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

- Наредба је конструкција помоћу које се контролише ток програма и одвијање операција у Јави.
- Наредбе се завршавају знаком ; (тачком-запетом).
- Разликујемо следеће типове наредби:
  - Наредба декларације
  - Празна наредба
  - Обележена наредба
  - Наредба доделе
  - Наредба израза
  - Наредба блока
  - **switch**, **break**, **return**, ...

Математички факултет

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

Наредба декларације

- Синтакса декларације локалне променљиве је следећа:
  - Најпре се наводи тип променљиве
    - Додатно, може се навести и кључна реч **final**, чије значење ћемо касније увести.
  - Потом следи један или више идентификатора раздвојених зарезима, који представљају називе променљивих
  - Специјално, након назива променљиве може уследити и израз којим се иницијализује вредност променвљиве

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.
  - Нпр. у for циклусу, када немамо иницијализацију или услов
  - Такође се може користити унутар тела циклуса, за реализацију „чекања“:
    - while(nekiUslov){;} или само while(nekiUslov);
- 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

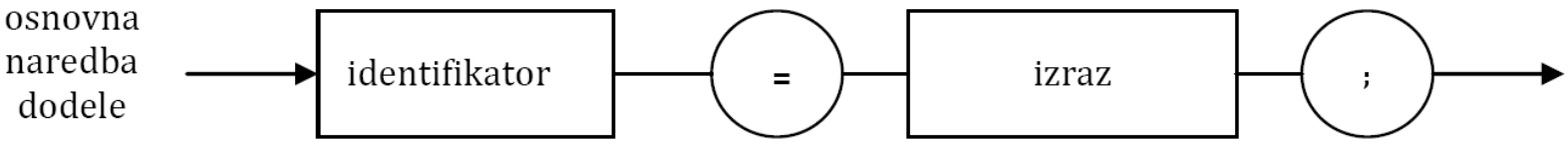
```
public class PrekidSaObelezjem {

        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 одређује редослед израчунавања.
- 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.
- Пример 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 (2)

- Ослањајући се на раније дефинисане операторе, можемо дефинисати следеће типове израза:
  - Израз доделе
  - Аритметички израз
  - Релациони израз
  - Логички израз
  - Израз са битовским операторима
  - Условни израз
  - Инстанцни израз
  - Кастовање
  - Или комбинација претходних

Математички факултет

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

Изрази (3)

- 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).

Математички факултет

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

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

Блок

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

- Уочава се 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

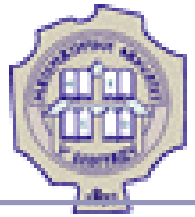
{vladaf, kartelj}@matf.bg.ac.rs

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

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)

- 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);
        }
}
```

