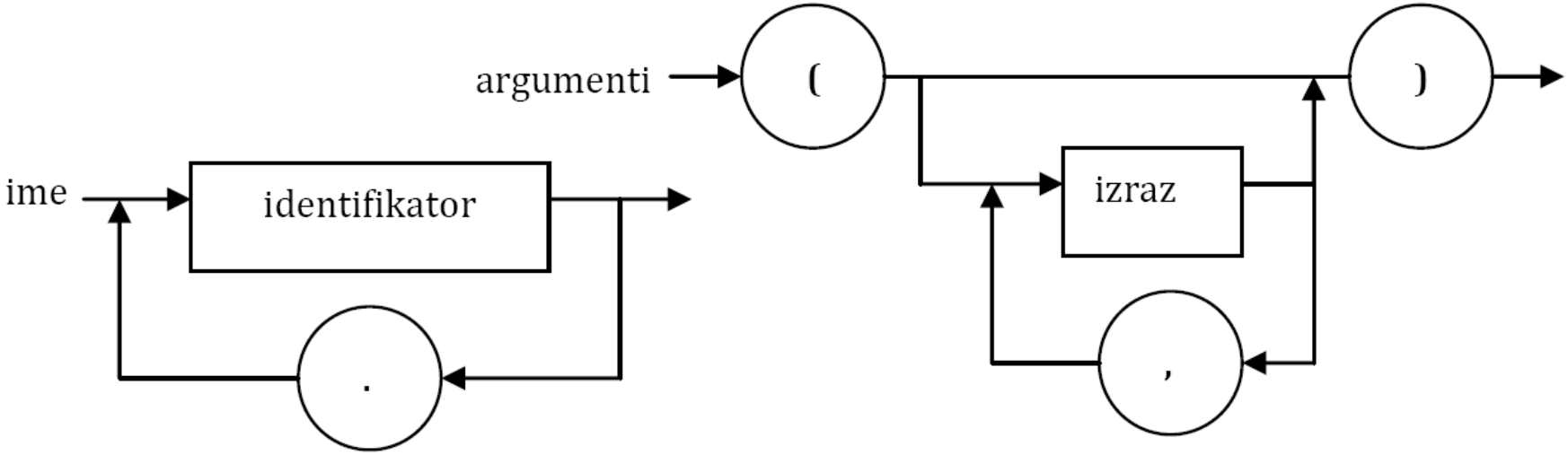
- Из наведене дефиниције видимо да израз може бити литерал, име променљиве, кључна реч this, али и израз између мале отворене и затворене заграде.
- Дакле, појам израза се дефинише преко самог себе, што значи да је претходна дефиниција израза рекурзивна.

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

Изрази (3)

- Метапроменљиве <ime> и <argumenti> можемо описати помоћу синтаксних дијаграма на следећи начин:



Примери неких израза:

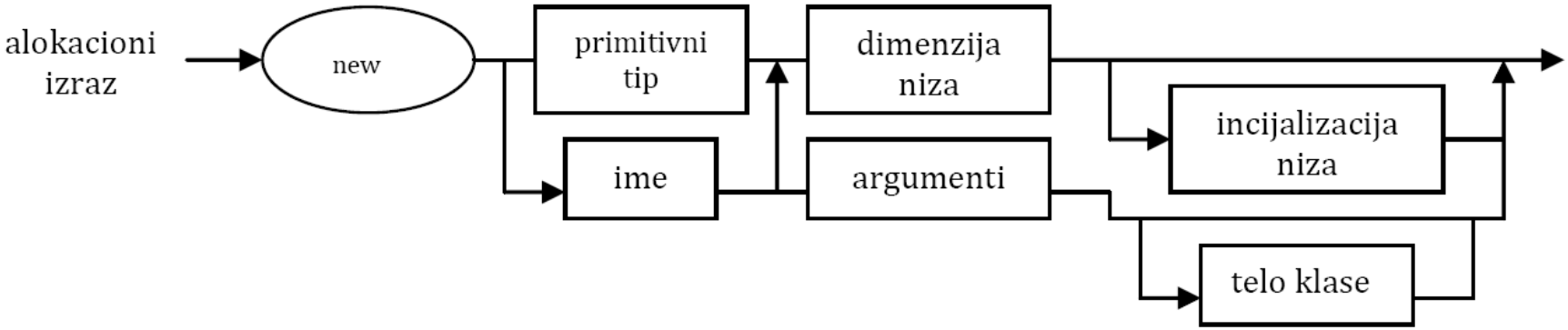
```
args.length
funk(x,y)
pera.mika.zika(a,b)
stampaj(a,b,c+funk(b,a))
Math.random()
System.out.print(niz)
```

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

Изрази (4)

- Синтаксу алокационог израза описујемо помоћу синтаксног дијаграма:



- У наведеном синтаксном дијаграму појављују се метапроменљиве <dimenzija niza> и <inicijalizacija niza>. Ове метапроменљиве ћемо дефинисати када будемо радили са низовима.

