Академија струковних
студија Шумадија
одсек у Крагујевцу

Увод у програмирање

Презентација 5

Академија струковних студија Шумадија

Одсек у Крагујевцу

Студијски програм Информатика

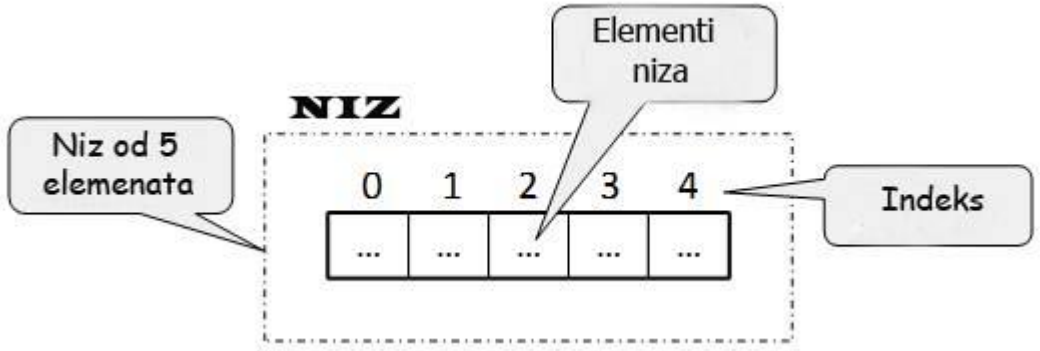
Крагујевац, 2020. година

Низови

- Основна јединица за чување података је променљива.
- Једна променљива може у сваком тренутку садржати само једну вредност.
- У неким случајевима је потребно истовремено имати више сродних података који чине једну целину.
- За такве типове података потребна нам је напреднија структура у коју такве податке можемо да сместимо, да им лако приступимо, уклањамо...
- Ако се има у виду колекција података организованих у овом смислу, онда се говори о структури података.
- Најосновнија структура података, коју смо управо описали, са заједничким именом у Јави се назива низ.
- **Низови су променљиве истог типа – примитивног или класног.**

Низови

- Појединачне променљиве низа називамо елементима или члановима низа.
- Сваки елемент низа има свој индекс који означава позицију елемента у низу.
- У Јава програмском језику индекс првог елемента је 0 (нула).
- Укупан број елемената низа се назива дужина низа.



- Дужина низа је једина "хоризонтална" димензија елемената низа, па се за ову врсту низова каже да су једнодимензионални.

Низови

- Низ се као целина представља једном променљивом специјалног типа, специјална врста објеката.
- Ради једноставнијег изражавања, ова сама променљива се често назива **низ**, мада је прецизније рећи "променљива која указује на низ елемената".
- Општи облик декларације низа у Јави:

```
tip[] imeNiza = new tip[duzina];
```

```
int[] ocene = new int[6];
```

- Конкретан пример креира низ типа **int**, назива **ocene**, који указује на низ од 6 елемената (оцене: 5, 6, 7, 8, 9 и 10).

Низови

- Ако је базни тип дефинисан да буде **`int`**, онда је свака променљива **`a[0]`**, **`a[1]`**… **`a[5]`** проста целобројна променљива, која се у програму може користити на сваком месту где су дозвољене целобројне променљиве.
- У претходном примеру, референца на новоконструисани објекат низа се додељује променљивој **`ocene`**.
- Дакле, **`ocene`** је променљива класног типа, њена вредност може бити референца на објекат низа или специјална референца **`null`**.
- Примери са другим типовима:

```
boolean[] odgovori = new boolean[10];
String[] studenti = new String[50];
```

Низови

- Низ се може иницијализовати приликом креирања (ређи случај коришћења али потпуно легитиман).