Низови у програмском језику Јава

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

Математички факултет

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

Низови у Јави

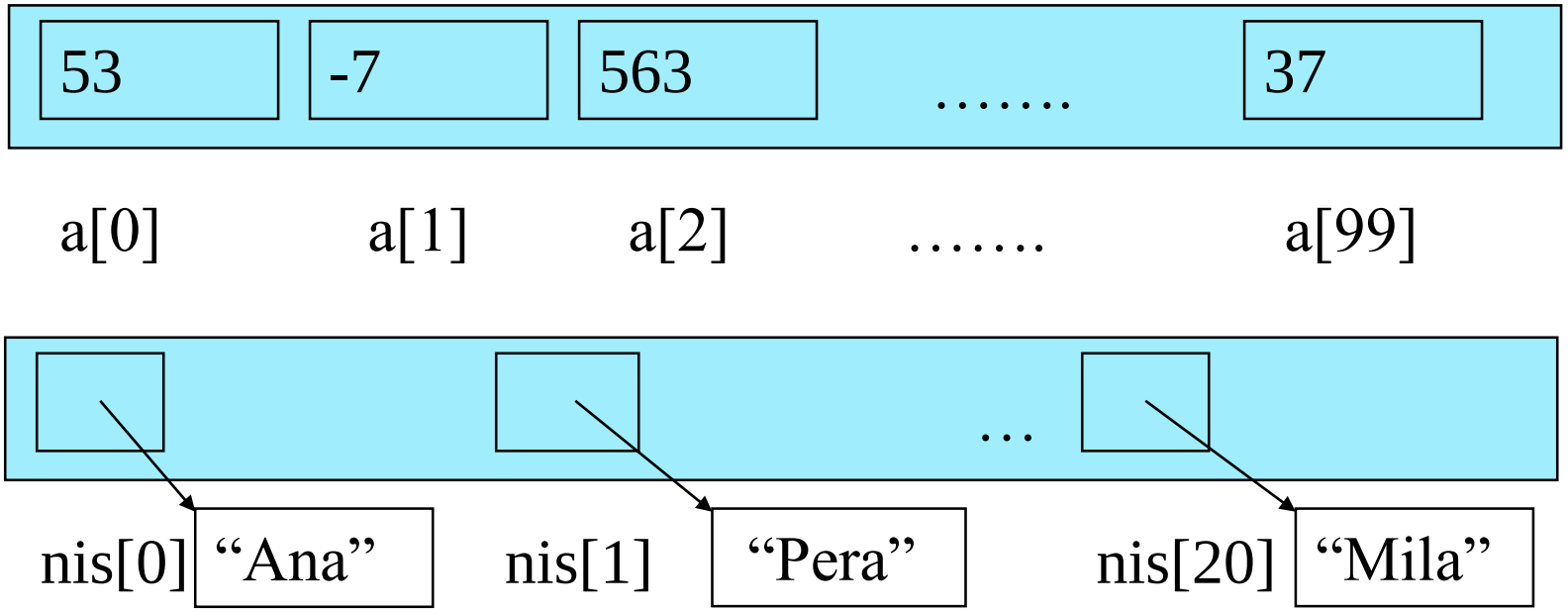
- Низ се дефинише као група променљивих истог типа које се појављују под заједничким именом.
- Низовни тип података у Јави има следећа својства:
  1. садржи линеарно уређен, унапред познат, број чланова,
  2. сви чланови су истог типа и имају заједничко име; чланови могу бити примитивног или објектног типа,
  3. сваком члану приступа се помоћу заједничког имена низа и индекса члана,
  4. сви индекси су целобројног типа,
  5. сви чланови низа се третирају као посебне променљиве (називају се и индексним променљивим).

Математички факултет

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

Једнодимензионални низ

- Једнодимензионални за приступ елементима користи тачно један индекс.



Математички факултет

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

Једнодимензионални низ (2)

Декларисање низова (постоје два начина):

Пример.

```
int pozicija[];
String niska[];
Knjiga naslov[];

// Iste deklaracije na drugi način
int [] pozicija;
String [] niska;
Knjiga [] naslov;
```

Уочава се разлика између ове две врсте декларација:

```
int [] a, b, zzz; // sve 3 promenljive a, b i zzz se deklarisu kao
                  //nizovne
int a, b, zzz[]; // samo zzz se deklarise kao nizovna promenljiva
```

Математички факултет

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

Једнодимензионални низ (3)

Креирање низова

- Постоје два начина:

  (1) помоћу оператора new,

  (2) набрајањем чланова

Пример.

```
Point [] tacke = new Point[20]; // prvi nacin
 int [] brojac = new int[100]; // prvi nacin

 // Sledi drugi nacin kreiranja
 String godDoba[], s="mika";
 godDoba ={"prolece", s, "leto", "jesen", "zima"};
 int celi[] = { 23, 46, -123, 17, 36, -12};
```

Једнодимензионални низ (4)

Приступ члановима низа

- Нумерисање чланова низа увек почиње од 0.
- Члановима низа се приступа помоћу: *ImeNiza[indeks]*
- *ImeNiza* – име које се појављује у декларацији
- *indeks* – целобројни литерал или израз

Пример.

