



Академија струковних  
студија Шумадија  
одсек у Крагујевцу

# Увод у програмирање

## Презентација 7

---

Академија струковних студија Шумадија

Одсек у Крагујевцу

Студијски програм Информатика

Крагујевац, 2020. година

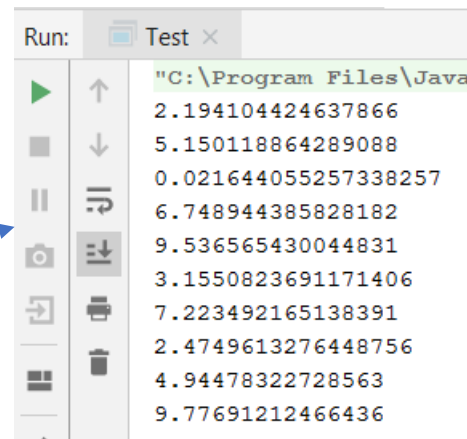
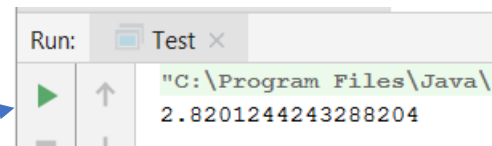


# Метода `Math.random()`

- Подсетићемо се на који начин функционише статички метод `random()` који је дефинисан у класи **`Math`**, у Јавином пакету **`java.lang`**.
- Да бисмо користили статичку методу неке класе није потребно имати инстанцу те класе (о томе ускоро на ООП-у).
- Статичким методама се приступа на следећи начин:  
**`(име_класе) . (име_статичке_методе) ;`**

```
public static void main(String[] args) {  
    double a = Math.random() * 10;  
    System.out.println(a);  
}
```

```
public static void main(String[] args) {  
    for(int i = 0; i < 10; i++){  
        double a = Math.random() * 10;  
        System.out.println(a);  
    }  
}
```





# Метода `Math.random()`

- У приказаним примерима смо дефинисали једну променљиву типа **double**, коју смо иницијализовали помоћу статичке методе **`Math.random()`**.
- Пример са петљом **for** нам служи да вам прикажемо да је коришћењем статичке методе **`Math.random()`** заиста добијен (сваки пут) насумична број у опсегу од 0.0 – 9.99999...
- Ова метода увек "враћа" неку вредност у опсегу између  **$0.0 \leq \text{број} \leq 1.0$** .
- Множењем тог опсега **са неким бројем** ми заправо добијамо **наш жењени опсег** за генерисање насумичног броја, нпр., као у приказаном примеру, множењем бројем 10 добијамо опсег од 0.0 – 9.99999...
- Да би смо добили целобројне вредности, које почињу **од броја 1**, морамо да **"кастујемо"** (експлицитна конверзија) статичку методу и да њен опсег померимо за 1 (запамтите да она даје опсег и када се "кастује" од 0 – 1).



# Метода `Math.random()`

- Зашто је ово битно да разумете?
- Пре свега јер **треба да разумете** како ова метода функционише, а други разлог је **провера знања која следи**.

**Пример:** генерисање низа насумично одабраних целих бројева.

```
public static void main(String[] args) {  
    int[] niz = new int[10];  
    for (int i : niz) {  
        niz[i] = (int) ((Math.random() * 10) + 1);  
        System.out.print(niz[i] + " | ");  
    }  
}
```

За генерисање насумичних бројева у Јави можете користити и Јавину класу **Random**.

