



Академија струковних  
студија Шумадија  
одсек у Крагујевцу

# Увод у програмирање

## Презентација 6

---

Академија струковних студија Шумадија

Одсек у Крагујевцу

Студијски програм Информатика

Крагујевац, 2020. година



# Подсетник

## Угнежђене петље

- Могуће је угнездити једну **for** петљу у другу **for** петљу.
- Овај приступ ћемо користи за обраду вишедимензионалних низова.
- Пример: таблица множења бројева од 1 до 9 (укључујући):

```
public static void main(String[] args) {  
    for (int i = 1; i < 10; i++) {  
        for (int j = 1; j < 10; j++) {  
            System.out.print(i * j + "\t");  
        }  
        System.out.println();  
    }  
}
```

The screenshot shows a Java IDE window titled "Test" with a "Run:" button. The output of the program is displayed in a text area, showing a 9x9 multiplication table. The table is printed with tabs between the numbers. Below the table, it says "Process finished with exit code 0".

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

Process finished with exit code 0



# Подсетник

- Када треба да обрадите више објеката **истог типа**, можете их сачувати у **низу**, а затим обрадити заједно као **једну целину**.
- Низ можете сматрати колекцијом елемената истог типа.
- Колекција даје **једно име** за своје елементе.
- Могући број елемената који се чувају утврђује се када се низ креира и **не може** се променити.
- Сачувани елемент **се може изменити** у било ком тренутку.

Index	0	1	2	3	4
Element	10.8	14.3	13.5	12.1	9.7



# Класа Arrays

- Ради лакшег рада са низовима, у Јави се може користити стандардна класа **Arrays** из пакета `java.util`.
- Класа **Arrays** садржи **разне методе** за рад с низовима, које премошћавају јаз између колекција и низова. (о овоме више на предмету ООП)
- Неке од метода класе **Arrays**:
  - `String toString(tip[] мојNиз)` - враћа се низ `мојNиз` у облику "стринга".
  - `tip[] copyOf(tip[] мојNиз, int n)` - враћа се нова копија низа `мојNиз` која се састоји од првих `n` његових елемената.
  - `tip[] copyOfRange(tip[] мојNиз, int i, int j)` - враћа се нова копија низа `мојNиз` од индекса `i` до индекса `j`.
  - `void sort(tip[] мојNиз)` - сортира се низ `мојNиз` у растућем редоследу.



# Klasa Arrays

```
package rs.edu.asss.test;
import java.util.Arrays;
public class Test {
    public static void main(String[] args) {

        int[] niz = new int[10];

        for (int i = 0; i < niz.length; i++) {
            niz[i] = -3 * i;
        }

        System.out.println("Niz sadrži sl. članove: ");

        for (int i : niz) {
            System.out.print(i + " | ");
        }
        System.out.println();
    }
}
```



Run:		Test x
		"C:\Program Files\Java\jdk1.8.0_121\bin\java.exe" ...
		Niz sadrži sl. članove:
		0   -3   -6   -9   -12   -15   -18   -21   -24   -27



# Klasa Arrays

```
Arrays.sort(niz);

System.out.println("Sortiran u rastućem redosledu: ");

for (int i : niz) {
    System.out.print(i + " | ");
}

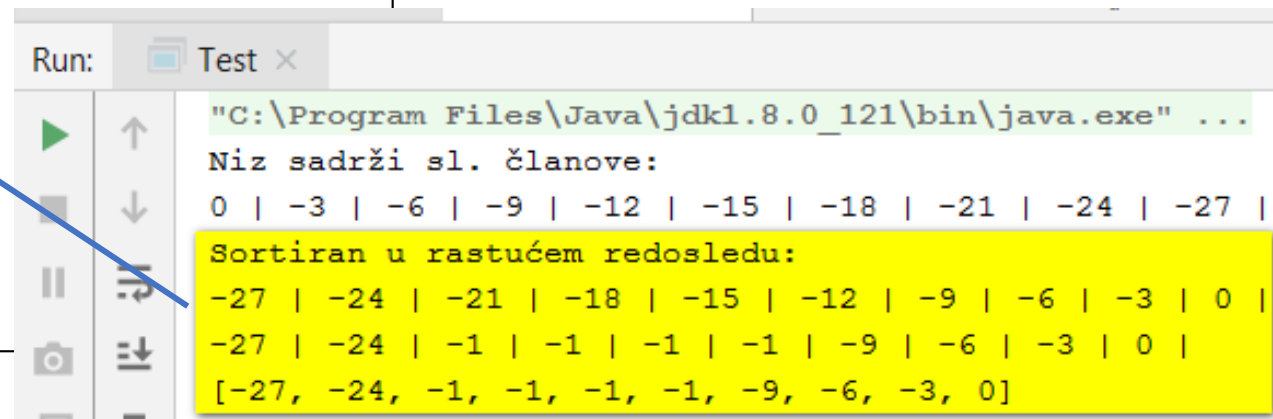
System.out.println();

Arrays.fill(niz, 2, 6, -1);

for (int i : niz) {
    System.out.print(i + " | ");
}

System.out.println();

System.out.println(Arrays.toString(niz));
```