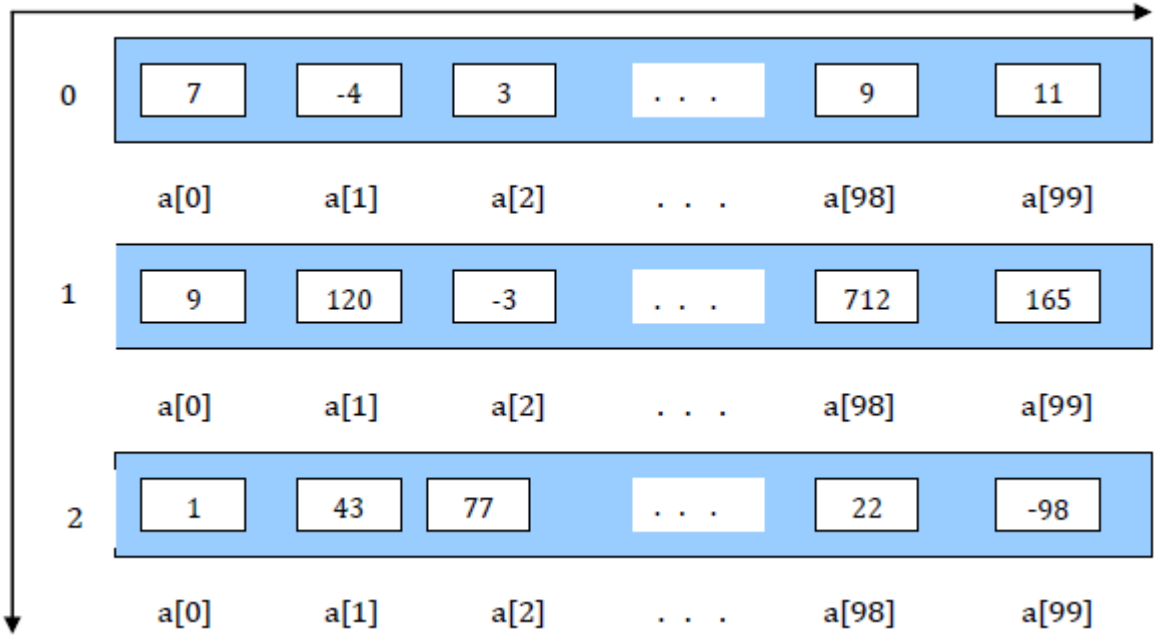
```
int [] brojac = new int[100];
String [] godDoba={“prolece”, “leto”, “jesen”, “zima”};
…
brojac[50]=2; // Ne bi valjalo: brojac[100] = 3;
a=brojac[i+j];
x = godDoba[3];
```

Математички факултет

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

Дводимензионални низ

- Дводимензионални низ чине елементи истог типа и сваком од њих приступа се помоћу имена низа и два индекса.
- Више једнодимензионалних низова се могу поређати један испод другог као на следећој слици:

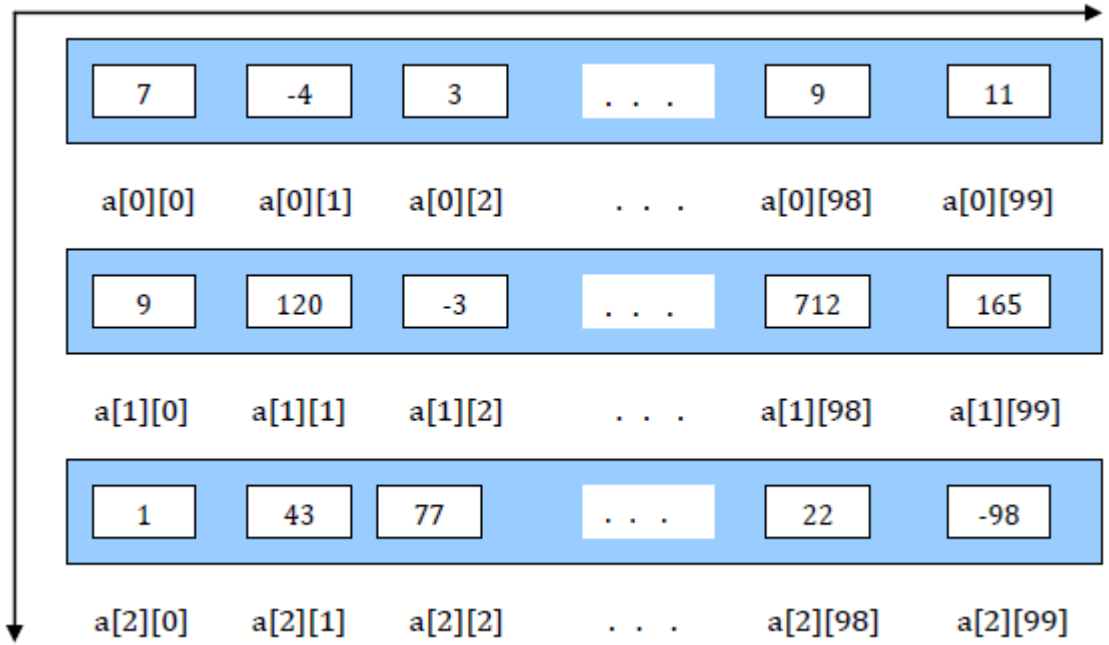


Математички факултет

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

Дводимензионални низ (2)

- Индексирање употребом два индекса.



- Дводимензионални низ представља једнодимензионални низ чији елементи су једнодимензионални низови.

Математички факултет

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

Дводимензионални низ (3)

- Декларацију имена низа можемо да извршимо на два начина:

```
int m, a[][]; // prvi nacin
int[][] c, b; //drugi nacin
```

- Након декларације низовне променљиве, креирамо низовни објекат, тј. вршимо резервисање меморијског простора за смештање чланова низа:

```
a = new int[50][100];
```

Овом наредбом је резервисано 5000 целобројних локација у меморији.

- Декларацију низовне променљиве и резервисање меморијског простора могли смо извршити једном наредбом, тј. могли смо писати:

```
int a[ ][ ] = new int[50][100];
```

Математички факултет

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

Вишедимензионални низ

- Дводимензионални низови представљају један, најчешће коришћен, случај вишедимензионалних низова.
- За низове чији је број димензија већи од два, принцип рада је потпуно исти као и са дводимензионалним низовима.
- Дакле, као што су формирани дводимензионални низови, на исти начин се могу формирати тродимензионални, четвородимензионални итд.
- Већ смо уочили да оперисање са матрицама можемо свести на оперисање са једнодимензионалним низовима, то важи и за све вишедимензионалне низове.

Управљачке структуре у програмском језику Јава

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

Математички факултет

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

Наредбе гранања

- Наредбе гранања омогућавају да се одабере извршавање једног дела програма у зависности од испуњења одређених услова.
- Користе се још и називи условне наредбе, односно наредбе одабирања.
- Постоје две наредбе гранања у језику Јава:
  1. наредба if
  2. и наредба switch

Математички факултет

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

Наредба if

- За условно извршење наредбе или за избор између извршења две наредбе обично се користи наредба if.
- Наредба if појављује се у два облика:
  1. if (<izraz>) <naredba> - непотпуни облик
  2. if (<izraz>) <naredba> else <naredba> - потпуни облик.

Математички факултет

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

Наредба if (2)

- Наредба if у непотпуном облику извршава се на следећи начин:
  1. израчунава се вредност израза;
  2. ако је вредност израза истинита, извршава се наредба која следи иза израза;
  3. ако је вредност израза неистинита, извршава се прва наредба која следи иза наредбе if.

```
if (U)
        Nar1;
Nar2;
```

Математички факултет

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

Наредба if (3)

- Наредба if у потпуном облику се извршава на следећи начин:
  1. израчунава се вредност израза;
  2. ако је вредност израза истинита, извршава се наредба која следи иза израза;
  3. у супротном, извршава се наредба иза резервисане речи else.

```
if (U)
        Nar1;
else
        Nar2;
```

Математички факултет

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

Наредба if (4)

- У случају када се појављује наредба if унутар наредбе if, са једном придруженом наредбом else, поставља се питање да ли је наредба else придружена првој наредби if или другој наредби if.
- Другачије формулисано, ако имамо конструкцију:

```
if (U1) if (U2) Nar1 else Nar2
```

- да ли ће се Nar2 извршити ако је:

(а) логички израз U1 тачан, а логички израз U2 нетачан, или

(б) ако је логички израз U1 нетачан?

