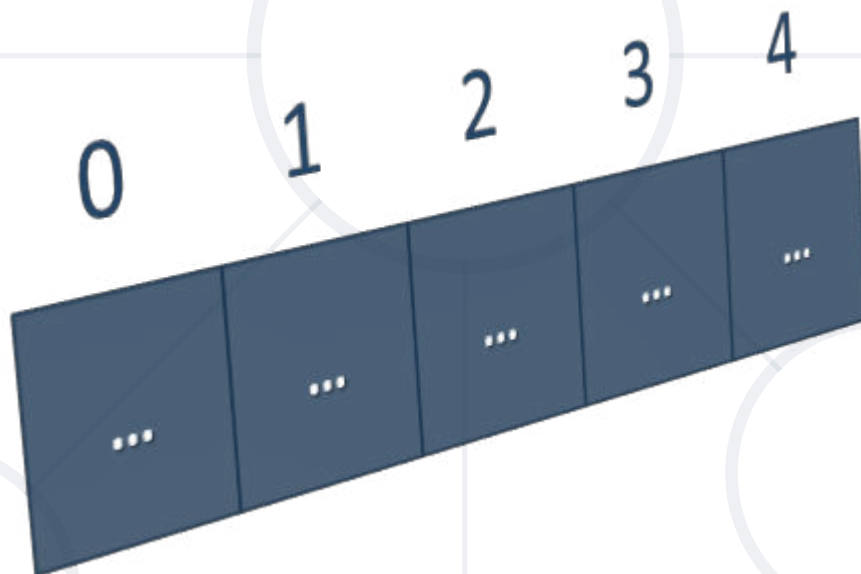


Lists

Processing Variable-Length Sequences of Elements



SoftUni Team
Technical Trainers



**Software
University**



**SoftUni
Foundation**



Software University

<http://softuni.bg>

Questions?

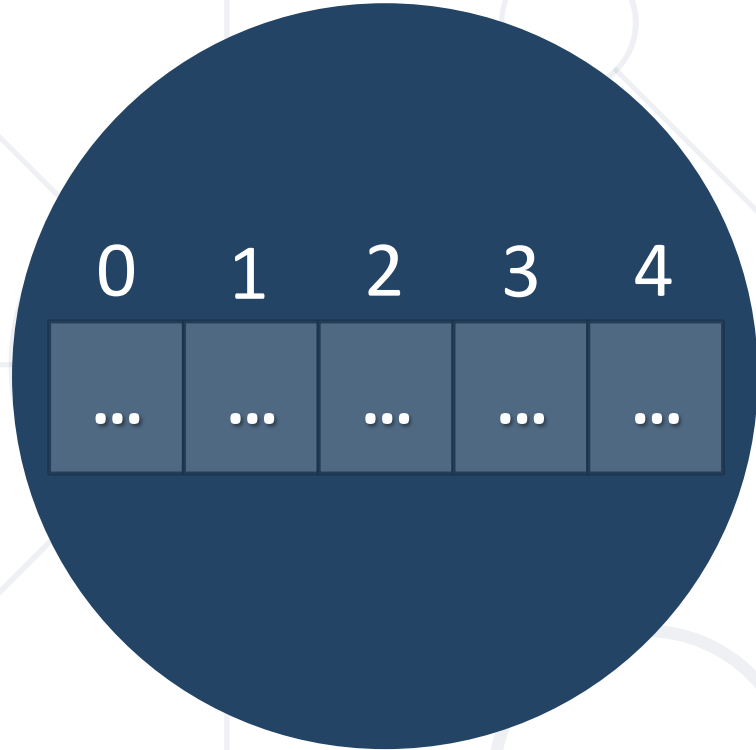
sli.do

#tech-java

Table of Contents

1. Lists Overview
2. List Manipulating
3. Reading Lists from the Console
4. Sorting Lists and Arrays



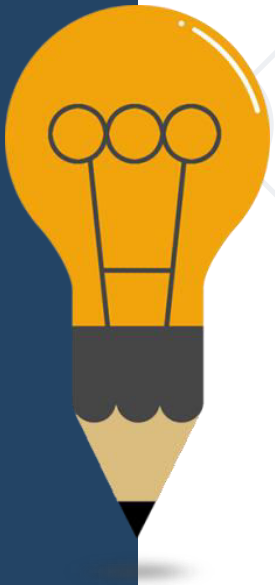


Lists


List<E> – Overview

- **List<E>** holds a list of elements of any type

```
List<String> names = new ArrayList<>();  
//Create a list of strings  
names.add("Peter");  
names.add("Maria");  
names.add("George");  
names.remove("Maria");  
for (String name : names)  
    System.out.println(name);  
//Peter, George
```



