NACIONĀLAIS ATTĪSTĪBAS PLĀNS 2020
EIROPAS SAVIENĪBA
Eiropas Sociālais fonds
IEGULDĪJUMS TAVĀ NĀKOTNĒ

Java programmēšanas pamati

7. NODARBĪBA

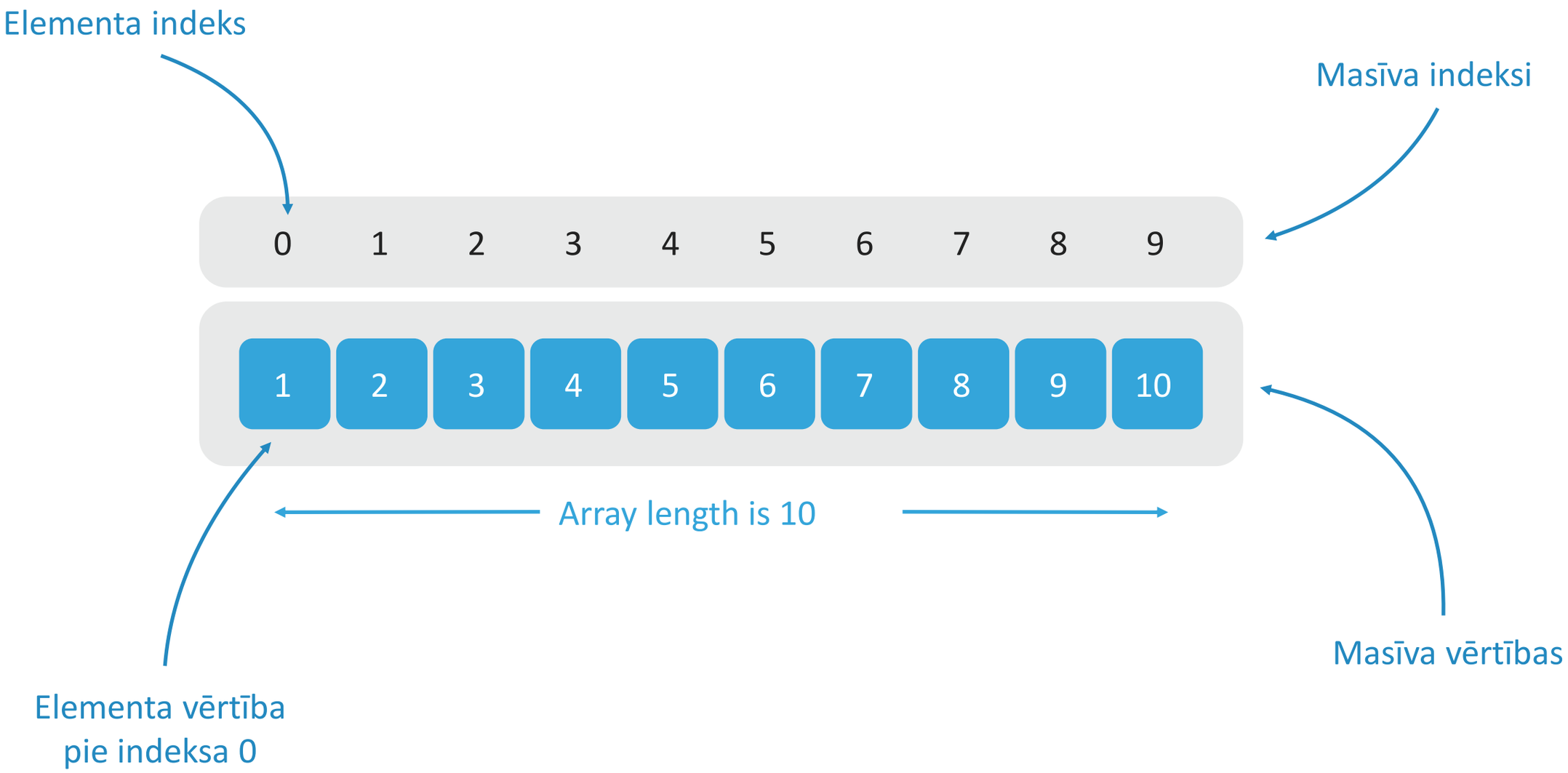
MASĪVI

MASĪVU PĀRSKATS

DEFINĪCIJA

- Masīvs ir konteinera objekts, kas satur fiksētu skaitu viena tipa elementu
- Masīva izmērs (length) tiek noteikts, kad masīvs tiek izveidots
- Pēc masīva izveides, tā izmērs ir nemainīgs

