



LATVIJAS
UNIVERSITĀTE
ANNO 1919

NACIONĀLAIS
ATTĪSTĪBAS
PLĀNS 2020



EIROPAS SAVIENĪBA

Eiropas Sociālais
fonds

IEGULDĪJUMS TAVĀ NĀKOTNĒ

Java programmēšanas pamati



VADĪBAS UN
UZŅĒMĒJDARBĪBAS
MĀCĪBU CENTRS

9. NODARBĪBA

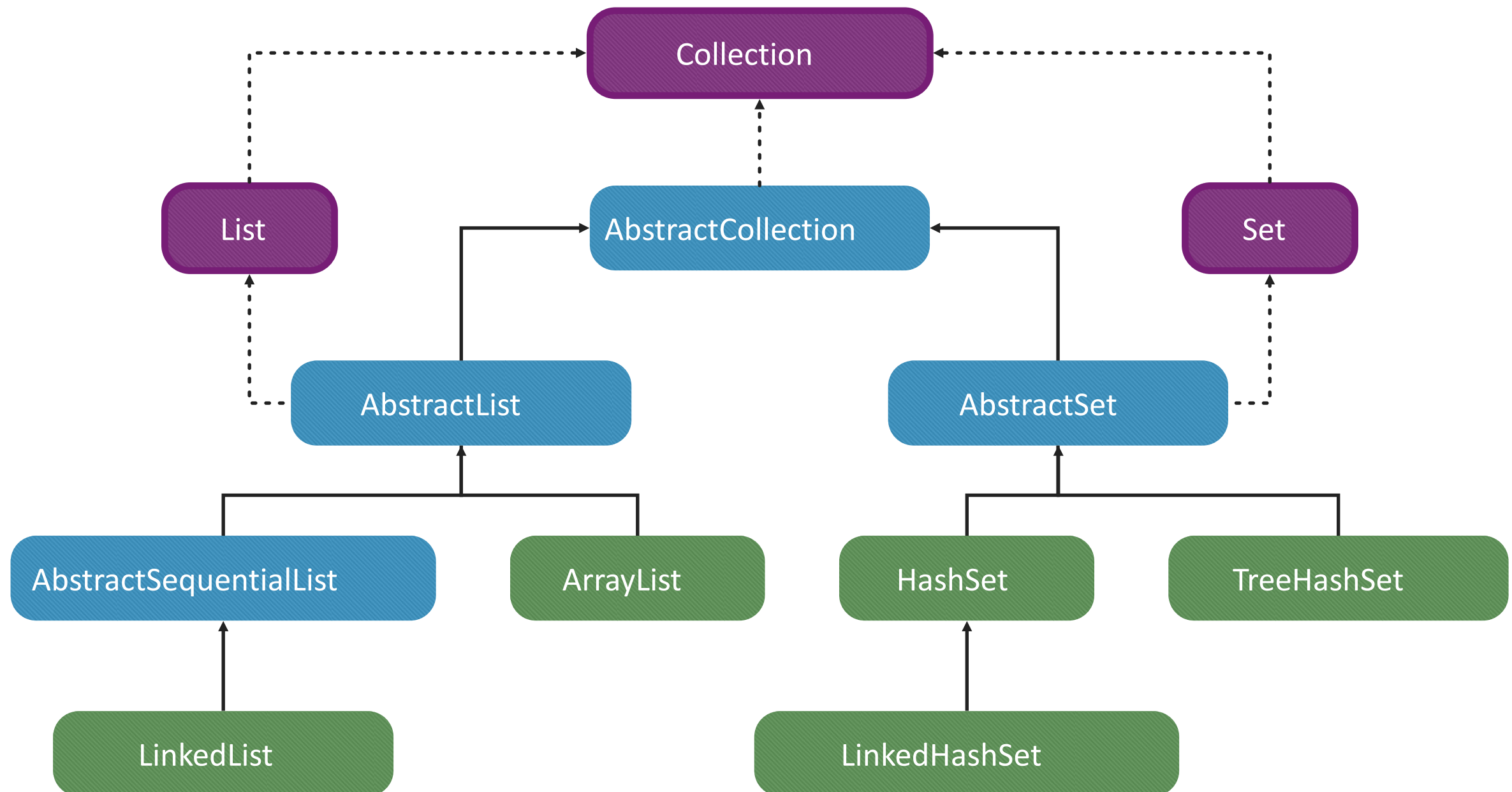
KOLEKCIJAS

IEVADS

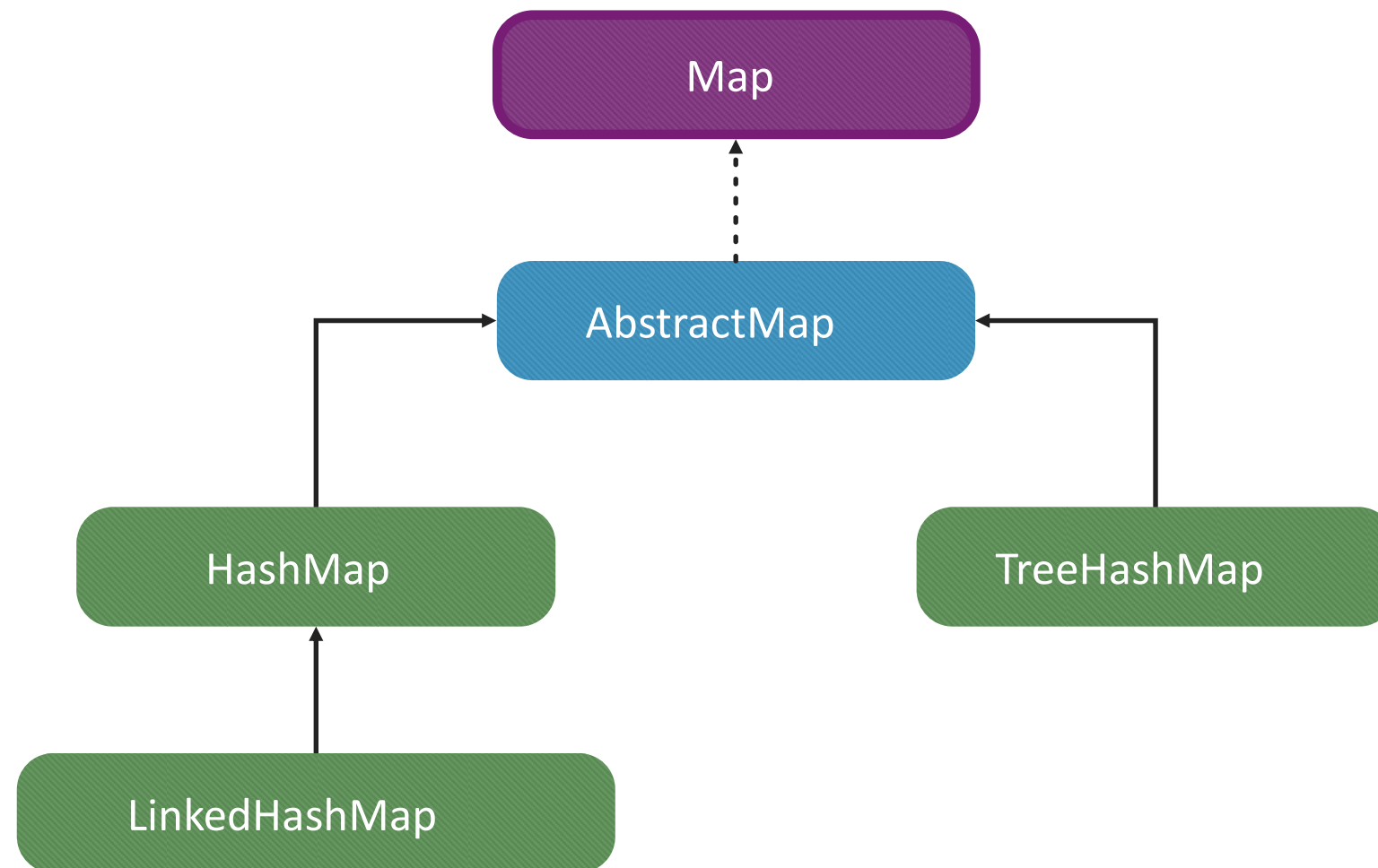
- ▶ Datu struktūras, kā, piemēram, masīvi, ir **vienkāršas** un **ātras**, taču darbs ar tām ir **apgrūtināošs**
- ▶ Sākotnēji Java **piedāvāja** rīkus darbam ar objektu grupām, taču tiem **trūka** vienojošas pieejas
- ▶ Valodas izstrādātāji vēlējās **izveidot** sistēmu, kas **atbilstu** vairākiem nosacījumiem
 - ▶ **Augsta** veikspēja
 - ▶ Augsta **savietojamība** un **abstrakcija**
 - ▶ Viegli **paplašināt (extend)** un **pielāgot** kolekcijas