List<E> – Overview (2)



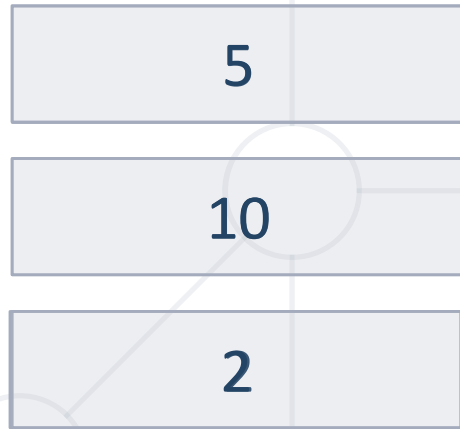
```
List<Integer> nums = new ArrayList<>(
    Arrays.asList(10, 20, 30, 40, 50, 60));
nums.remove(2); // Remove by index
nums.remove(Integer.valueOf(40)); // Remove by value
nums.add(100); // Inserts an element to index
nums.add(0, -100); // Items count
for (int i = 0; i < nums.size(); i++)
    System.out.print(nums.get(i) + " ");
```



-100 10 20 50 60 100

- **List<E>** holds a list of elements (like array, but extendable)
- Provides operations to **add** / **insert** / **remove** / **find** elements:
 - **size()** – number of elements in the List<E>
 - **add(element)** – adds an element to the List<E>
 - **add(index, element)** – inserts an element to given position
 - **remove(element)** – removes an element (returns true / false)
 - **remove(index)** – removes element at index
 - **contains(element)** – determines whether an element is in the list
 - **set(index, item)** – replaces the element at the given index

add – Appends an Element



List<Integer>

Count: 

remove – Deletes an Element

10

List<Integer>

2

10

5

Count:

3

add (index, el) – Inserts an Element at Position

-5

List<Integer>

2

5

Count:

2



Reading Lists from the Console

Using for Loop or String.split()

Reading Lists From the Console

- First, read from the console the array **length**:

```
Scanner sc = new Scanner(System.in);  
int n = Integer.parseInt(sc.nextLine());
```

- Next, create a list of given size **n** and read its **elements**:

```
List<Integer> list = new ArrayList<>();  
for (int i = 0; i < n; i++) {  
    int number = Integer.parseInt(sc.nextLine());  
    list.add(number);  
}
```

Reading List Values from a Single Line

- Lists can be read from a **single line** of **space separated values**:

```
2 8 30 25 40 72 -2 44 56
```

```
String values = sc.nextLine();  
List<String> items = Arrays.stream(values.split(" "))  
    .collect(Collectors.toList());  
List<Integer> nums = new ArrayList<>();  
for (int i = 0; i < items.size(); i++)  
    nums.add(Integer.parseInt(items.get(i)));
```

Convert a collection
into **List**

```
List<Integer> items = Arrays.stream(values.split(" "))  
    .map(Integer::parseInt).collect(Collectors.toList());
```

- Printing a list using a **for**-loop:

```
List<String> list = new ArrayList<>(Arrays.asList(
    "one", "two", "three", "four", "five", "six"));
for (int index = 0; index < list.size(); index++)
    System.out.printf
        ("arr[%d] = %s%n", index, list.get(index));
```