Run: Test ×

"C:\Program Files\Java\jdk1.8.0\_121\bin\java.exe" ...

Niz sadrži sl. članove:

0 | -3 | -6 | -9 | -12 | -15 | -18 | -21 | -24 | -27 |

Sortiran u rastućem redosledu:

-27 | -24 | -21 | -18 | -15 | -12 | -9 | -6 | -3 | 0 |

-27 | -24 | -1 | -1 | -1 | -1 | -9 | -6 | -3 | 0 |

[-27, -24, -1, -1, -1, -1, -9, -6, -3, 0]



# Klasa Arrays

```
int[] kopijaA = Arrays.copyOf(niz, 3);

for (int i : kopijaA) {
    System.out.print(i + " | ");
}

System.out.println();

int[] kopijaB = Arrays.copyOfRange(niz, 5, 10);

for (int i : kopijaB) {
    System.out.print(i + " | ");
}

System.out.println();
}
```

Run: Test x

"C:\Program Files\Java\jdk1.8.0\_121\bin\java.exe" ...

Niz sadrži sl. članove:

0 | -3 | -6 | -9 | -12 | -15 | -18 | -21 | -24 | -27 |

Sortiran u rastućem redosledu:

-27 | -24 | -21 | -18 | -15 | -12 | -9 | -6 | -3 | 0 |

-27 | -24 | -1 | -1 | -1 | -1 | -9 | -6 | -3 | 0 |

[-27, -24, -1, -1, -1, -1, -9, -6, -3, 0]

-27 | -24 | -1 |

-1 | -9 | -6 | -3 | 0 |

Process finished with exit code 0



# Вишедимензионални низови

- Вишедимензионални подаци (нпр. матрице које су дводимензионални низови) могу да се представе у Јави коришћењем **вишедимензионалних низова**.
- Вишедимензионални низ је веома сличан једнодимензионалном низу **у сваком погледу**, а његова декларација се врши на следећи начин:

```
int[][] mojaMatrica = new int[4][5];
```

- Наведеном наредбом се креира дводимензионални низ са 4 реда и 5 колона.
- Колико димензија ће имати ваш низ одређен је бројем угластих заграда, нпр. тродимензионални низ:

```
int[][][] mojaMatrica;
```





# ДВОДИМЕНЗИОНАЛНИ НИЗОВИ

- Ми ћемо на овом предмету да се ограничимо на дводимензионалне низове, јер се сви концепти у вези с њима лако проширују на више димензија.
- Дводимензионални низови се **популарно** називају и "матрице".
- Најчешће, ради лакшег приказивања неких основних концепта, у нашим примерима користимо тзв. квадратне "матрице".
- Квадратна "матрица" је дводимензијални низ који има исти број колона и редова.
- Дводимензионални низ чине елементи истог типа и сваком од њих приступа се помоћу **имена низа** и **два индекса**.
- Ово је **исти принцип** као и код једнодимензионалних низова, само што сада имамо **два индекса**.



# ДВОДИМЕНЗИОНАЛНИ НИЗОВИ

- Дводимензионални низ типа 3 x 4 (ред x колона):

```
public static void main(String[] args) {  
    int[][] a = new int[3][4];  
    a[0][0] = 1; a[0][1] = 5; a[0][2] = 6; a[0][3] = 2;  
    a[1][0] = 4; a[1][1] = 2; a[1][2] = 3; a[1][3] = 3;  
    a[2][0] = 0; a[2][1] = 9; a[2][2] = 8; a[2][3] = 0;  
  
    for (int i = 0; i < a.length; i++) {  
        for (int j = 0; j < a[i].length; j++) {  
            System.out.print(a[i][j] + " | ");  
        }  
        System.out.println();  
    }  
}
```

	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

```
Run: Test x  
"C:\Program Files\Java\jdk1.8.0_12  
1 | 5 | 6 | 2 |  
4 | 2 | 3 | 3 |  
0 | 9 | 8 | 0 |  
  
Process finished with exit code 0
```

- Први индекс указује на ред у "матрици".
- Други индекс указује на колону у "матрици".