Користи се на сличан начин као и приказана

**`Math.random()`** статичка метода.

За дефинисање опсега од неке минималне до неке максималне вредности користите следећи код:

```
Math.random() * (max - min + 1) + min
```

Test x

```
"C:\Program Files\Java\jdk1.8.0_121\bin\ja  
6 | 8 | 1 | 8 | 8 | 2 | 2 | 10 | 9 | 6 |  
Process finished with exit code 0
```



# Метода Math.random()

**Пример:** генерисање низа насумично одабраних целих бројева у опсегу од 200 – 500, а затим и у опсегу од -100 до 100 (*проблем: никада нећемо добити минималну вредност код негативног броја за минимум, за позитиван то не важи*).

```
public static void main(String[] args) {
    int min = 200, max = 500;
    int[] niz = new int[10];
    for (int i : niz) {
        niz[i] = (int) ((Math.random() * (max - min + 1)) + min);
        System.out.print(niz[i] + " | ");
    }
}
```

Test x

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

207 | 208 | 315 | 450 | 211 | 492 | 353 | 231 | 499 | 319 |

Process finished with exit code 0

```
public static void main(String[] args) {
    int min = -100, max = 100;
    int[] niz = new int[10];
    for (int i : niz) {
        niz[i] = (int) ((Math.random() * (max - min + 1)) + min);
        System.out.print(niz[i] + " | ");
    }
}
```

Test x

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

46 | 50 | -67 | 34 | 16 | -79 | -75 | 57 | 47 | -96 |

Process finished with exit code 0



# Позивање метода

- Наредба позивања методе има општи облик:

*ime-metoda(lista-argumenata)*

- У загради се као листа аргумената наводе стварни аргументи за позив методе уместо формалних параметара у дефиницији методе.
  - Ако се подсетимо првог примера са прошлог часа, то су били аргументи **br1** и **br2**.
  - У поменутом примеру, формални параметри методе **zbirBrojeva** су биле локалне променљиве **int x** и **int y**.
- Дакле, у листи аргумената на месту где се позива метода наводе се стварни аргументи **без навођења типова података**.
- Као аргументи позива метода могу се навести и **изрази**.
- Уколико се ради о методи са повратном вредношћу тип стварног аргумента **морда да се поклапа** са типом методе (примери са прошлог часа то приказују).



# Позивање метода

- Метода се може позвати на било ком месту у вашем програму, а најчешће једна метода се позива у другој методи на оном месту где је потребан резултат позване методе.
- Аргументи **у позиву** методе се раздвајају зарезима и у позиву методе **морају се слагати** по броју, типу и редоследу **како су дефинисани** приликом креирања методе.
- Уколико метода нема параметре, обавезне су празне заграде, и у дефиницији и у позиву методе.

```
double metod1 (double x, double y, double z)
```

Дефинисање методе са три параметра типа `double`

```
void metod2 (String a, int b)
```

Дефинисање методе са два различита типа параметра

```
void metod3 ()
```

Дефинисање методе који не садржи параметре



# Позивање метода

- Када се позове метода са параметром, **вредност аргумента** се прослеђује параметру.
- Овакав начин прослеђивања се још назива прослеђивање по **вредности**.
- Уколико је стварни аргумент **променљива**, а не литерал, вредност променљиве се прослеђује **параметру**.
- Вредност прослеђене променљиве **се не мења**, без обзира на промене параметра унутар тела методе.
- Када се аргументи прослеђују по вредности, **прави се локална копија аргумената**.
- Ако се вредности параметара промене **унутар тела методе**, то значи да се промени вредност **само локалних копија** аргумената.
- Да напоменем још једном, **оригинални аргументи остају непромењени**.





# Преклапање метода

- Да би се неки метод могао позвати ради извршавања, мора се познавати његово **име**, као и **број, редослед** и **типови** његових параметара (ово смо већ разјаснили на претходни слајдовима).
- Ове информације се називају **потпис методе**.

```
public static void metod2 (String a, int b) {  
    //naredbe  
}
```

Име методе

Број, редослед и типови  
параметара

- **Једна класа** може садржати дефиниције **више метода** са **истим именом**.