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

Изрази (5)

- Ослањајући се на раније дефинисане операторе, можемо дефинисати следеће типове израза:

```
<izraz dodele>              ::= <izraz> <operator dodele> <izraz>
<aritmetički izraz>         ::= <izraz> <aritmetički operator> <izraz>
<relacioni izraz>           ::= <izraz> <relacioni operator> <izraz>
<logički izraz>             ::= <izraz> <logički operator> <izraz>
<izrazi sa operatorima
po bitovima>                ::= <izraz> <operator po bitovima > <izraz>
<uslovni izraz>             ::= <izraz> ? <izraz> : <izraz>
<instancni izraz>           ::= <izraz> instanceOf <tip>
<izraz kastovanja>          ::= (<tip> ) <izraz>
```

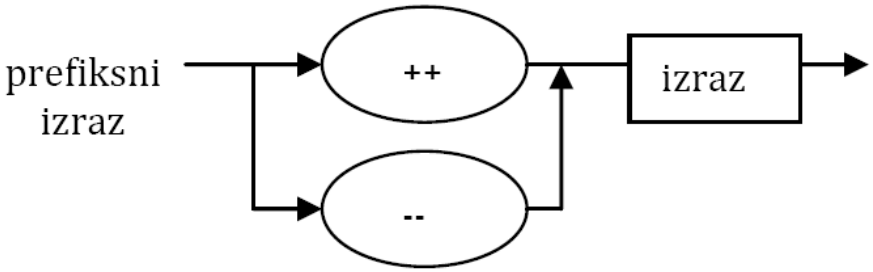
- У изразу доделе као оператор доделе може да се појави и =. Дакле, основна наредба доделе представља специјални случај израза доделе.

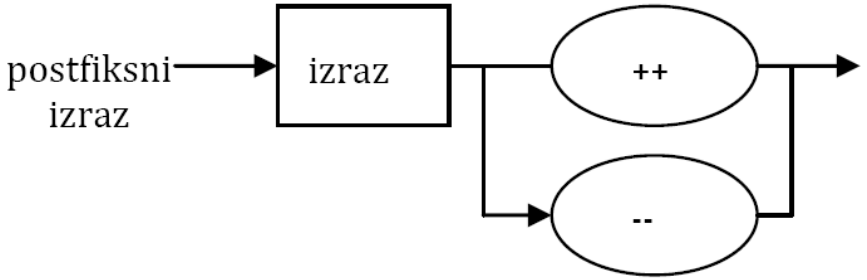
Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

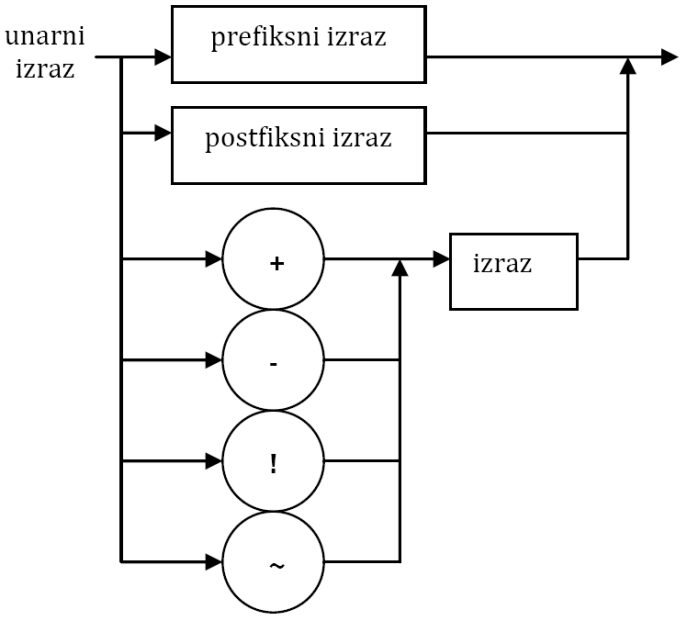
Изрази (6)

- Да би се дефинисао унарни израз, потребно је прво дефинисати префиксни израз и постфиксни израз:





- Сада је унарни израз дефинисан следећим ситнакним дијаграмом:



Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

Изрази (7)

- У једном изразу може да се појави већи број оператора па се намеће питање који је коректан редослед њихове примене?
- Сваком оператору придружен је приоритет.
- Шта ако више оператора имају исти приоритет?
- Тада се примењује карактеристика асоцијативности.
- Оператор може бити:
  - лево- асоцијативан (аритметички оператори),
  - десно-асоцијативан (оператор доделе),
  - или неасоцијативан (релациони оператори, нема смисла израз a < b < c).