# ДВОДИМЕНЗИОНАЛНИ НИЗОВИ

- Елементи дводимензионалног низа се иницијализују уобичајеним подразумеваним вредностима након конструисања таквог низа.
- Међутим, као о код једнодимензионалних низова ви можете вредности задавати одмах:

```
int[][] a = {{3, 1, 4},{1, 5, 9},{2, 6, 5}};
```

- За пролазак кроз дводимензионални низ користимо тзв. дуплу **for** петљу.

```
for (int i = 0; i < a.length; i++) {  
    for (int j = 0; j < a[i].length; j++) {  
        System.out.print(a[i][j] + " | ");  
    }  
    System.out.println();  
}
```

- Форматирањем приказа у конзоли бавимо се у наредбама за штампање: **print()** и **println()**.



# ДВОДИМЕНЗИОНАЛНИ НИЗОВИ

- Дужина дводимензионалног низа се изражава бројем редова.
- Дужина дводимензионалног низа не може бити дефинисана у пару (за редове и колоне).
- Сваки ред може имати различит број елемената (нећемо се бавити оваквим низовима).

```
public static void main(String[] args) {  
    int[][] m = {{ 2, -4, 0, 5 }, { -1, 4, 5}, { -2, 5, 0, 1, 6 } };  
    System.out.println("Dužina dvodimenzionalnog niza je: " + m.length );  
    System.out.println("Duzina reda [0] je: " + m[0].length );  
    System.out.println("Duzina reda [1] je: " + m[1].length );  
    System.out.println("Duzina reda [2] je: " + m[2].length );  
}
```

```
Run: Test x  
"C:\Program Files\Java\jdk1.8.0_121\bin\java.exe"  
Dužina dvodimenzionalnog niza je: 3  
Duzina reda [0] je: 4  
Duzina reda [1] je: 3  
Duzina reda [2] je: 5  
Process finished with exit code 0
```



# Потпрограми

- Потпрограм се састоји од низа наредби и променљивих које представљају неку функционалну целину са својим посебним именом.
- Потпрограм се у литератури назива још функција или метод.
- Потпрограм се може позвати (извршити) у различитим деловима програма.
- Издавају се две целине прилоком писања и коришћења потпрограма:
  - Дефинисање потпрограма.
  - Позивање потпрограма.
- Дефинисањем потпрограма се описује **шта потпрограм ради**, тј. писањем скупа наредби које чине **тело потпрограма**.
- Позивањем потпрограма се **извршава дефинисани потпрограм**, односно извршава се скуп наредби које чине тело потпрограма у дефиницији.
- Најчешће се користе за извршавање неког специфичног задатка.



# Потпрограми

- До сада смо користили искључиво Јавине методе, `print()` и `println()` су само неки од примера.
- Ми сада учимо како да дефинишемо и користимо **сопствене методе**, које ћемо најчешће користити као део главног програма (због тога се и називају потпрограми).
- Метод се дефинише:
  - модификатором приступа (за сада користимо увек **public**),
  - типом повратне вредности (могу бити са и без повратне вредности),
  - својим именом (трудимо се да декларативно задајемо имена),
  - параметрима (улазне променљиве с којима метод ради) и
  - телом метода (овде исписујемо скуп наредби које наш потпрограм извршава).

```
modifikator tip-rezultata ime-metoda(lista-parametara) {  
    skup-naredbi;  
}
```



# Потпрограми

- Дефинисање метода који не мора да врати никакву вредност:

```
public void zbirBrojeva(int x, int y) {  
    // скуп наредби које се извршавају  
    // у телу метода  
}
```

Кључна реч **void** означава метод који не мора да садржи наредбу **return**, тј. не враћа никакву вредност.

- Дефинисање метода који мора да врати повратну вредност:

```
public int zbirBrojeva(int x, int y) {  
    // скуп наредби које се  
    // извршавају у телу метода  
    return x + y;  
}
```

Наредбом **return** се резултат израза и контрола враћају позивајућем методу.