# Преклапање метода

- Да би класа могла да садржи више метода истог имена, мора да буде задовољен услов да свака од тих метода **има различит потпис**.
- Методе из једне класе са истим именом и различитим потписом се називају преклопљене методе (**преоптерећене**, енгл. *overloaded*).
- Који је метод позван у неком делу програма Јава преводаилац јасно одређује на основу потписа свеке методе (пошто имају иста имена, разликују се по броју и/или типу параметара).
- Ово је појам који ћемо детаљније да обрађујемо у ООП-у.


```
public static void mojMetod (String a, int b) {  
    //naredbe  
}  
public static void mojMetod (double x) {  
    //naredbe  
}
```

Исто име методе, са различитим бројем параметара, чак се и тип параметара разликује (што не мора да буде случај).



# Преклапање метода

```
public class Test {  
    public static void suma(int[] n) {  
        int sum = 0;  
        for (int i = 0; i < n.length; i++) {  
            sum += n[i];  
        }  
        System.out.println("Suma niza brojeva je: " + sum);  
    }  
    public static void suma(double[] n) {  
        double sum = 0;  
        for (int i = 0; i < n.length; i++) {  
            sum += n[i];  
        }  
        System.out.println("Suma niza brojeva je: " + sum);  
    }  
    public static void main(String[] args) {  
        int[] nizA = {7, 4, 0, -3, 5, 8};  
        double[] nizB = {7.5, 4.22, 0.38, -3.48, 5.15, 8.33};  
        suma(nizA);  
        suma(nizB);  
    }  
}
```



```
Run: Test  
"C:\Program Files\Java\jdk1.8.0_121\  
Suma niza brojeva je: 21  
Suma niza brojeva je: 22.1  
Process finished with exit code 0
```



# Рекурзивне методе

- Методе у Јави могу у својој дефиницији да **позивају сами себе**.
- Такав начин решавања проблема се назива **рекурзија**.
- Рекурзија је важна и ефикасна техника за решавање **одређених сложених проблема**.
- Важно је напоменути и запамтити да се сваки рекурзивни проблем може решити и писањем класичног **итеративног поступка**.
- Метода позива саму себе уколико се **у њеној дефиницији** налази позив самог метода који се дефинише.
- Механизам позивања и извршавања рекурзивних метода се не разликује од уобичајеног поступка који смо до сада користили (научили).
- Аргументи у позиву се преносе параметрима методе и са тим почетним вредностима се извршава тело методе.



# Рекурзивне методе

- Примери проблема који се могу решити рекурзивним приступом има пуно.
- У литератури се најчешће налазе примери израчунавања факторијела броја, Фибоначијев низ, Еуклидов алгоритам, проблем Хановских кула, ...
- Када се пишу рекурзивне методе, обавезно је дефинисати (**обезбедити**) **излазак из рекурзије** – у супротном се улази у бесконачну петљу.
- У телу рекурзивног метода мора разликовати **базни случај** за најпростији задатак **чије је решење унапред познато** и које се не добија рекурзијом.
- Рекурзија, уопштено гледано, омогућава писање елегантнијих решења од коришћења итеративног поступка (већ је напоменуто да је сваки проблем могуће решити оисањем кода уз одговарајући број итерација).
- Битно је напоменути да рекурзивне методе троше много више меморије него итеративне, које решавају исти проблем.



# Рекурзивне методе

Један од задатак са тестирања кандидата за обуку (*frontend* – HTML, CSS и JS), који се спроводио у нашој Школи (Одсеку).

- Нека је резултат позива функције `max()` максимум целих бројева `x` и `y`. Написати наредбу којом се коришћењем функције `max()` одређује максимум целих бројева `x`, `y` и `z`.