Математички факултет

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

Наредба if (5)

Пример.

```
if (a < b ) System.out.println("a je manje od b");
if (tezina>200) {
   tezina = 200;
   ind = true;
}else
   ind =false;
```

Уместо наредбе if у неким ситуацијама може се употребити условни оператор ?:

```
(uslov) ? then-izraz : else-izraz
```

Пример.

```
min = (x<y) ? x : y;
```

Математички факултет

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

Наредба switch

- Наредба switch служи за избор једне наредбе из скупа од неколико могућих, а на основу вредности неког израза.

```
switch (doba) {
        case 'p':
                System.out.println("Setnja");
                break;
        case 'l':
                System.out.println("Kupanje");
                break;
        case 'j':
                System.out.printl("Branje grozdja");
                break;
        case 'z':
                System.out.println("Skijanje");
                break;
        default:
                System.out.println("Greska");
}
```

Математички факултет

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

Наредба switch (2)

- Резервисана реч default може се појавити само једнапут, што није посебно назначено.
- Израз иза резервисане речи switch назива се селектор и мора бити типа:
  1. byte, char, short или int;
  2. Енумерисани тип (од верзије 5);
  3. String, као и типа неке од класа-омотача простих типова тј.: Character, Byte, Short или Integer (од верзије 7).
- Вредности (ознаке) које се појављују иза наредбе case морају бити истог типа као и селектор и међусобно различите.

Математички факултет

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

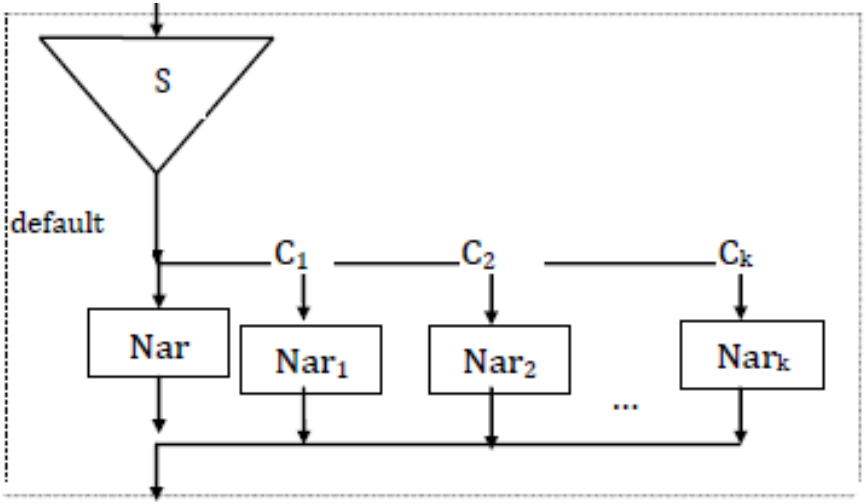
Наредба switch (3)

- Наредба switch се извршава на следећи начин:
  1. Израчунава се вредност селектора и ако није типа int, вредност се преводи у тип int.
  2. Упоређује се израчуната вредност са вредностима (ознакама) иза наредби case.
  3. Ако се деси поклапање извршава се поклапајућа case наредба.
  4. Ако се не поклапа ни са једном case наредбом ом онда се извршава default наредба.
  5. Ако нема наредбе која има ознаку default, наредба после наредбе switch биће следећа која се извршава.