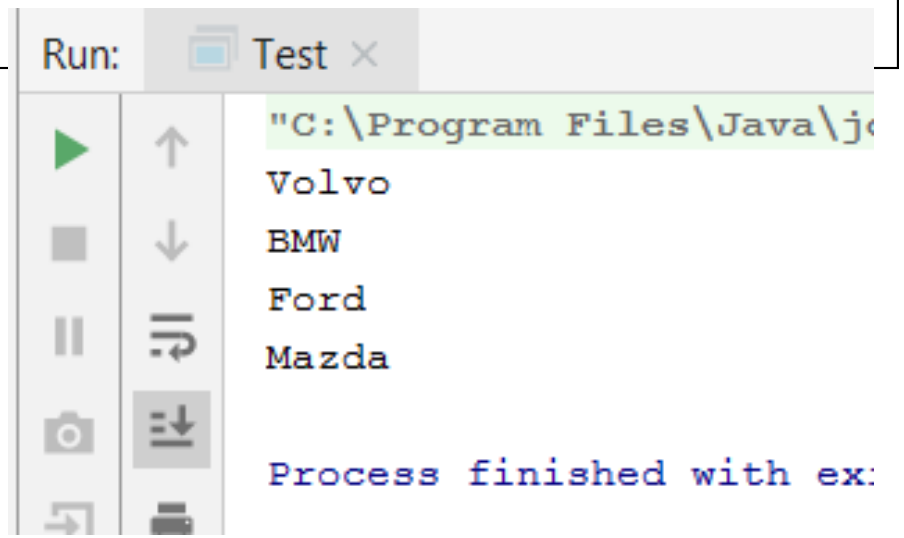
```
int[] ocene = {5, 6, 7, 8, 9, 10};
String[] studenti = {"Sandra", "Lazar", "Nina", "Ana"};
```

- Низови имају фиксну дужину која се не може мењати!
- Ако дефинишемо низ од 10 бројева, он тада може да прими 10 бројева.
- Заправо, дефинисањем низа алоцира се меморија за тај низ, коме је целобројном ненегативном поменљивом дефинисана дужина или капацитет низа.
- У низу морају сви елементи бити иницијализовани или низ не може да се користи.

Низови

Пример

```
public class Test {
    public static void main(String[] args) {
        String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
        for (String i : cars) {
            System.out.println(i);
        }
    }
}
```



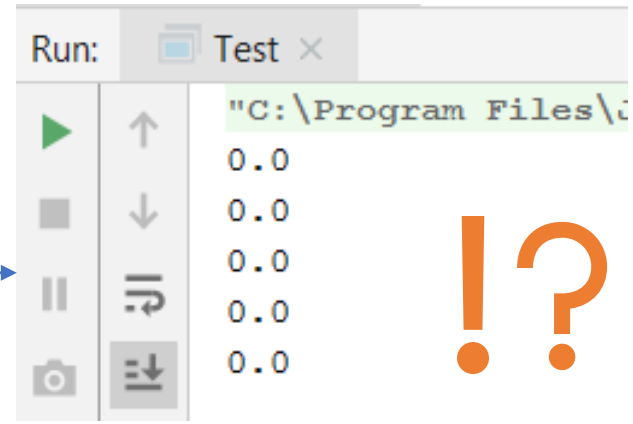
Која петља се користи у наведеном примеру?

Да ли смо могли "обичну" `for` петљу да користимо у овом примеру?

Низови

- У наведено примеру је коришћен класни тип – променљива **`String`**.
- На почетку се указује на низ **`String[]`**, чиме смо заправо и одредили какав тип променљивих ће бити чланови низа – значи елементи низа морају бити типа **`String`**.
- Референца на новоконструисани објекат низа (у примеру испод кључна реч **`new`**) се додељује променљивој **`cars`**. (у овом примеру се низ иницијализује у истом реду у коме се и декларише).
- Исто би се односило да смо креирали низ који садржи просте променљиве.

```
public static void main(String[] args) {
    double[] temperatura = new double[5];
    for (double i : temperatura) {
        System.out.println(i);
    }
}
```