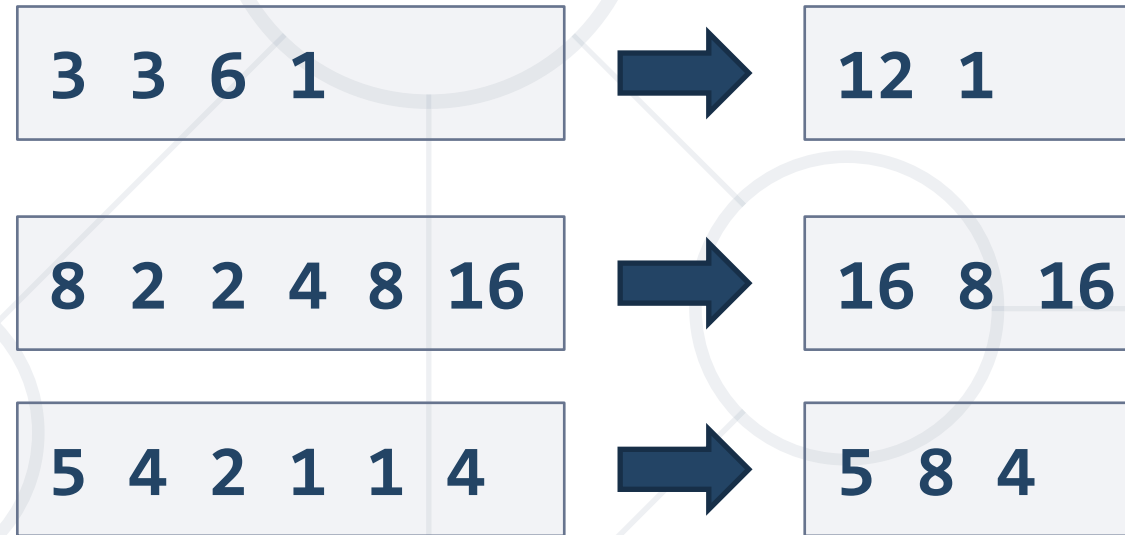
Gets an element
at given index

- Printing a list using a **String.join(...)**:

```
List<String> list = new ArrayList<>(Arrays.asList(
    "one", "two", "three", "four", "five", "six"));
System.out.println(String.join("; ", list));
```

Problem: Sum Adjacent Equal Numbers

- Write a program to sum all adjacent equal numbers in a list of decimal numbers, starting from left to right
- Examples:



Check your solution here: <https://judge.softuni.bg/Contests/1295/>

Solution: Sum Adjacent Equal Numbers (1)

```
Scanner sc = new Scanner(System.in);
List<Double> numbers = Arrays.stream(sc.nextLine().split(" "))
    .map(Double::parseDouble).collect(Collectors.toList());
for (int i = 0; i < numbers.size() - 1; i++)
    if (numbers.get(i).equals(numbers.get(i + 1))) {
        numbers.set(i, numbers.get(i) + numbers.get(i + 1));
        numbers.remove(i + 1);
        i = -1;
    }
//Continue on the next slide
```

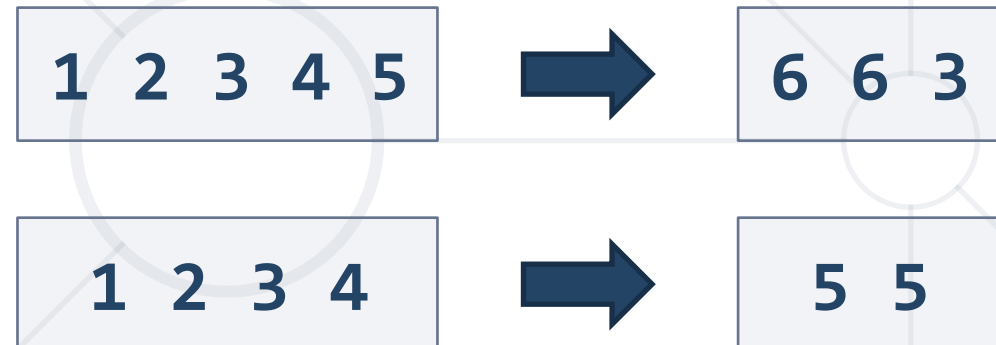
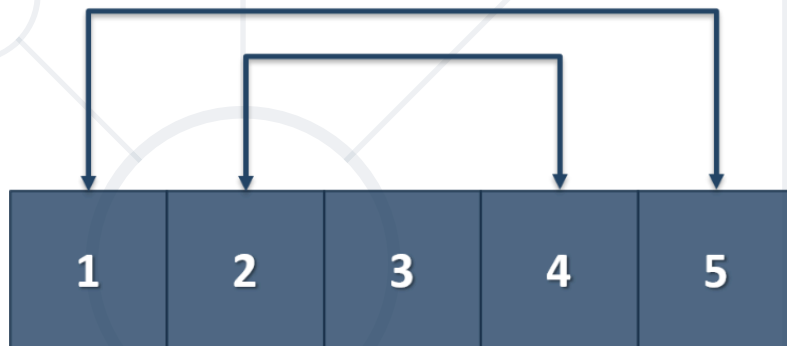