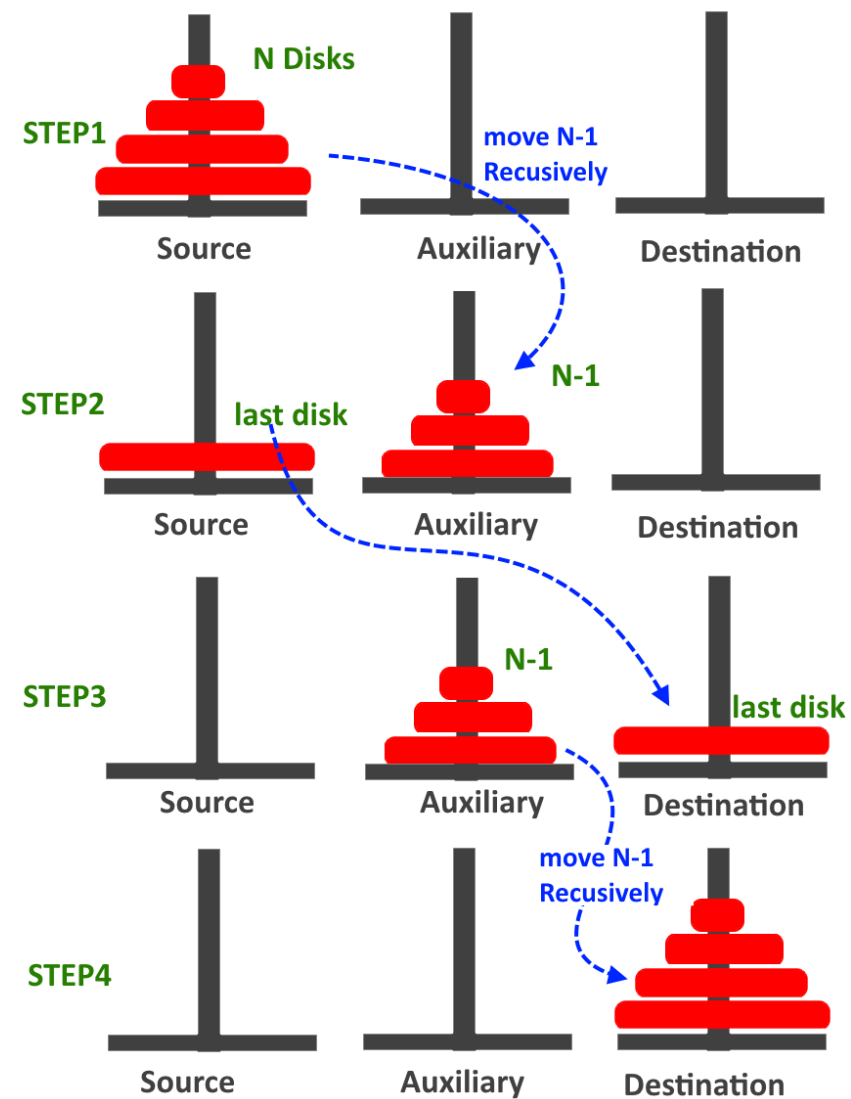
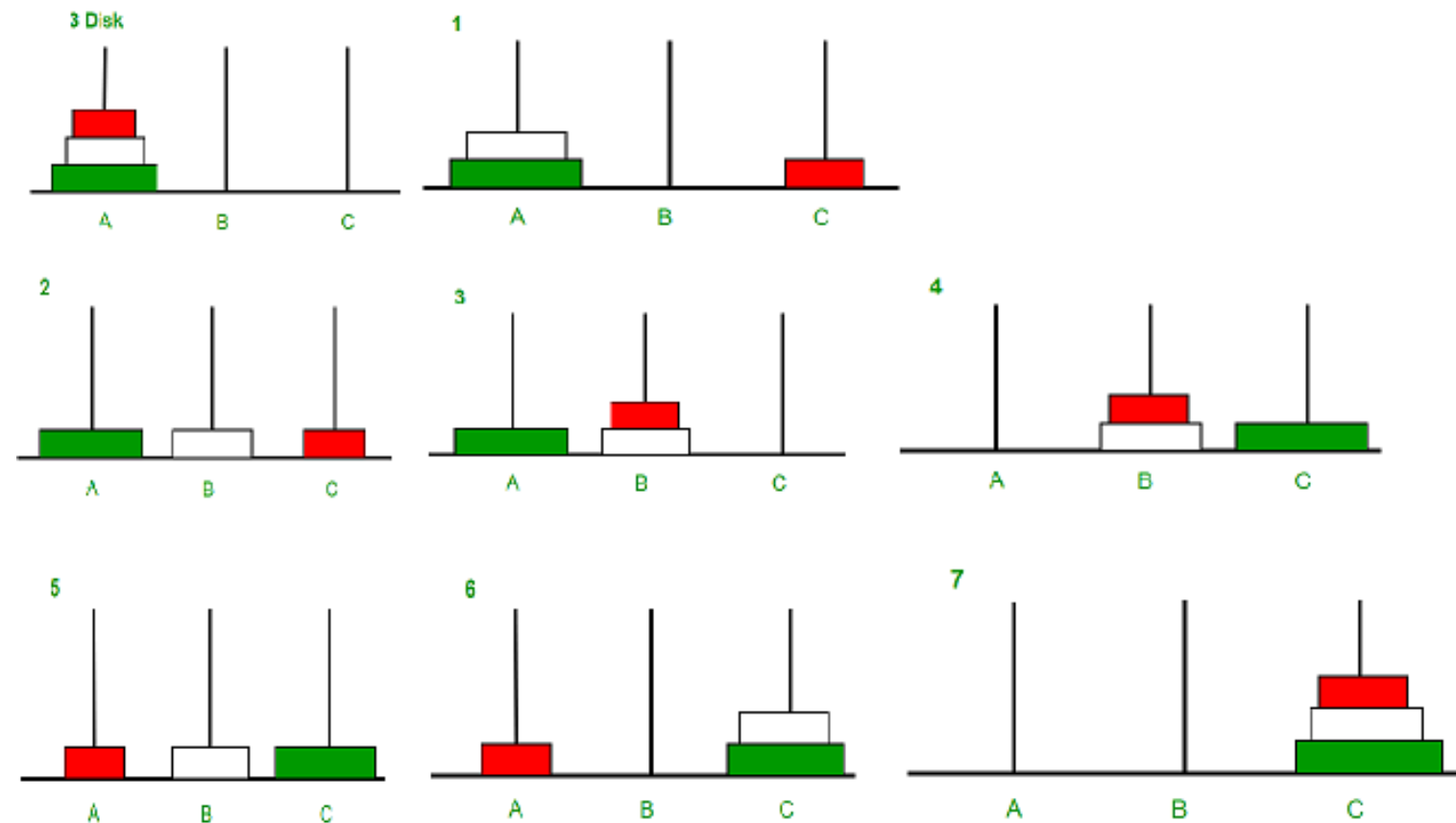
```
public class Test {  
    public static int max(int x, int y) {  
        int max;  
        if (x > y) {  
            return max = x;  
        } else {  
            return max = y;  
        }  
    }  
  
    public static void main(String[] args) {  
        System.out.println("Unesi dva cela broja:");  
        Scanner ulaz = new Scanner(System.in);  
        int x = ulaz.nextInt(); int y = ulaz.nextInt();  
        System.out.println("Uneti su brojevi: " + x + " i " + y);  
        System.out.println("Veći broj je: " + max(x, y));  
        System.out.println("Unesi još jedan broj: ");  
        int z = ulaz.nextInt();  
        System.out.println("Najveći broj je: " + max(z, max(x, y)));  
    }  
}
```

Рекурзивни позив

```
Run: Test x  
"C:\Program Files\Java\jdk1.8.0_121\  
Unesi dva cela broja:  
2  
6  
Uneti su brojevi: 2 i 6  
Veći broj je: 6  
Unesi još jedan broj:  
5  
Najveći broj je: 6  
  
Process finished with exit code 0
```

# Ханојске куле

- Принцип решавања и рекурзивни метод



# Припрема за колоквијум







# Литература

Градиво седмог предавања :

- 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-UTrxF0y8kK49N01V5ttb2Xaua7JfyXu>