JAVA STANDARTA KOLEKCIJAS

INTERFEISS “COLLECTION”



INTERFEISS “MAP”



KOLEKCIJU RAKSTURLIELUMI

- ▶ Sakārtota (ordered)
 - ▶ Iespēja **atkārtoti** iterēt sakārtotas kolekcijas elementus **paredzamā** kārtībā
- ▶ Elementu unikalitāte
 - ▶ Dažās kolekcijās **nav atļauts dublēt** elementus
- ▶ Plūsmu drošība (thread safety)
 - ▶ **Drošs** darbs ar kolekciju **daudzplūsmu** vidē

KOLEKCIJU VEIDI

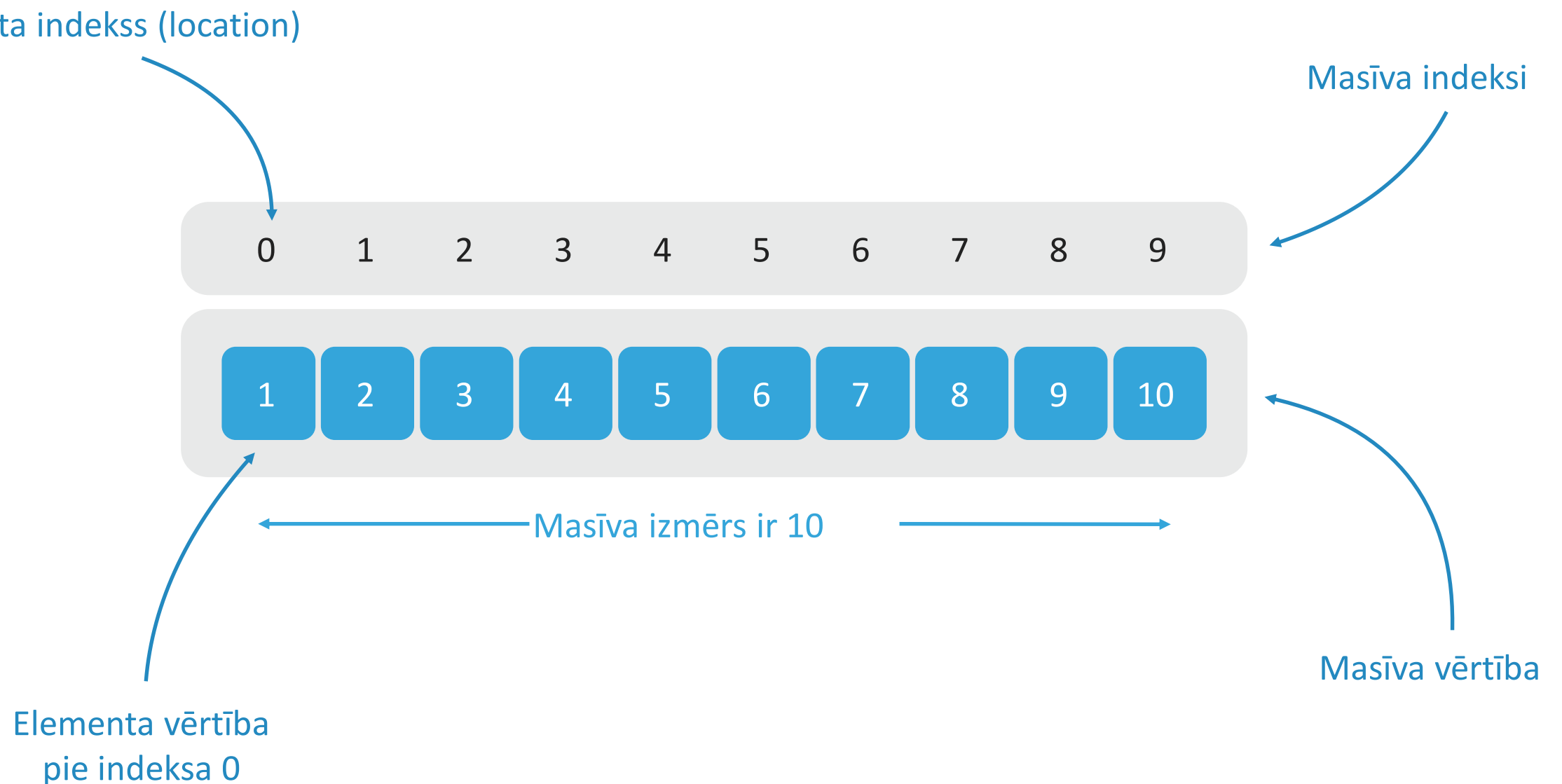
- ▶ Masīvu (array) tipa krātuve
 - ▶ Ātra elementa lasīšana, bet salīdzinoši lēna dzēšana vai ievietošana
- ▶ Saistīta-saraksta (linked-list) tipa krātuve
 - ▶ Ātra elementa dzēšana vai ievietošana, bet lēna nolasīšana
- ▶ Haš (hash) tipa krātuve
 - ▶ Efektīva piekļuve
- ▶ Koks (tree) tipa krātuve
 - ▶ Efektīva meklēšanai

JAVA SARAĶSTS: ARRAYLIST

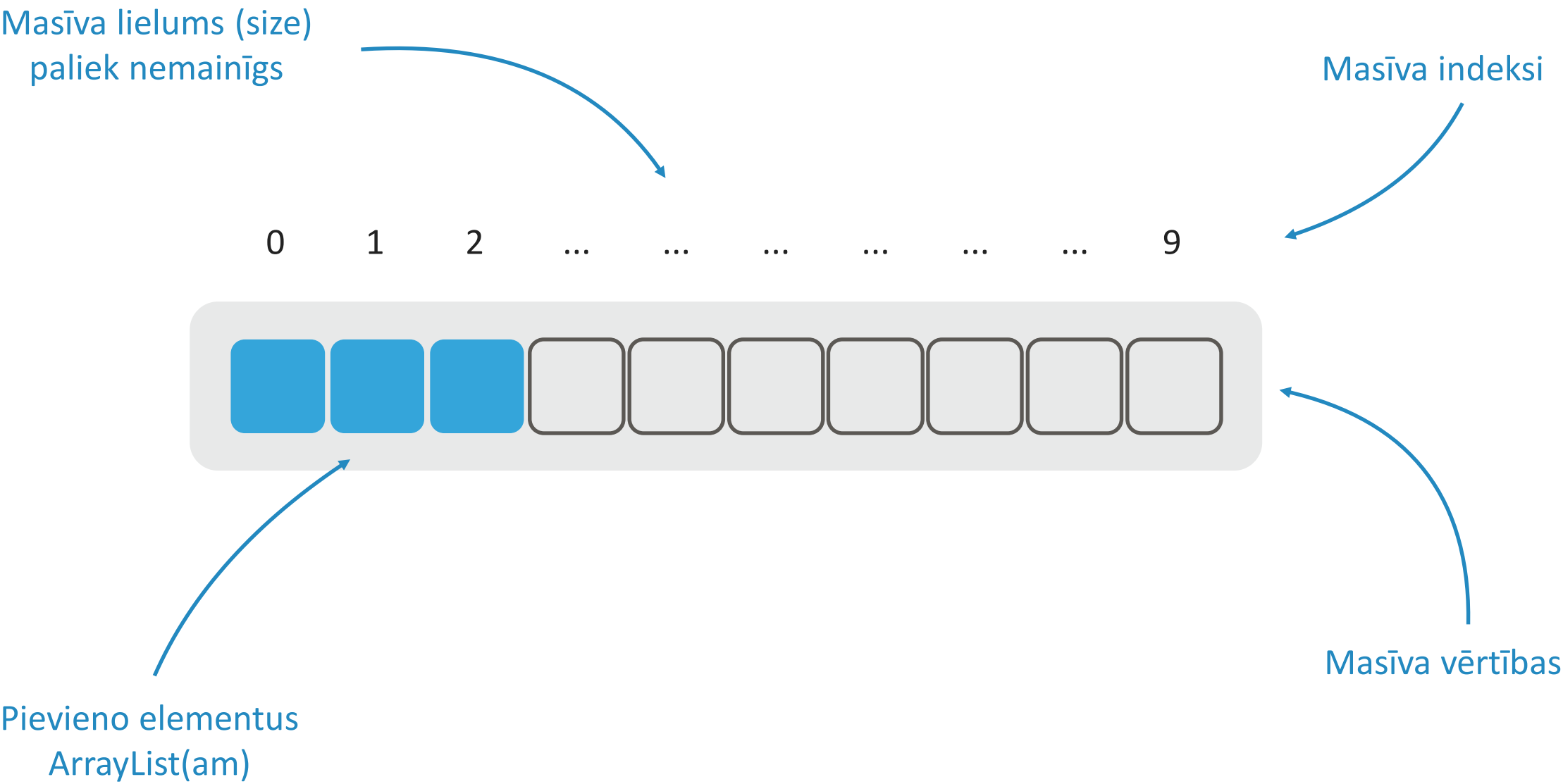
SARAKSTA IMPLEMENTĀCIJA MAINĪGA IZMĒRA MASĪVAM