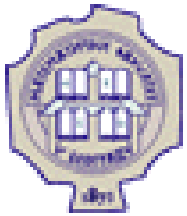
Изрази (8)

- У следећој таблици приказани су оператори са одговарајућим приоритетима и асоцијативностима.

Приоритет	Оператор	Асоцијативност
1	`()`, `[]`	неасоцијативан
2	`new`	неасоцијативан
3	`.`	лево-асоцијативан
4	`++`, `--`	неасоцијативан
5	`-(унарни)`, `+(унарни)`, `!`, `~`, `++`, `--`, *(tip)*	десно-асоцијативан
6	`*`, `/`, `%`	лево-асоцијативан
7	`+`, `-`	лево-асоцијативан
8	`<<`, `>>`, `>>>`	лево-асоцијативан
9	`<`, `>`, `<=`, `>=`, `instanceof`	неасоцијативан
10	`==`, `!=`	лево-асоцијативан
11	`&`	лево-асоцијативан
12	`^`	лево-асоцијативан
13	`|`	лево-асоцијативан
14	`&&`	лево-асоцијативан
15	`||`	лево-асоцијативан
16	`?:`	десно-асоцијативан
17	`=`, `*=`, `/=`, `%=`, `-=`, `<<=`, `>>=`, `>>>=`, `&=`, `^=`, `|=`	десно-асоцијативан

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs



Компилационе јединице

У састав компилационе јединице могу ући:

- package директива (пакетне наредбе),
- import директива (наредбе увоза) и
- дефиниције класа и/или интерфејса и/или енумерација .

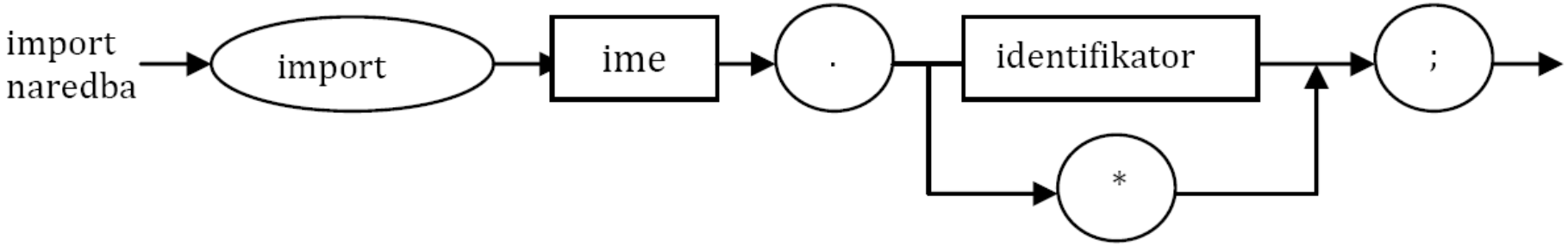
- Пакетна наредба служи за одређивање да се дата класа/интерфејс (која се дефинише) налази у датом пакету.
- Наредба увоза омогућује да се, приликом позива тј. слања поруке, уместо пуног имена класе/интерфејса (имена које обухвата име пакета).

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs

Компилационе јединице (2)

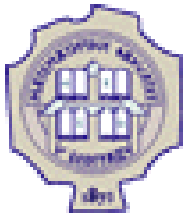
- Синтакса наредбе import се описује следећим дијаграмом:



- Наредба увоза import представља само помоћ програмеру ради скраћивања нотације.
- На пример, класа за читање података са улаза је једнозначно одређена пуним именом java.util.Scanner.
- Наредба import омогућује да, тој класи више не мора да се приступа коришћењем пуног имена те класе, већ коришћењем њеног скраћеног имена: Scanner.

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs



Компилационе јединице (3)

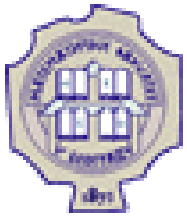
увоз статичких метода.

Пример 1:

```
package oop.apleti;
import java.applet.*;
import java.awt.*;

public class Crta1 extends Applet{
        public void paint (Graphics g){
                g.drawLine(50, 50, 100, 100);
                g.fillRect(100, 100, 50,50);
                g.drawRoundRect(10, 10, 100, 80, 30, 30);
                g.fill3DRect(200,200, 50, 50, true);
                g.draw3DRect(1,200, 50, 50, false);
        }
}
```

Математички факултет

{vladaf, kartelj}@matf.bg.ac.rs



Захвалница

Велики део материјала који је укључен у ову презентацију је преузет из презентације коју је раније (у време када је он држао курс Објектно орјентисано програмирање) направио проф. др Душан Тошић.

Хвала проф. Тошићу што се сагласио са укључивањем тог материјала у садашњу презентацији, као и на помоћи коју ми је пружио током конципцирања и реализације курса.