MASĪVU VIZUALIZĀCIJA



MASĪVA DEKLARĒŠANA: SINTAKSE

- Masīva deklarēšana bez objekta izveides

```
type[] name;
```

- Masīva deklarēšana ar objekta izveidi

```
type[] name = new type[size];
```

- Masīva deklarēšana ar objekta inicializāciju (inline)

```
type[] name = {var1,.., varN};
```

MASĪVU IZVEIDOŠANA

Izejas kods

```
int[] leapYears = new int[3];

leapYears[0] = 2020; leapYears[1] = 2016; leapYears[2] = 2012;

System.out.println("Leap years = " + Arrays.toString(leapYears));
```

Konsules izvade

```
Leap years = [2020, 2016, 2012]

Process finished with exit code 0
```

MASĪVU IZVEIDOŠANA (INLINE)

Izejas kods

```
int[] leapYears = {2020, 2016, 2012};

System.out.println("Leap years = " + Arrays.toString(leapYears));
```

Konsules izvade

```
Leap years = [2020, 2016, 2012]

Process finished with exit code 0
```

DARBS AR MASĪVIEM

DARBS AR MASĪVIEM

- Tā kā masīviem piemīt iterējama daba, tad darbā ar masīviem izmanto ciklus
- Masīvs satur viena tipa elemntus un masīva izmērs ir fiksēts un zināms iepriekš

1. MASĪVA ELEMENTU IZVADE

```
public class PrintingArrayDemo {

    public static void main(String[] args) {

        String[] alphabet = new String[5];

        alphabet[0] = "A";
        alphabet[1] = "B";
        alphabet[2] = "C";
        alphabet[3] = "D";
        alphabet[4] = "E";

        for (int i = 0; i < alphabet.length; i++) {
            System.out.println("[" + i + "]: " + alphabet[i]);

        }

    }

}
```

2. MASĪVU ELEMENTU SUMMA

```
public class SumOfArrayElementsDemo {

    public static void main(String[] args) {

        int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9};
        int sum = 0;

        for (int i = 0; i < numbers.length; i++) {
            sum += numbers[i];
        }

        System.out.println("Sum = " + sum);

    }
}
```

MAZĀKAIS ELEMENTS MASĪVĀ

```
public class SmallestArrayElementDemo {

    public static void main(String[] args) {

        int[] numbers = {61, 97, 4, 37, 12};
        int min = numbers[0];

        for (int i = 0; i < numbers.length; i++) {
            if (numbers[i] < min) {
                min = numbers[i];
            }
        }

        System.out.println("min = " + min);

    }

}
```

UZLABOTIE ITERĒŠANAS PAŅĒMIENI

CIKLS “FOR EACH”

- Cikls “for each” ir vienkāršots veids kā “ieziet cauri” masīvam, pārbaudot katru elemetu
- Indeksam jeb skaitītāja mainīgajam nav papildus pielietojuma
- Ciklā “for each” deklarētajam datu tipam ir jāatbilst masīva datu tipam, kurš tiek iterēts
- Var piekļūt tikai tekošajam elementam
- Ievērojami samazina koda daudzumu

CIKLS “FOR EACH”: SINTAKSE

“For each” cikla deklarēšana

```
type[] name = {var1,.., varN};

for (type item : name) {
   statements...
}
```

Iterators

Koda daļa, kas tiek izpildīta ciklā

CIKLS “FOR EACH”: PIEMĒRS

```
public class ForEachDemo {

    public static void main(String[] args) {

        String[] dogBreeds = {
                "Beagle",
                "Golden Retriever",
                "Pug",
                "Shiba Inu"
            };

            for (String breed : dogBreeds) {
                System.out.println(breed);
            }

    }

}
```