Наредба switch (4)

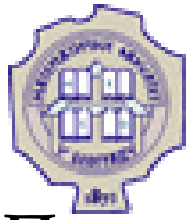
- У ускоj вези са наредбом switch је наредба break.
- Следећим дијаграмом описана је синтакса наредбе break:
- Ако наредба break не садржи идентификатор, управљање се преноси на прву следећу наредбу иза наредбе у којој се налази.

```
switch  (S)
{
        case C1:
                Nar1;
                break;
        case C2:
                Nar2;
                break;
         …
        case Ck:
                Nark;
                break;
        default:
                Nar;
                break;
}
```



Математички факултет

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



Наредба switch (5)

Пример. Уместо:

```
if (oper = ='+')
        sabrati(arg1, arg2);
else if (oper == '-')
        oduzeti(arg1, arg2);
else if (oper == '*')
        pomnoziti(arg1,arg2);
.....
```

прегледније је користити:

```
switch(oper) {
        case '+':
                sabrati(arg1, arg2);;
                break;
        case '-':
                oduzeti(arg1, arg2);
                break;
        .....
        default:
                podrazumevanaOperacija(arg1,arg2); // opciono
}
```

Математички факултет

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

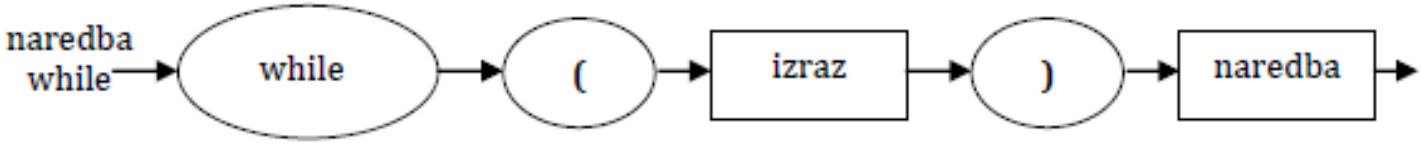
Наредба switch (6)

Пример. Дисјункција више услова може бити задата узастопним case наредбама без наредне break.

```
switch (month) {
        case 1: case 3: case 5:
        case 7: case 8: case 10: case 12:
                numDays = 31;
                break;
        case 4: case 6: case 9: case 11:
                numDays = 30;
                break;
        case 2:
                if (((year % 4 == 0) && !(year % 100 == 0))
                        || (year % 400 == 0))
                        numDays = 29;
                else
                        numDays = 28;
                break;
        default:
        System.out.println("Invalid month.");
        break;
}
```

Наредба while

- Наредба while се још назива наредба циклуса са предусловом.
- У њој се најпре проверава да ли је испуњен услов и ако јесте извршавају се наредбе циклуса.



- Израз у наредби while је логичког типа. Ефекат наредбе while је следећи:
  1. израчунава се вредност израза;
  2. ако је вредност израза истинита,
     извршава се наредба и понављају кораци 1. и 2. ;
  3. ако је вредност израза неистинита,
     завршава се извршавање наредбе while
     и прелази се на прву следећу наредбу иза while.

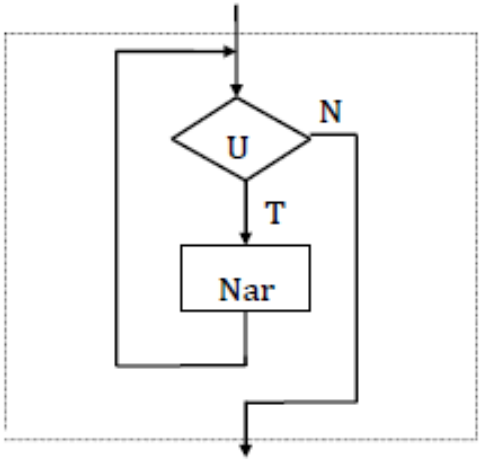
Математички факултет

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

Наредба while (2)

Наредба while се извршава на слелећи начин:

```
while (U)   Nar
```



Пример.

```
{
        …
        double x=5.0;
        while (x>0.1) {
                x=Math.random();
                System.out.println("x="+x);
        }
}
```

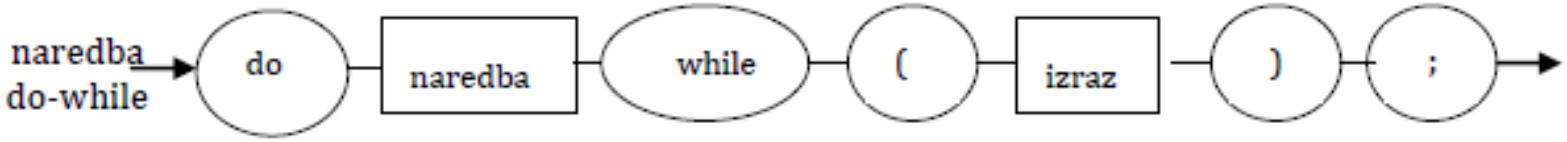
Математички факултет

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

Наредба do-while

- Наредба do-while се још назива наредба циклуса са постусловом.