# Потпрограми

## Предности коришћења метода:

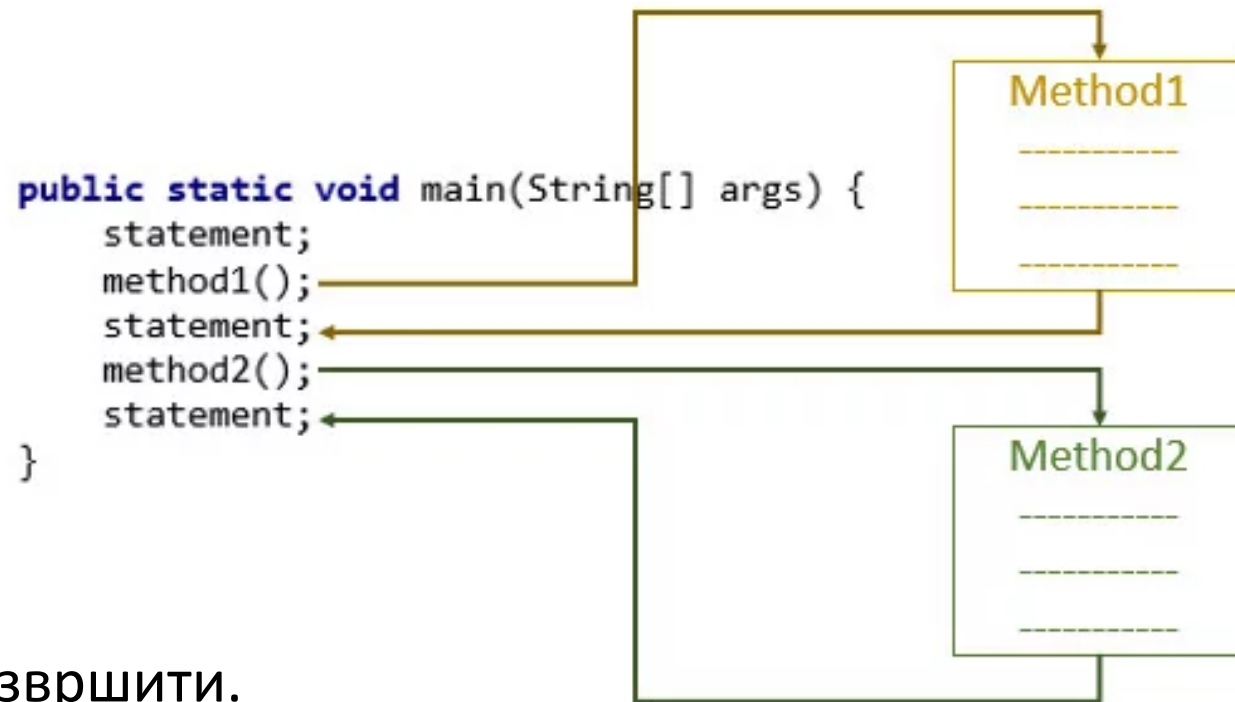
- Мање делове програма је лакше написати и тестирати.
- Методе се могу независно и паралелно писати од стране више програмера.
- Методе се могу користити више пута.
- Методе генерално гледано смањују укупну величину програмског кода.

По конвенцији, име методе се пише малим словом, уколико има више речи свака наредна реч почиње великим словом.



# Потпрограми

- На слици су приказане наредбе за позивање метода (1 и 2).
- Метод се позива наредбом која садржи његово име и листу аргумената.
- Изводимо неколико закључака:
  - Дефинисањем метода се успоставља његово постојање.
  - Метод се уопште не извршава док се не "позове" на неком месту у самом програму.
  - Уколико се метод не "позове" у неком делу програма он се неће никада ни извршити.