STATIC
PĀRSKATS

ATSLĒGVĀRDS “STATIC”

- Atslēgvārds “static” norāda, ka konkrētais eksemplārs pieder pašam tipam nevis šī tipa instancei
- Static definētam elementam tiek izveidota tikai viena instance (eksemplārs), kas tiek iedalīts katrai klases instancei
- Var izmantot sekojošiem elementiem:
  - Lauki (mainīgie)
  - Metodes
  - Iekšējās metodes (Inner)
  - “Static” koda blokiem

“STATIC” LAUKI

- Visas klases instances, kurās ir deklarēts statisks lauks, savā starpā izmanto tikai vienu vienīgu šī lauka kopiju
- Neatkarīgi no klase inicializēšanas skaita, vienmēr tiks izmantota vienīgā statiskā lauka kopija

1. KLASE MESSAGE

```
public class Message {

    public static int instancesCreated = 0;

    private String text;

    public Message(String text) {
        this.text = text;

        System.out.println("Creating message = '" + text + "'");
        instancesCreated++;
    }

}
```

2. OBJEKTS MESSAGE

Code

```
System.out.println("Created = " + Message.instancesCreated);
Message greeting = new Message("Hi!");
Message question = new Message("How are you?");
Message farewell = new Message("Goodbye!");
System.out.println("Created = " + Message.instancesCreated);
```

Console output

```
Created = 0
Creating message = 'Hi!'
Creating message = 'How are you?'
Creating message = 'Goodbye!'
Created = 3
```

KAD IZMANTOT “STATIC” LAUKUS?

- Ja lauka vērtība ir neatkarīga no objekta
- Ja lauka vērtību ir nepieciešams koplietot starp visām klases instancēm

ATCERIES!

- Tā kā statiski lauki pieder klasei, tad piekļut tiem var tieši - izmantojot klases nosaukumu. Objekta norādes nav nepieciešama.
- Statiski mainīgie var tikt deklarēti tikai klases līmenī
- Statiskiem laukiem var piekļūt bez objekta inicializācijas
- Ieteicams piekļūt statiskiem laukiem ir izmantojot klases nosaukumu, kaut arī tas ir iepsējams ar objekta norādi

“STATIC” METODES

- Pieder klasei nevis objektam
- Var izsaukt bez klases objekta veidošanas

1. KLASE QUICKMATHS

```java
public class QuickMaths {

    public static int min(int[] numbers) {
        if (numbers.length == 0) {
            return 0;
        }

        int min = numbers[0];

        for (int number : numbers) {
            if (number < min) {
                min = number;
            }
        }

        return min;
    }

}
```

2. KLASES QUICKMATHS IZMANTOŠANA

Code

```
int[] values = {44, 65, 61, 16, 89};

int result = QuickMaths.min(values);

System.out.println("result = " + result);
```

Console output

```
result = 16

Process finished with exit code 0
```

KAD IZMANTOT “STATIC” METODES?

- Izmanto, lai piekļūtu statiskajiem mainīgajiem, kā arī jebkuriem citiem statiskiem klases elementiem
- Izmanto no instaces veidošanas neatkarīgu operāciju veikšanai
- Plaši izmanto “palīgklasēs” (utility classes), lai tās varētu izmantot neveidojot objektu

ATCERIES!

- Statiskās metodes nevar pārrakstīt (overridden)
- Instances metode var tieši piekļūt gan statiskajām instances metodēm, gan statiskajiem instances objektiem
- Statiskās metodes var piekļūt visiem statiskajiem mainīgajiem, kā arī pārējām statiskajām metodēm
- Statiskās metodes nevar piekļūt instances mainīgajiem un instances metodēm tieši (tikai izmantojot objekta norādi)

ATSAUCES

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
- https://www.javatpoint.com/array-in-java
- https://www.baeldung.com/java-arrays-guide
- https://www.baeldung.com/java-static
- https://www.geeksforgeeks.org/static-keyword-java/

LATVIJAS UNIVERSITATE
ANNO 1919

VUMC
VADĪBAS UN UZŅĒMĒJDARBĪBAS MĀCĪBU CENTRS