Java dokumentācija

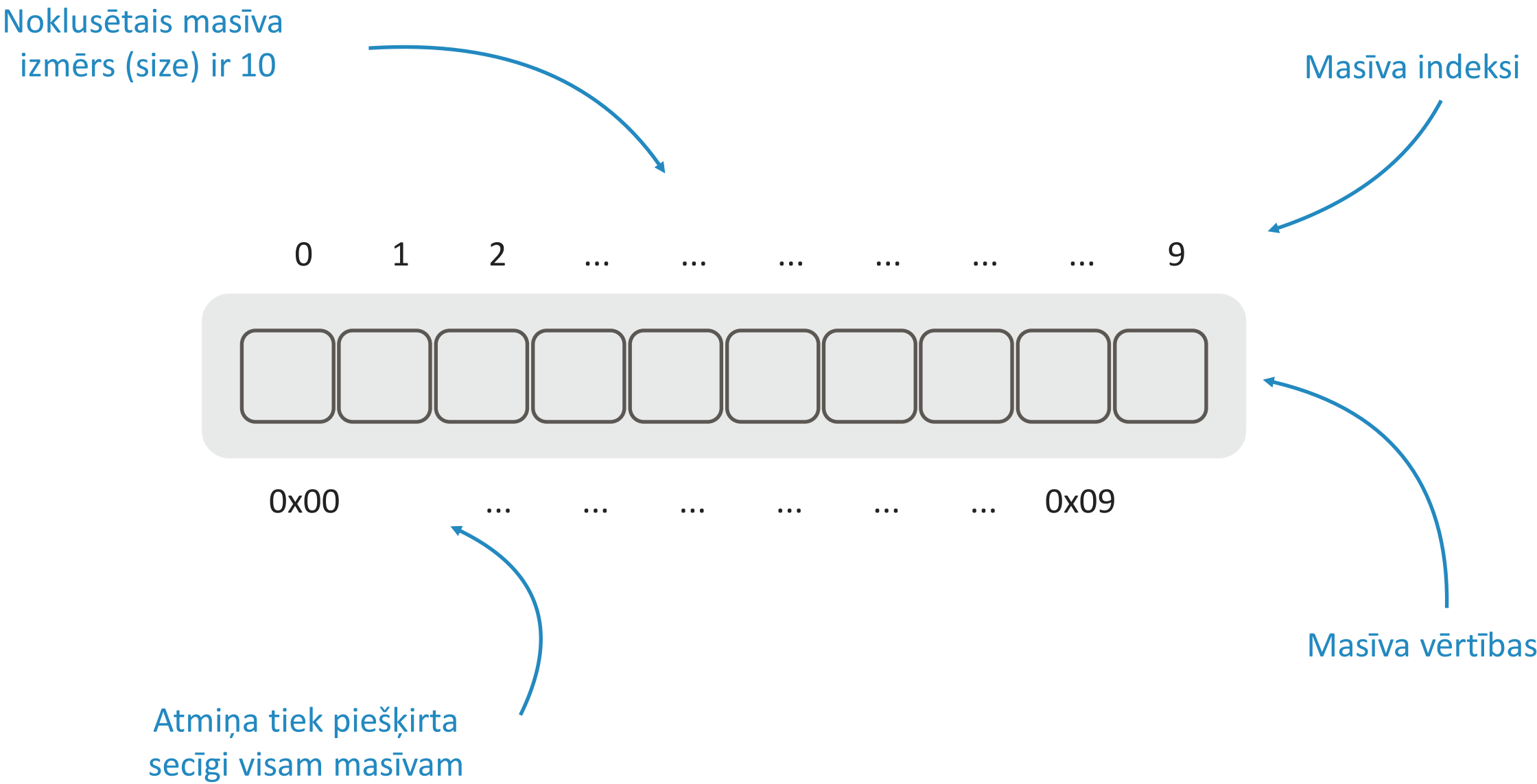
ARRAYLIST: DATU STRUKTŪRA



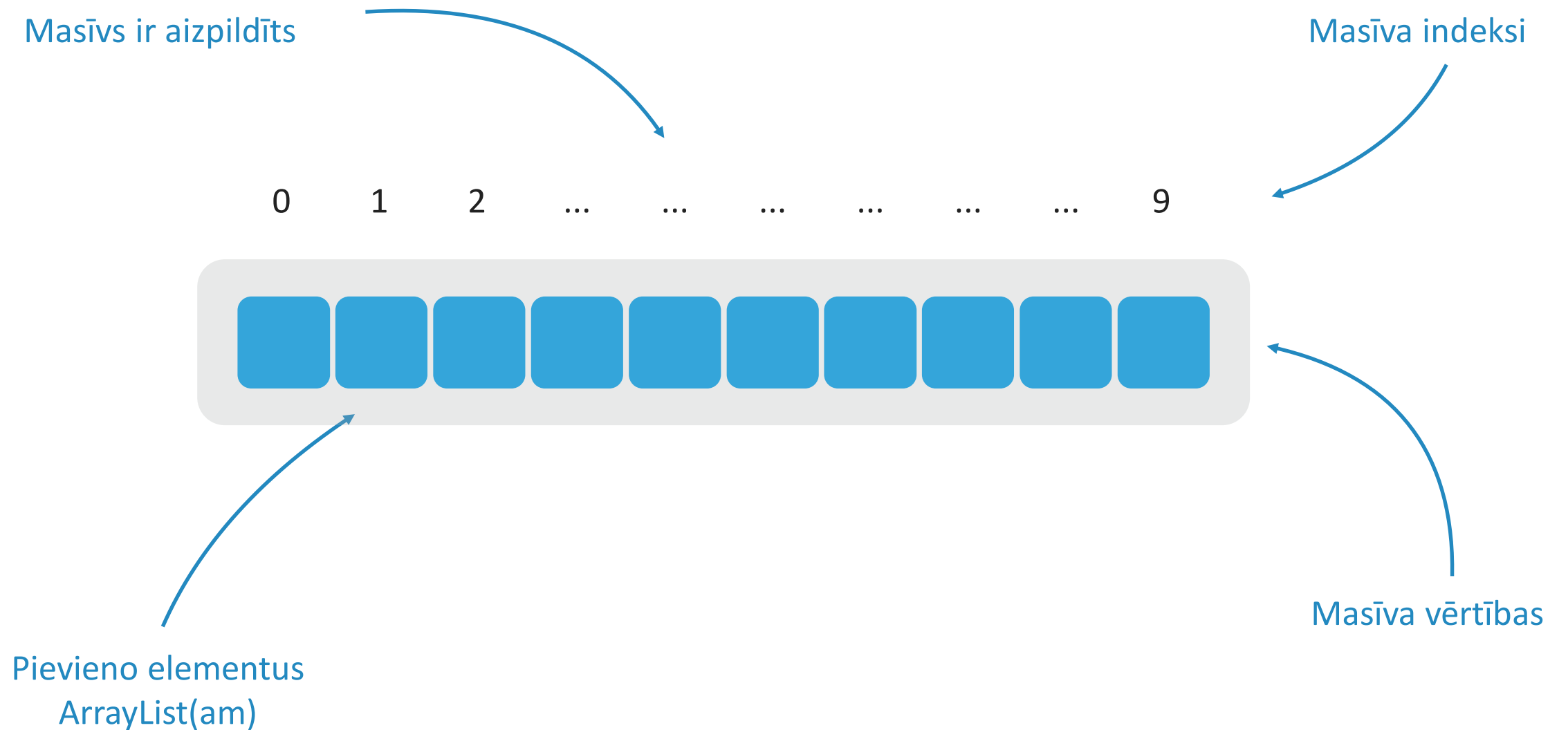
1. ARRAYLIST: ELEMENTA PIEVIENOŠANA



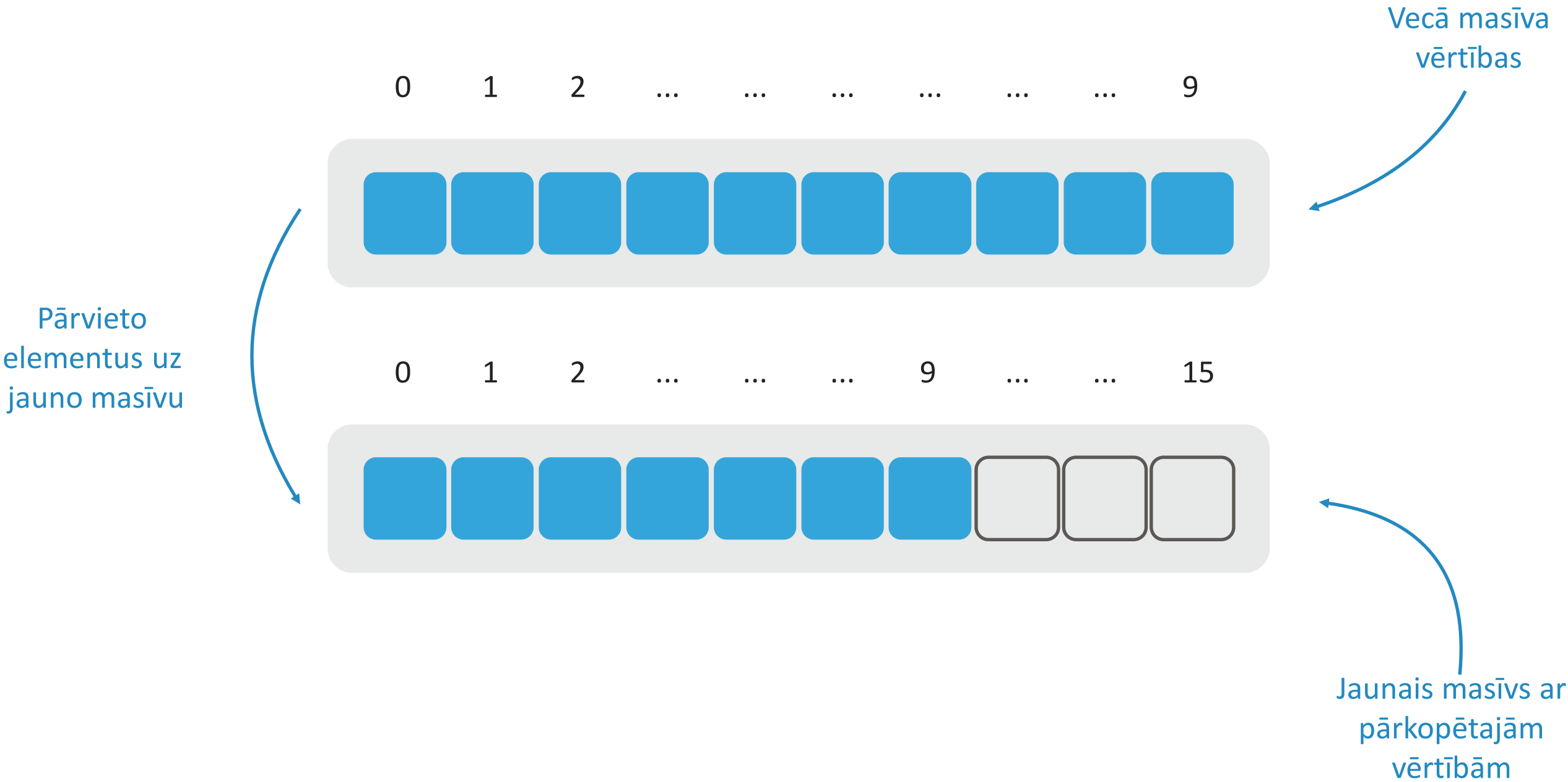
2. ARRAYLIST: INICIALIZĀCIJA



3. ARRAYLIST: PĀRSNIEDZ IZMĒRU



4. ARRAYLIST: KOPĒŠANA JAUNĀ MASĪVĀ



ARRAYLIST IZMĒRA APRĒĶINĀŠANA

Palielina izmēru par apmēram 50%



```
int newCapacity = (oldCapacity * 3)/2 + 1;
```


ARRAYLIST (NOKLUSĒTS IZMĒRS): PIEMĒRS

Pirmkods

```
List<String> scaryStories = new ArrayList<>();  
scaryStories.add("Your browser history is public");  
scaryStories.add("You didn't kill that spider");  
  
for (String story : scaryStories) { System.out.println(story); }
```

Konsules izvade

```
Your browser history is public  