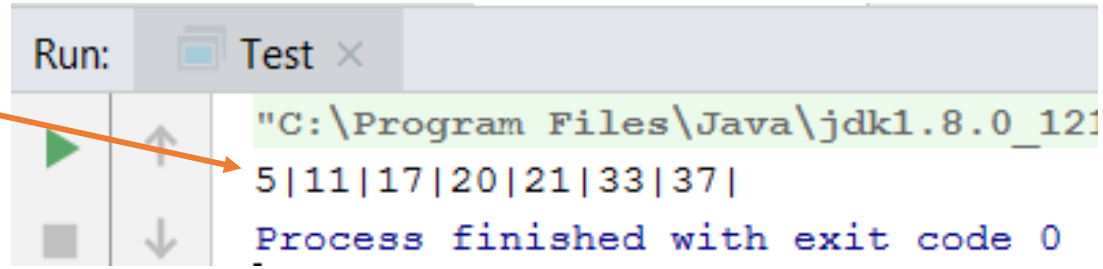
Низови

- У наведеним примерима је коришћена посебна ("побољшана") петља **`for`**.
- Ова петља, изговара се *foreach*, се користи за пролаз кроз низ или колекцију у Јави.
- Лакша је за употребу од једноставне **`for`** петље зато што нема потребе да инкрементирамо вриедност и користимо индексну нотацију.
- Она ради на бази елемената, а не индекса, и враћа елемент један по један у дефинисану променљиву (**`cars`** и **`temperatura`**).
- Употреба *foreach* петље елиминише шансе да се појави грешка при дефинисању петље и њеној аритметици (немогуће је добити грешку **`IndexOutOfBoundsException`**).
- Наравно, петља **`for`** се регуларно користи, поготово уколико нам није потребно да "дохватимо" сваки члан (елемент) низа.

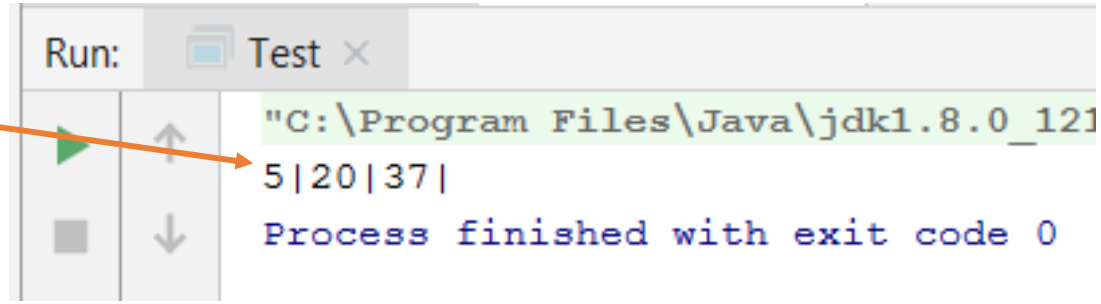
Низови

Пример

```
public static void main(String[] args) {
    int[] lotoBrojevi = new int[7];
    lotoBrojevi[0] = 5; lotoBrojevi[1] = 11;
    lotoBrojevi[2] = 17; lotoBrojevi[3] = 20;
    lotoBrojevi[4] = 21; lotoBrojevi[5] = 33;
    lotoBrojevi[6] = 37;
    for (int i=0; i<lotoBrojevi.length; i++) {
        System.out.print(lotoBrojevi[i] + "|");
    }
}
```




```
public static void main(String[] args) {
    int[] lotoBrojevi = new int[7];
    lotoBrojevi[0] = 5; lotoBrojevi[1] = 11;
    lotoBrojevi[2] = 17; lotoBrojevi[3] = 20;
    lotoBrojevi[4] = 21; lotoBrojevi[5] = 33;
    lotoBrojevi[6] = 37;
    for (int i=0; i<lotoBrojevi.length; i+=3) {
        System.out.print(lotoBrojevi[i] + "|");
    }
}
```