# Потпрограми

Неколико речи везано за ООП и статичке методе (за сада још увек само информативно да би сте могли да пратите код у примерима).

```
Test.java x
1 package rs.edu.asss.test;
2 public class Test {
3     public static void main(String[] args) {
4         int br1 = 3, br2 = 8;
5         int suma = zbirBrojeva(br1, br2);
6         System.out.println(suma);
7     }
8     public int zbirBrojeva(int x, int y) {
9         return x + y;
10    }
11 }
```

Non-static method 'zbirBrojeva(int, int)' cannot be referenced from a static context

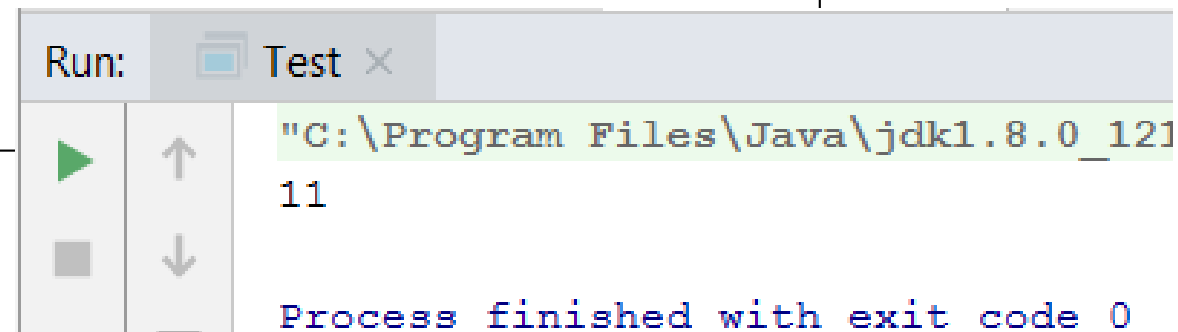
- Статичке методе су методе које обезбеђује класа.
- Могу се позивати и користити без креирања инстанце класе (објекта).
- **Нестатичке методе се не могу позивати из статичког метода** (наш главни метод `main` је иницијално статички).



# Потпрограми

## Пример: метод са повратном вредношћу

```
package rs.edu.asss.test;
public class Test {
    public static void main(String[] args) {
        int br1 = 3, br2 = 8;
        int suma = zbirBrojeva(br1, br2);
        System.out.println(suma);
    }
    public static int zbirBrojeva(int x, int y) {
        return x + y;
    }
}
```

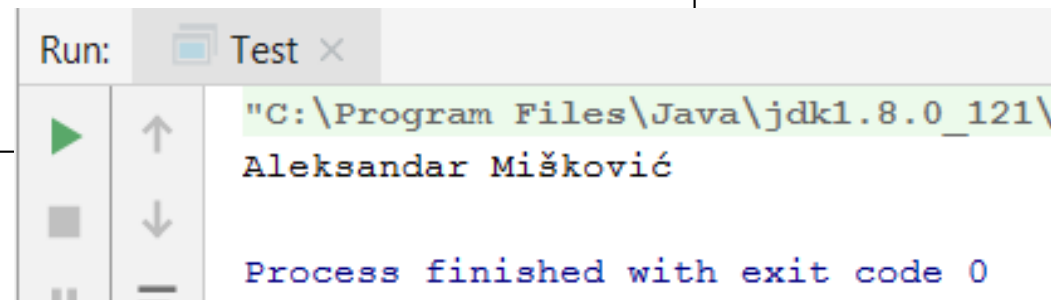




# Потпрограми

## Пример: метод без повратне вредности

```
package rs.edu.asss.test;
public class Test {
    public static void main(String[] args) {
        String ime, prezime;
        ime = "Aleksandar";
        prezime = "Mišković";
        imePrezime(ime, prezime);
    }
    public static void imePrezime(String a, String b) {
        System.out.println(a + " " + b);
    }
}
```





# Следећи час

---

Настављамо с потпрограмима:

- Позивање метода - детаљније.
- Преклапање метода (информативно, више о томе на ООП-у).
- Рекурзивне методе (информативно уз пар примера).
- Упутство (савети) за прву проверу знања.



# Литература

Градиво шестог предавања :

- Поглавље 8, до краја, и поглавље 6:

<https://singipedia.singidunum.ac.rs/izdanje/40716-osnove-java-programiranja>

- Ако пратите препоручене видео лекције, градиво које смо обрадили у овом предавању се односи на видео лекције 8, 9 и 10.

<https://www.youtube.com/playlist?list=PL-UTrxFOy8kK49N01V5ttb2Xaua7JfyXu>

- Одабрана поглавља из књиге: **Java JDK9: Комплетан приручник**
  - Аутор: Herbert Schildt (може и старије издање JDK7).
  - Не морате да купујете наведену књигу.

За испит је, поред скрипти са предавања и вежби, обавезно да учите из књиге која је линкована на почетку.