You didn't kill that spider
```

ARRAYLIST (NORĀDĪTS IZMĒRU): PIEMĒRS

Pirmkods

```
List<String> scaryStories = new ArrayList<>(15);  
scaryStories.add("Your browser history is public");  
scaryStories.add("You didn't kill that spider");  
  
for (String story : scaryStories) { System.out.println(story); }
```

Konsules izvade

```
Your browser history is public  
You didn't kill that spider
```

ARRAYLIST RAKSTURLIELUMI

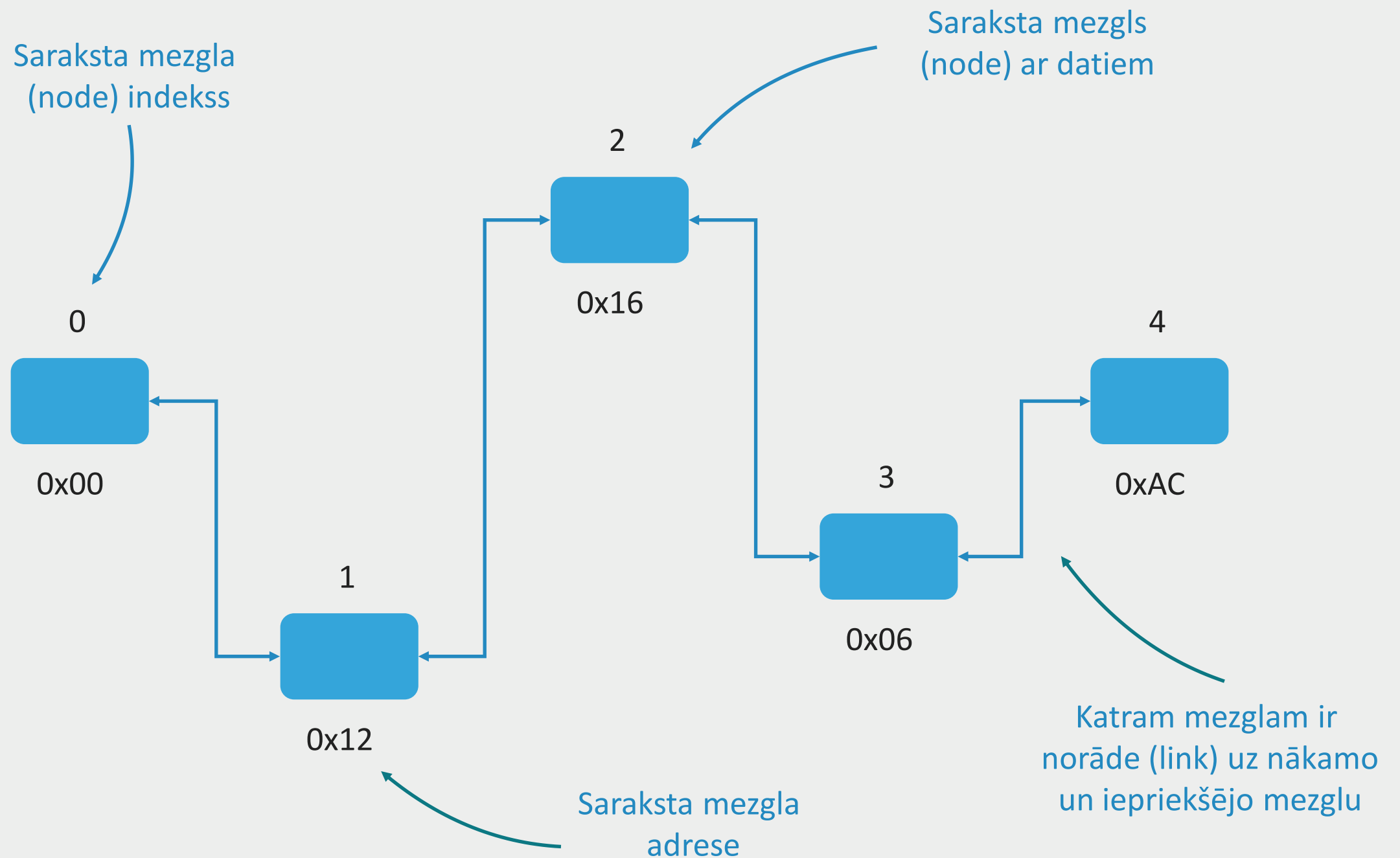
- ▶ ArrayList ir **maināma izmēra** masīvs, tiek saukts arī par dinamisko masīvu
- ▶ ArrayList izmanto **masīvu**, lai saglabātu elementus
- ▶ ArrayList ļauj glabāt **dublikātus** un **nulles** vērtības
- ▶ ArrayList ir **sakārtota** kolekcija
- ▶ ArrayList var saglabāt tikai **objektus** (primitīvas vērtības nevar)