Синтакса наредбе do-while је следећа:



- Израз је логичког типа.
- Извршавање наредбе do-while реализује се преко следећих корака:
  1. извршава се наредба иза резервисане речи do;
  2. израчунава се вредност израза у заградама;
  3. ако је вредност израза истинита, процес се наставља;
     у супротном, прелази се на прву наредбу иза наредбе do-while.

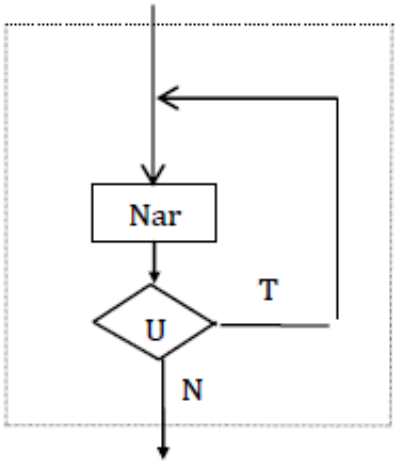
Математички факултет

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

Наредба do while (2)

Наредба do while се извршава на следећи начин:

```
do Nar while (U)
```



Пример.

```
{ …
        do
                x=Math.random();
                System.out.println("x="+x);
        while (x<0.9);
        …
}
```

Математички факултет

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

Наредба for

- Наредба for је моћна наредба за опис циклуса.
- Најчешће се користи за опис циклуса код којих је број понављања наредби унапред познат.
- Наредба for има следеће сегменте:
  1. Део за иницијализацију
     - Обично иницијализација управљачких (бројачких) променљивих
  2. Део за постављање условног израза останка у циклусу
  3. Део за итерацију петље
     - Обично промена вредности бројачке променљиве
  4. Тело циклуса

Математички факултет

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

Наредба for (2)

Наредба for се извршава на следећи начин:

1. Прво се извршава иницијализациони део петље for:
   - Подешава се вредност управљачке променљиве петље for.
   - Иницијализација се извршава само једанпут.
2. Након иницијализације:
   - Израчунава се вредност условног израза (ако постоји) који мора бити логичког типа.
   - Ако израз не постоји, подразумевана вредност је true.
3. У зависности од вредности израза:
   - Ако има вредност true, извршава се тело петље, а затим се извршава итерација петље и поново се иде на корак 2.
   - Уколико израз има вредност false, окончава се извршавање петље.

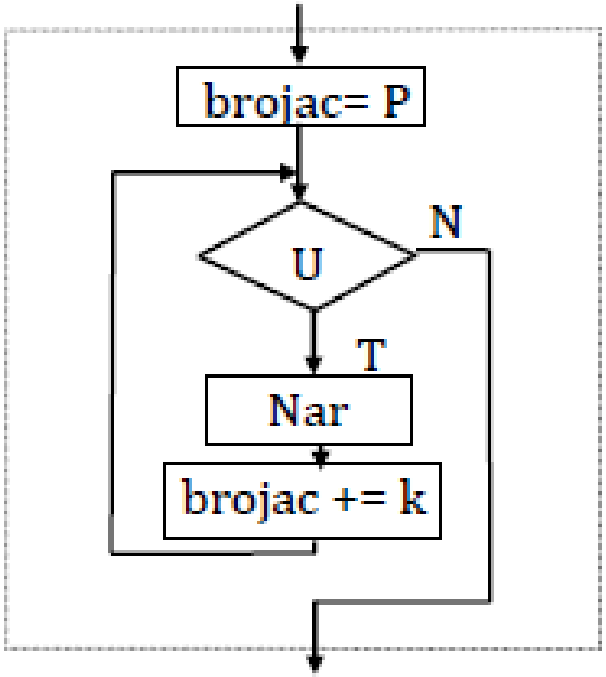
Математички факултет

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

Наредба for (3)