Низови (необавезни пример)

Пример за вежбање (питајте на вежбама за савет али свакако покушајте сами да решите задатак):

- Проширите претходни задатак тако да се насумичним путем извуку "лото бројеви". Наравно, те бројеве је потребно сместити у неки низ.
- Играч може да употреби бројеве које ви унапред дефинишете (као у приказаном примеру), међутим, свакако је боље да играч сам унесе жељене бројеве (*Помоћ*: класа **`Scanner`**).
- Након тога је потребно утврдити колико је бројева играч погодио приликом насумичног извлачења (симулирамо рад "лото бубња").
  - Помоћ: није битно да ли прво насумично генеришете низ од 7 чланова (извучени "лото бројеви") или генеришете корисников низ боројева (корисникова "лото комбинација", такође 7 чланова).
  - Помоћ: потребно је да користите више петљи, да дефинишете услов за поређење у једној од петљи, да штампате резултате (извучени, корисникови и погођени бројеви).

Низови

Када се низ креира али се не иницијализују елементи низа, елементи низа ће се аутоматски инцијализовати на подразумеване вредности, а то су:

- **0** (нула) за нумеричке вредности (*наш пример за температуру*)
- ***false*** за ***boolean***
- ***'\u0000'*** за ***char***
- ***null*** за ***objekte***

Сваки низ има дефинисано константно поље ***length*** које садржи дужину низа (број елемената).

- У претходном примеру смо ово користили (`lotoBrojevi.length`) како би смо "обухватили" све чланове низа.
- Могли смо и "ручно" да их пребројимо (нпр. `i<8`).

Низови

Најчешће грешке

- Могућа два типа грешака које изазивају прекид извршавања програма.
- Први – ако променљива **`cars`** (наш први пример) типа низа садржи вредност ***null***, а покуша се приступ елементу **`cars`**`[i]`.
- Други – индекс ван дозвољених граница.
  - Дешава се ако се израчунавањем индекса **`i`** добије да је **`i < 0`** или **`i >`** **`length`**

Копирање низа

- Променљиву једног низа можете копирати у другу али онда оне обе показују на исти низ.
- Измене у променљивој једног низа ће се одразити на другу променљиву низа, јер показује на исти низ.
- Копирање можете одрадити ручно или коришћењем Јавине класе **`Arrays`**.