JAVA SAISTĪTAIS SARAKSTS: LINKEDLIST

DUBULTI SAISTĪTA SARAKSTA IMPLEMENTĀCIJA SARAKSTAM UN TĀ INTERFEISIEM

Java Dokumentācija

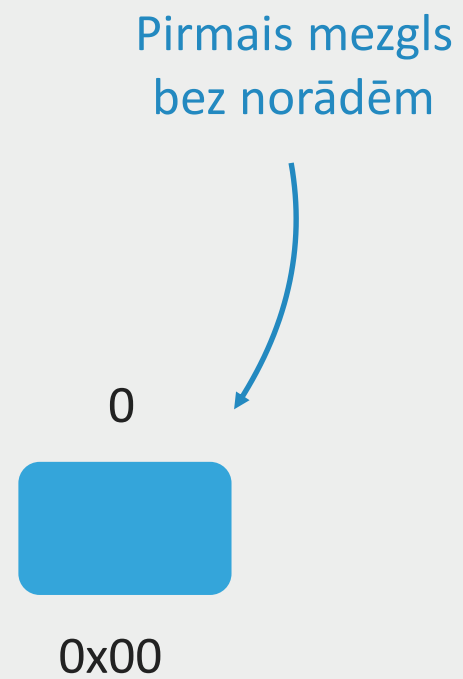
LINKEDLIST STRUKTŪRA



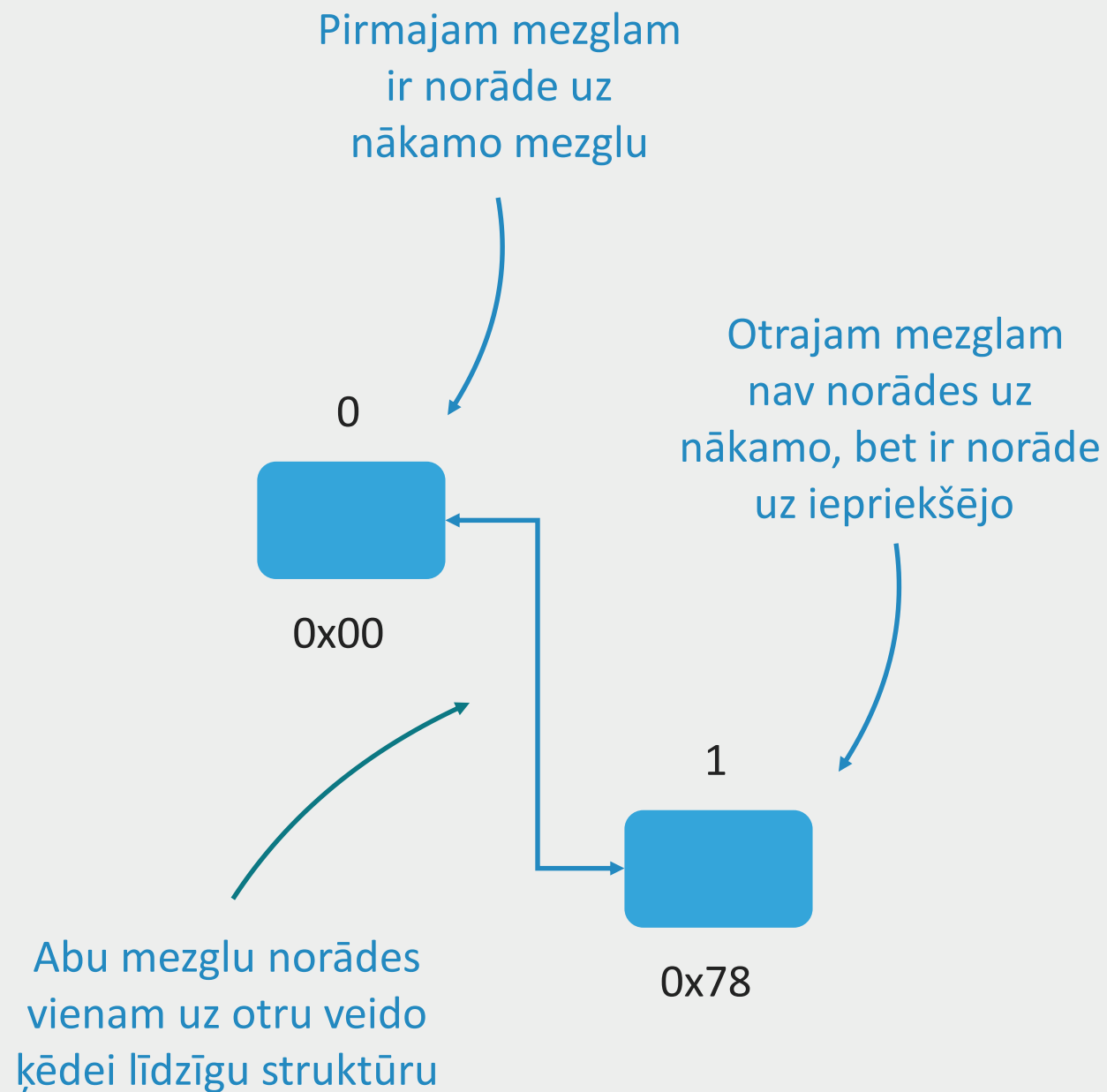
1. LINKEDLIST: INICIALIZĀCIJA

Pilnībā tukšs..

2. LINKEDLIST: PIEVIENO ELEMENTU



3. LINKEDLIST: PIEVIENO N-TO ELEMENTU



LINKEDLIST: PIEMĒRS

Pirmkods

```
List<String> things = new LinkedList<>();  
things.add("Computer");  
things.add("Coffee");  
  
for (String thing: things) { System.out.println(thing); }
```

Konsules izvade

```
Computer  
Coffee
```

LINKEDLIST RAKSTURLIELUMI

- ▶ Iekšēji izmanto **dažādus** objektus, kas **atsaucas** viens uz otru
- ▶ Ir **atļauti** dublikāti un null vērtības
- ▶ Tā ir **sakārtota** (ordered) kolekcija
- ▶ Tajā var glabāt tikai **objektus**