- Извршавање наредбе for показује и следећи дијаграм:

```
for(brojac = P; U; brojac += K)
    Nar
```



Пример.

```
{
        …
        for (int i=1; i<=10; i++)
                System.out.println(”Ana voli programiranje”);
        …
}
```

Математички факултет

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

Колекцијска наредба for

- Колекцијска наредба for служи за пролазак кроз колекцију/низ.
- Притом променљива у наредби for не представља бројач, већ редом узима вредности елемената низа тј. колекције.

Пример.

```
{ ...
        int n = sc.nextInt();
        double a[] = new double[n];
        …
        double najveci=a[0];
        for (double elem: a)
                if ( elem > najveci )
                        najveci = elem;
        ...
}
```

Математички факултет

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

Обележена наредба

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

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

Математички факултет

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

Наредба break

- Наредба break је у уској вези не само са наредбом switch, већ и са наредбама циклуса.

- Поред тога што се наредба break може користити:
  1. за излазак из наредбе switch,
  2. она се може искористити за излазак из петље,
  3. али и за безусловни пренос управљања на обележену наредбу (под одређеним условима).
     Ово представља једну врста “контролисане” goto наредбе.

Математички факултет

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

Наредба break (2)

Пример.

- Да би се прекинуло извршавање наредби унутрашње петље и вратило се у спољашњу, користи се наредба break без обележја

```
for(int i=0; i<100; i++) {
        for(int j=0; j<100; j++) {
                if (uslov)
                        break;
        }
}
```

Математички факултет

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

Наредба break (3)

Пример.

- Да би се прекинуло извршавање наредби обе петље, користи се наредба break са обележјем.

```
...
izlaz:
for(int i=0; i<100; i++) {
        for(int j=0; j<100; j++) {
                if (uslov)
                        break izlaz;
        }
}
```

Математички факултет

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

Наредба continue

- Наредба continue је по синтакси слична наредби break.
- Она се може користити само у наредбама за опис циклуса.
- Нареба continue је корисна када треба да се настави извршавање циклуса, уз прескакање одређеног скупа наредби.
- Помоћу ове наредбе управљање се преноси на израз за проверу услова изласка из петље.

Математички факултет

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

Наредба continue (2)

Пример.

Да би се прескочио препис оних елемената низа који су једнаки нули, користи се наредба continue без обележја.

```
...
br1 = 0;
br2 = 0;
while (br1 < niz.length) {
        if (niz[br1] = = 0){
                br1++;
                continue;
        }
        vektor[br2++]=niz[br1++];
}
```

Математички факултет

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

Наредба continue (3)

Пример.

- Учитавање низа од четири бинарне цифре и спајање учитаних цифара у један стринг, те приказ стринга.
- Наставља се са следећом итерацијом петље обележене обележјем ако цифра није бинарна.

```
...
String s="";
System.out.println("Unesite cetiri binarne cifre ");
vrh:
do {
        d=sc.nextInt();
        if((d!=0)&&(d!=1))
                continue vrh;
        s +=d;
}
while(s.length()<4);
System.out.println("Formiran je binarni string: "+s);
```

Математички факултет

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

Наредба continue (4)

Пример.

- Наставља се са следећом итерацијом спољне (обележене) петље.

```
String text = "Trazimo neki podstring u ovom stringu";
String podstring = "pod";
boolean pronadjen = false;
int max = text.length() - podstring.length();

test:
for (int i = 0; i <= max; i++) {
        int n = podstring.length();
        int j = i;
        int k = 0;
        while (n-- != 0) {
                if (text.charAt(j++) != podstring.charAt(k++)) {
                        continue test;
                }
        }
        pronadjen = true;
        break test;
}
System.out.println(pronadjen ? "Pronadjen" : "Nije pronadjen");
```

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.