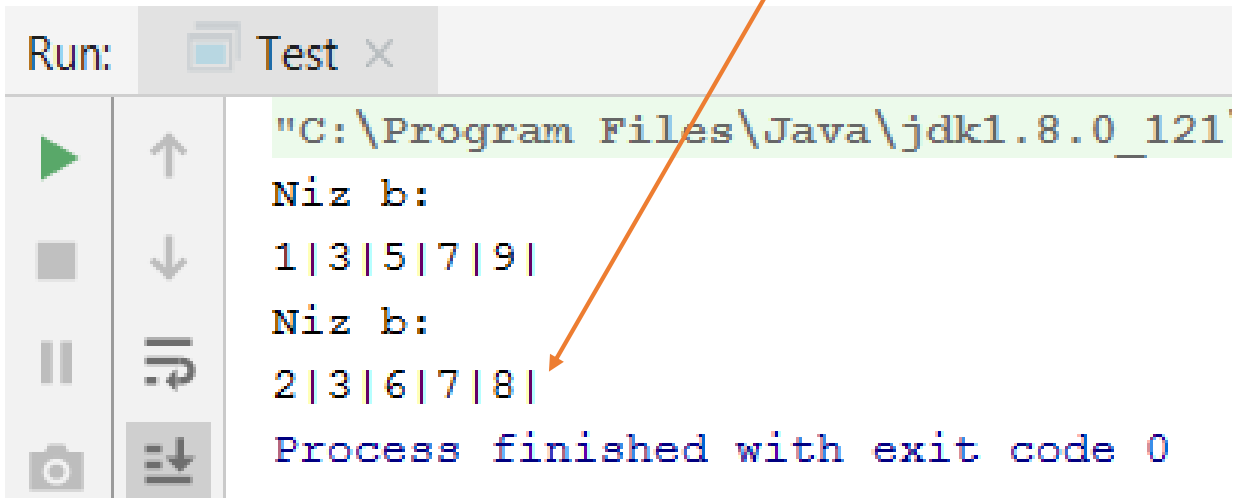
Низови

- Нпр. направимо два низа, **a** и **b**.

```
int[] a = new int[5];
a[0] = 1; a[1] = 3; a[2] = 5;
a[3] = 7; a[4] = 9;
int[] b;
b = a;
System.out.println("Niz b:");
for (int i = 0; i < b.length; i++){
    System.out.print(b[i] + "|");
}
//промена чланова низа a
a[0] = 2; a[2] = 6; a[4] = 8;
System.out.println();
System.out.println("Niz b:");
for (int i = 0; i < b.length; i++){
    System.out.print(b[i] + "|");
}
```

копирање низа **a** у низ **b**

промена чланова низа **a** се пресликава у низ **b**



Зашто је то тако?

Преписује се само референца!

И даље постоји само један примерак сваког елемента низа!

Низови

- Јавина класа **`Arrays`** се налази у **`java.util`** пакету.
- Класа **`Arrays`** знатно олакшава рад са низовима.
- Садржи неколико статичких метода које обезбеђују корисне операције над низовима. У књизи имате објашњење за неколико, овде издвајам:
- За копирање користити:

```
Arrays.copyOf();
```

- За сортирање низа користити:

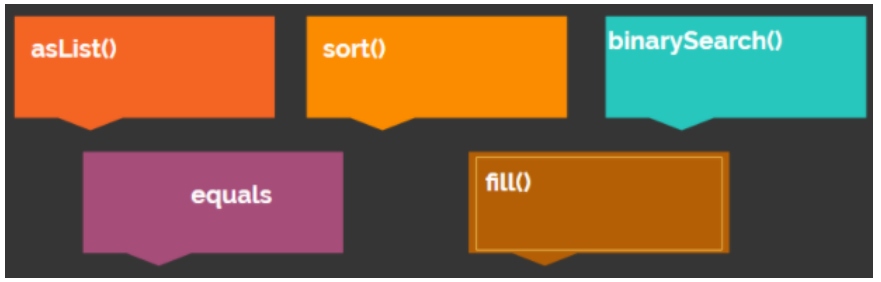
```
Arrays.sort(niz);
```

- Међутим, **<u>није неопходно за полагање овог испита</u>** али је неопходно да знате да примените неке познате алгоритме за сортирање.
- Не морате их учити напамет: *Selection sort, Insertion sort, Bubble sort, Shell sort, Quick sort, Merge sort, …* *https://www.javatpoint.com/sorting-algorithms*

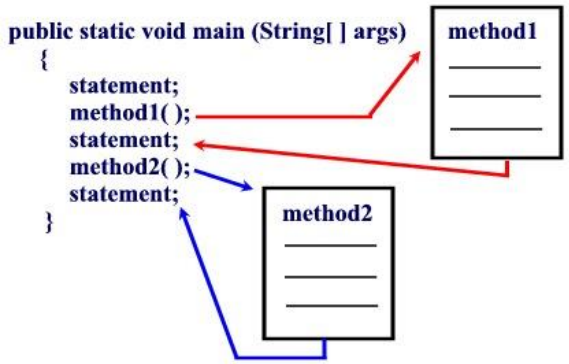
Низови

Следеће предавање:

- Класа **`Arrays`**
- Дводимензионални низови
- Јава потпрограми (фунције или методе).



	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]



Литература

Градиво <u>петог предавања</u> :

- Поглавље 8

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 лекцију 8.

https://www.youtube.com/playlist?list=PL-UTrxF0y8kK49N01V5ttb2Xaua7JfyXu

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

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