JAVA HASHMAP

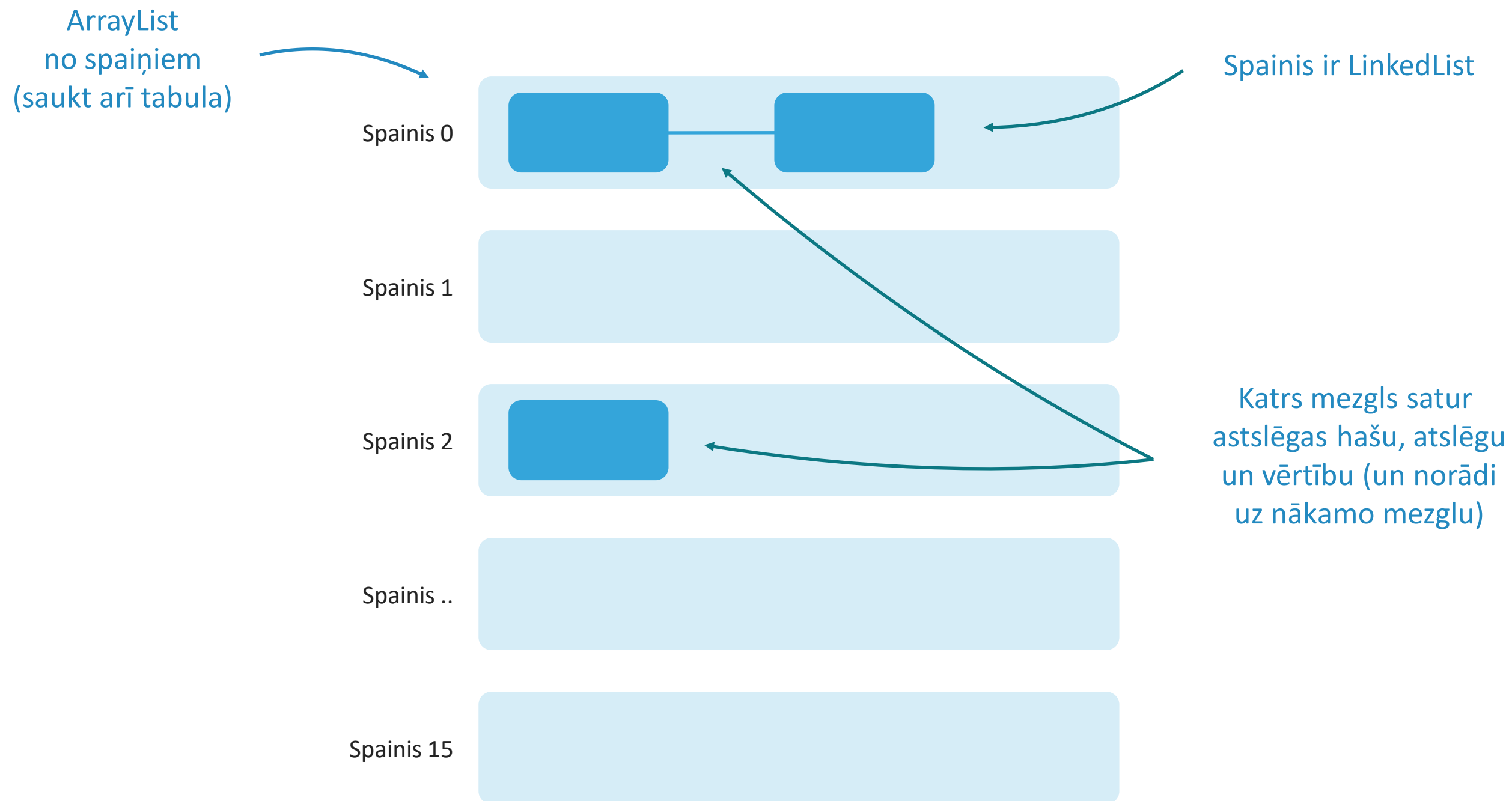
ARRAYLIST PRET LINKEDLIST

- ▶ Atmiņas patēriņš:
 - ▶ LinkedList **patērē vairāk atmiņas** nekā ArrayList, jo tajā tiek glabātas arī **nākamās** un **iepriekšējās atsauces** kopā ar datiem
- ▶ Piekļuve datiem:
 - ▶ ArrayList elementam var **piekļūt** $O(1)$ laikā (tieši pēc **indeksa**)
 - ▶ Lai piekļūtu saistītā saraksta elementam, nepieciešams $O(n)$ laiks (**nolasot katru elementu sarakstā, izmantojot atsauces**)
- ▶ Pievienošana un dzēšana:
 - ▶ ArrayList **parasti ir lēnāks**, jo, kad elements tiek pievienots vai noņemts vidū, tad visi atlikušie elementi ir **jāpārvieto** masīvā (svarīgas arī ir masīva izmēra izmaiņas)
 - ▶ LinkedList ir **ātrāks**, jo nepieciešams mainīt tikai norādes.

MAP INTERFEISA IMPLEMENTĀCIJA HAŠTABULAI

Java Dokumentācija

HASHMAP: DATU STRUKTŪRA



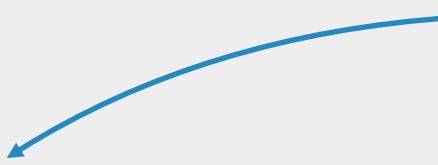
IEVADS HAŠOŠANĀ

- ▶ Hašfunkcija ir funkcija, kas atgriež **noteiktu vērtību**
- ▶ Katram argumentam tiek izveidots **unikāla hašvērtība**
- ▶ Izsaucot hašfunkciju **vairākas** reizes ar **vienu** un to pašu argumentu, vienmēr tiek atgriezta **viena hašvērtība**
- ▶ **Vienādi argumenti** atgriež **vienādas hašvērtības**
- ▶ Kad divi **dažādi** argumenti atgriež **vienādas hašvērtības**, tad tādu situāciju sauc par **kolīziju**
- ▶ Hašfunkcija darbojas **vienā virzienā**: oriģinālo argumenta vērtību **nevar** izskaitļot no hašvērtības

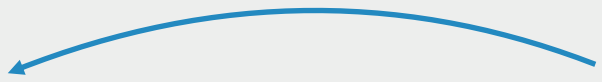
1. HASHCODE UN EQUALS: PIEMĒRS

```
public class Bag {  
  
    private String brand;  
    private String material;  
  
    public Bag(String brand, String material) {  
        this.brand = brand;  
        this.material = material;  
    }  
  
    @Override  
    public boolean equals(Object o) {  
        if (this == o) return true;  
        if (o == null || getClass() != o.getClass()) return false;  
        Bag bag = (Bag) o;  
        return Objects.equals(brand, bag.brand) &&  
            Objects.equals(material, bag.material);  
    }  
  
    @Override  
    public int hashCode() {  
        return Objects.hash(brand, material);  
    }  
}
```

Pārrakstam(Override) *equals*
metodi norādot, kurus laikus
salīdzināt



Pārrakstam(Override) *hashCode*
metodi norādot,
kurus laikus hašot



“HASHCODE” UN “EQUALS” LĪGUMS

- ▶ Izstrādātājiem ir jāpārraksta (override) abas metodes, lai sasniegtu pilnībā strādājošu vienādības mehānismu
- ▶ Ja divi objekti ir vienādi saskaņā ar equals() metodi, tad izsaucot metod hashCode() katram objektam ir jāiegūst vienādas hašvērtības

2. HASHCODE UN EQUALS: PIEMĒRS

Pirmkods

```
Bag mk = new Bag("Michael Kors", "suede");  
Bag gucci = new Bag("Gucci", "leather");  
  
System.out.println("Michael Kors = " + mk.hashCode());  
System.out.println("Gucci = " + gucci.hashCode());
```

Konsules izvade

```
Michael Kors = 1944981575  
Gucci = -2100362481
```

“HASHMAP” UN “EQUALS” LĪGUMA PIELIETOJUMS

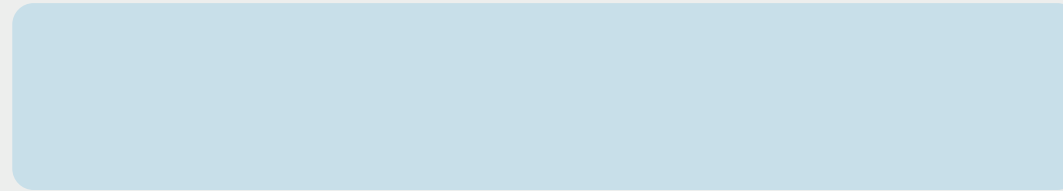
- ▶ Objekta **hašvērtība** ļauj algoritmiem objektus ievietot **nodalījumos**, kas paātrina to atrašanu
- ▶ Objekta **“equal”** metode ļauj algoritmam **atrast** pareizajā **nodalījumā** meklējamo objektu

1. HASHMAP: INICIALIZĀCIJA

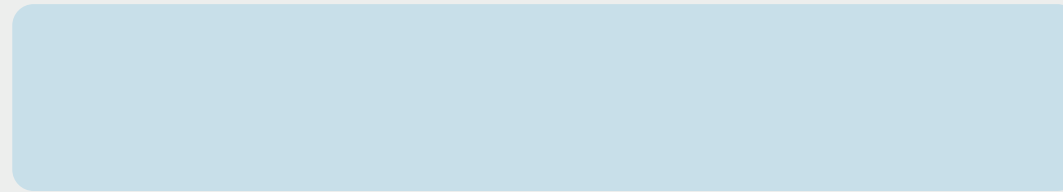
Izveido ArrayList
ar 16 tukšiem
spaiņiem



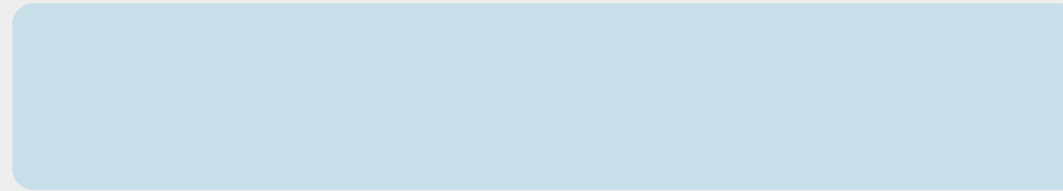
Spainis 0



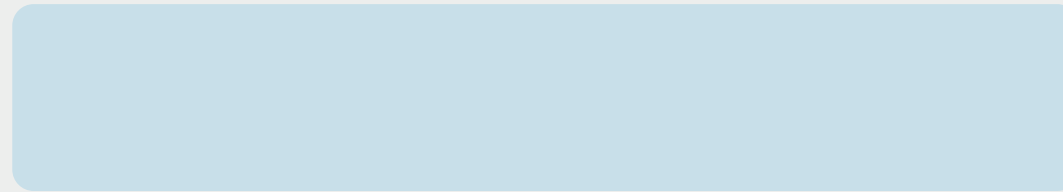
Spainis 1



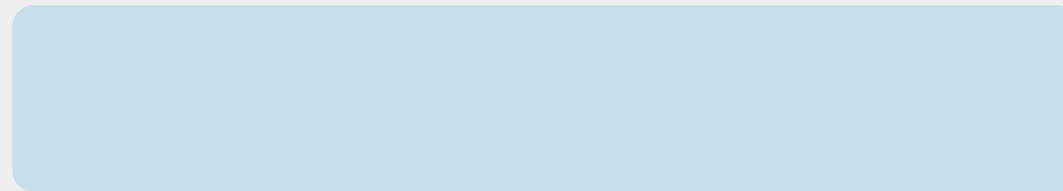
Spainis 2



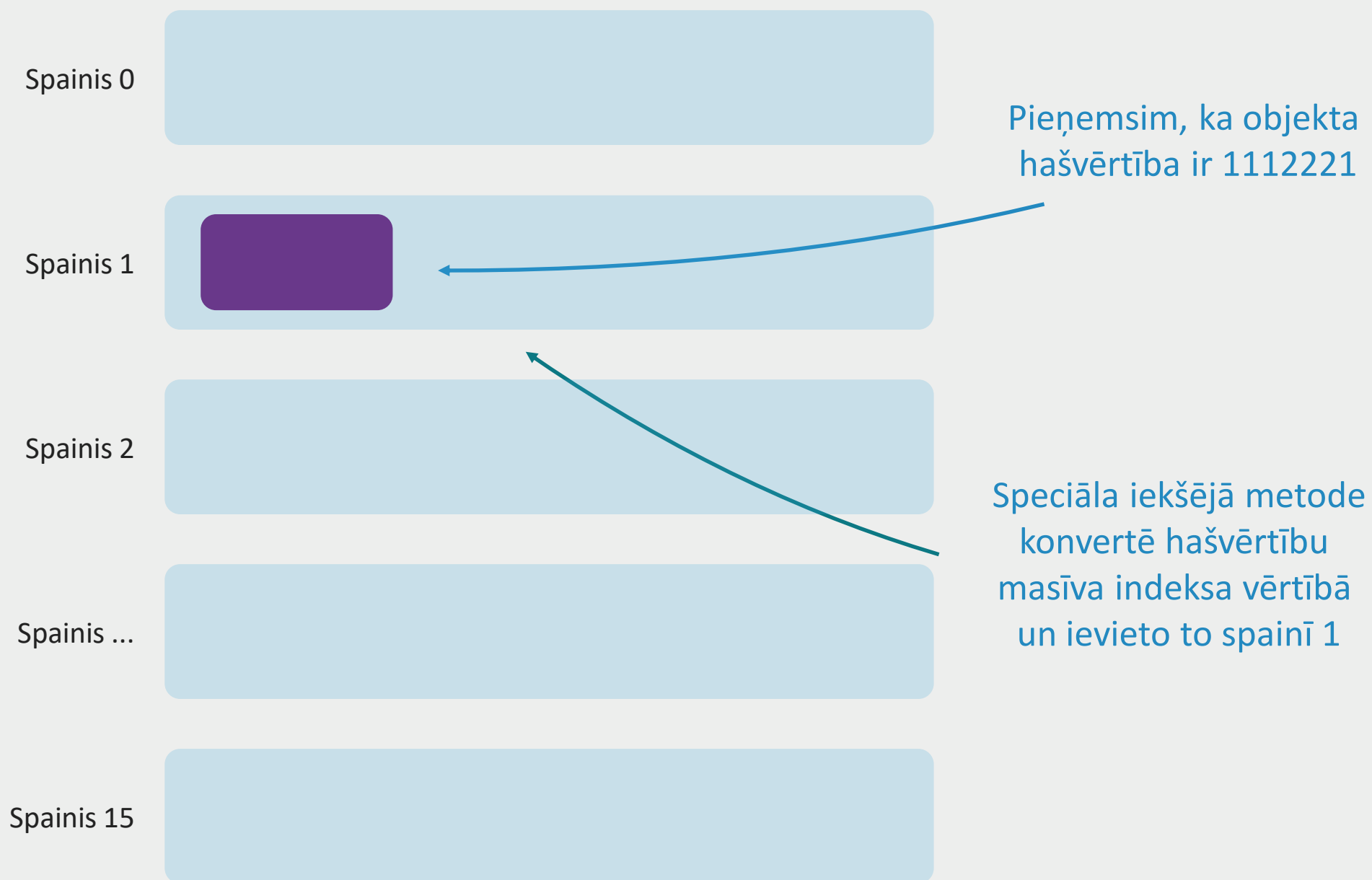
Spainis ...



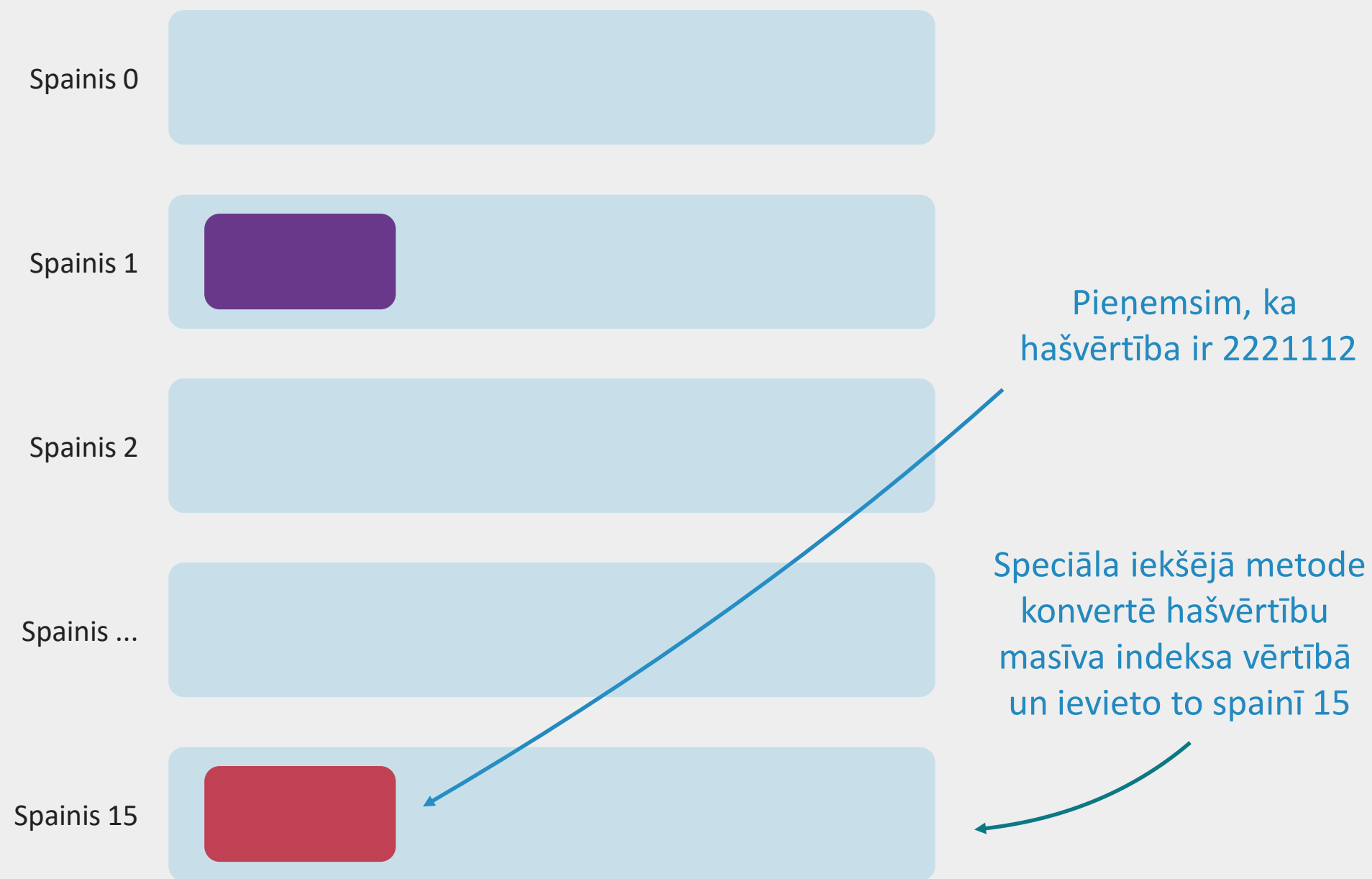
Spainis 15



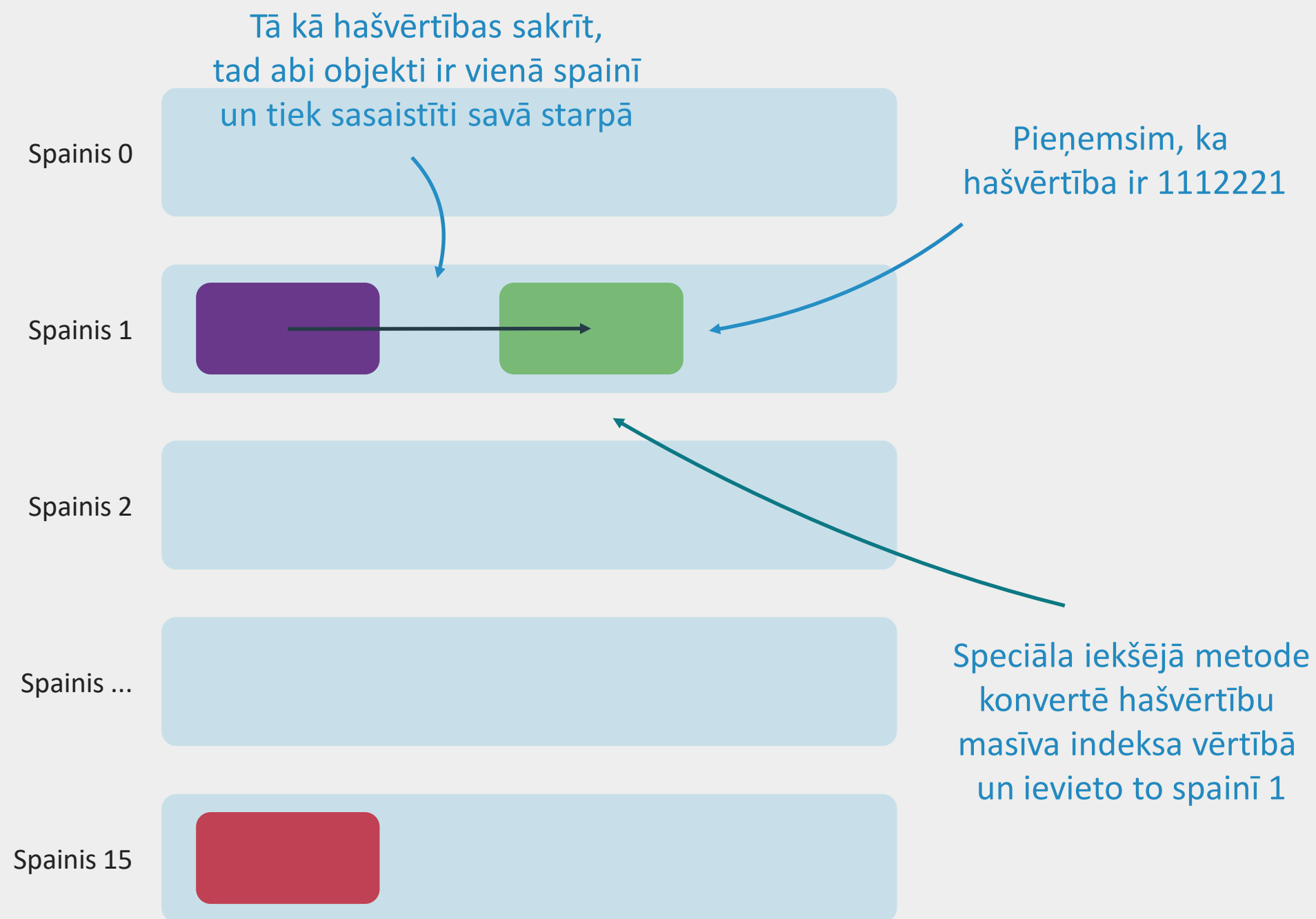
2. HASHMAP: PIEVIENO 1. ELEMENTU



3. HASHMAP: PIEVIENO 2. ELEMENTU



3. HASHMAP: PIEVIENO 3. ELEMENTU



HASHMAP: PIEMĒRS

Pirmkods

```
Map<String, Integer> tableOfContents = new HashMap<>();  
tableOfContents.put("Introduction", 3);  
tableOfContents.put("Chapter 1", 15);  
tableOfContents.put("Chapter 2", 48);  
  
System.out.println(tableOfContents);
```

Konsules izvade

```
{Introduction=3, Chapter 1=15, Chapter 2=48}
```

HASHMAP RAKSTURLIELUMI

- ▶ Iekšienē izmanto `ArrayList` (spaiņus kā elementus) un katrs spainis satur `LinkedList`
- ▶ Nav atļautas dublicējošas atslēgas
- ▶ Tiek atļauta viena null atslēga un daudz null vērtības
- ▶ Tā ir nesakārtota “kolekcija”
- ▶ Tā var glabāt tikai objektu vērtības



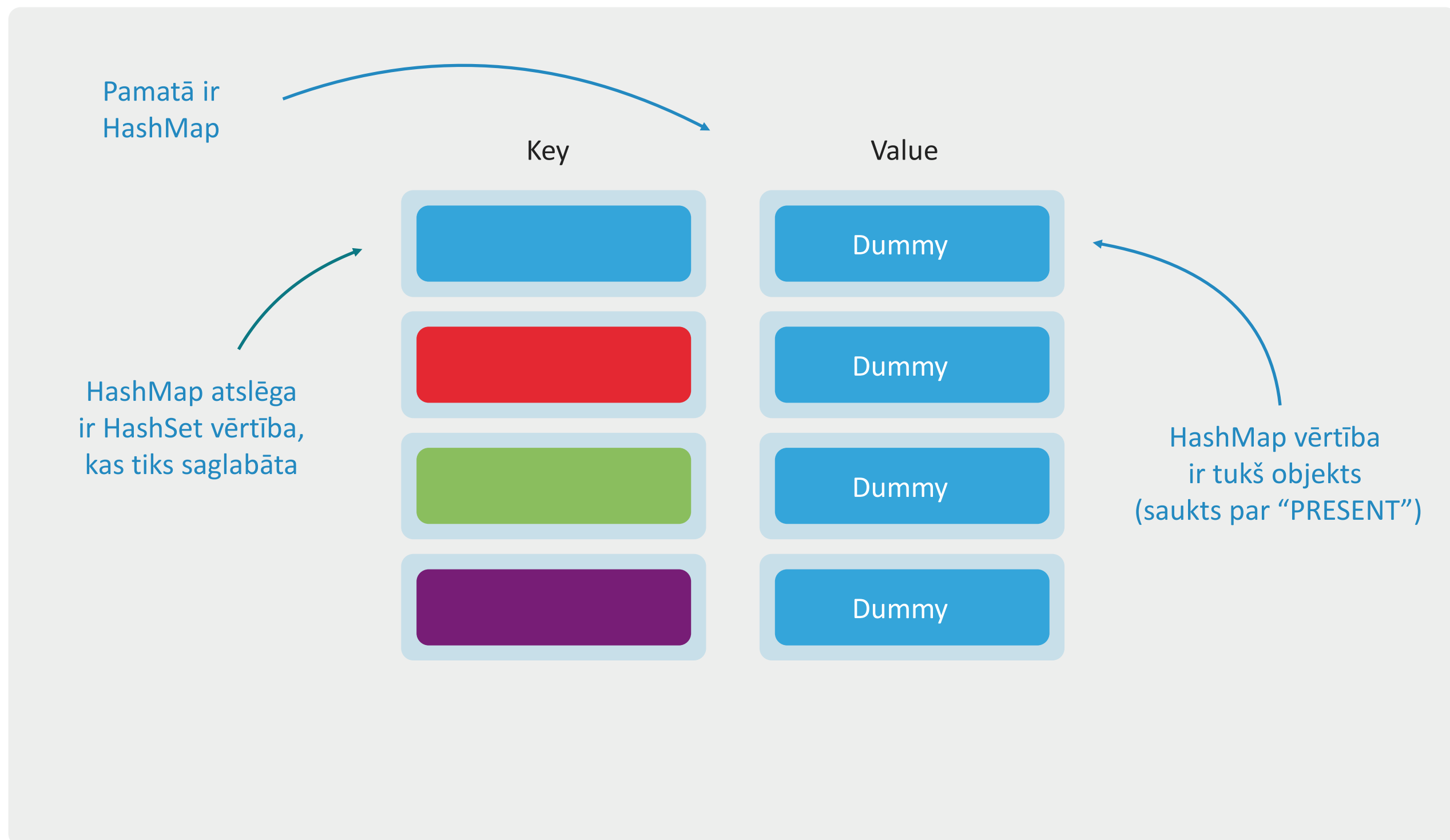
JAVA

HASHSET

KLASE IMPLEMENTĒ SET
INTERFEISU, KAS BALSTĪTS UZ
HAŠTABULAS

Java Dokumentācija

HASHMAP: IEKŠĒJĀ DATU STRUKTŪRA



HASHSET: PIEMĒRS

Pirmkods

```
Set<String> cities = new HashSet<>();  
cities.add("Rīga"); cities.add("Ogre"); cities.add("Rīga");  
  
System.out.println("cities = " + cities);
```

Konsules izvade

```
cities = [Rīga, Ogre]
```

HASHSET RAKSTURLIELUMI

- ▶ Iekšienē izmanto `HashMap` elementu glabāšanai
- ▶ Nav atļauti dublikāti
- ▶ Tiek atļauta viena null vērtība
- ▶ Tā ir nesakārtota “kolekcija”
- ▶ Tā var glabāt tikai objekta vērtību

ATSAUCES

- ▶ <https://www.callicoder.com/java-arraylist/>
- ▶ <https://www.callicoder.com/java-linkedlist/>
- ▶ <https://www.callicoder.com/java-hashmap/>
- ▶ <https://www.callicoder.com/java-hashset/>
- ▶ <https://netjs.blogspot.com/2015/08/how-arraylist-works-internally-in-java.html>
- ▶ <https://netjs.blogspot.com/2015/08/how-linked-list-class-works-internally-java.html>
- ▶ <https://netjs.blogspot.com/2015/05/how-hashmap-internally-works-in-java.html>
- ▶ <https://netjs.blogspot.com/2015/09/how-hashset-works-internally-in-java.html>



LATVIJAS
UNIVERSITĀTE
ANNO 1919



VADĪBAS UN
UZŅĒMĒJDARBĪBAS
MĀCĪBU CENTRS