Solution: Sum Adjacent Equal Numbers (2)

```
String output = joinElementsByDelimiter(numbers, " ");  
System.out.println(output);
```

```
static String joinElementsByDelimiter  
    (List<Double> items, String delimiter) {  
    String output = "";  
    for (Double item : items)  
        output += (new DecimalFormat("0.#").format(item) + delimiter);  
    return output;  
}
```

Problem: Gauss' Trick

- Write a program that sum all numbers in a list in the following order:
 - $\text{first} + \text{last}$, $\text{first} + 1 + \text{last} - 1$, $\text{first} + 2 + \text{last} - 2$, ... $\text{first} + n$, $\text{last} - n$
- Examples:



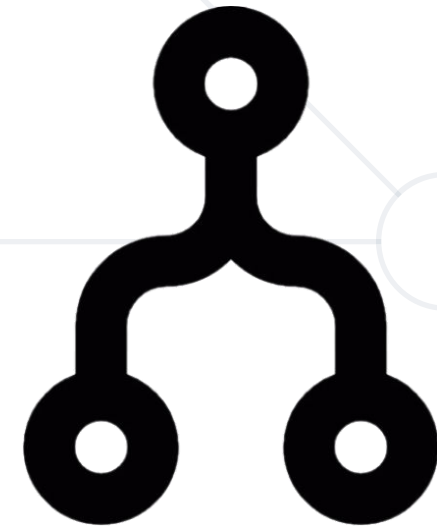
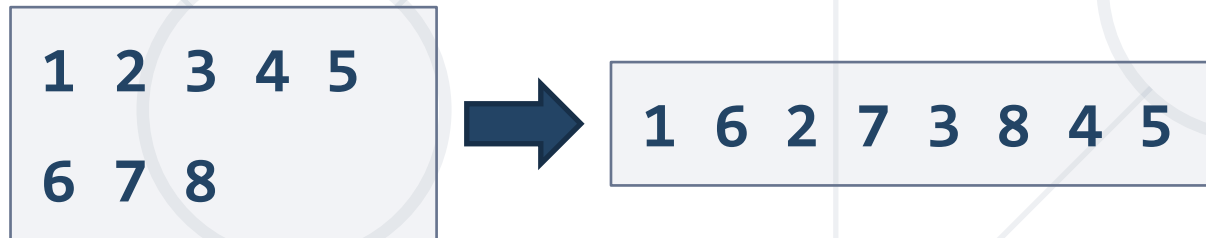
Check your solution here: <https://judge.softuni.bg/Contests/1295/>

Solution: Gauss' Trick

```
Scanner sc = new Scanner(System.in);  
List<Integer> numbers = Arrays.stream(sc.nextLine().split(" "))  
    .map(Integer::parseInt).collect(Collectors.toList());  
int size = numbers.size();  
for (int i = 0; i < size / 2; i++) {  
    numbers.set(i, numbers.get(i) + numbers.get(numbers.size() - 1));  
    numbers.remove(numbers.size() - 1);  
}  
System.out.println(numbers.toString().replaceAll("[\\[\\]", ""), "");
```

Problem: Merging Lists

- You receive two lists with numbers. Print a result list which contains the numbers from both of the lists
 - If the length of the two lists are not equal, just add the remaining elements at the end of the list:
 - `list1[0], list2[0], list1[1], list2[1], ...`



Check your solution here: <https://judge.softuni.bg/Contests/1295/>

Solution: Merging Lists (1)

//TODO: Read the input

```
List<Integer> resultNums = new ArrayList<>();
```

```
for (int i = 0; i < Math.min(nums1.size(), nums2.size()); i++) {
```

//TODO: Add numbers in resultNums

```
}
```

```
if (nums1.size() > nums2.size())
```

```
    resultNums.addAll(getRemainingElements(nums1, nums2));
```

```
else if (nums2.size() > nums1.size())
```

```
    resultNums.addAll(getRemainingElements(nums2, nums1));
```

```
System.out.println(resultNums.toString().replaceAll("[\\[\\]", ""), ""));
```

Solution: Merging Lists (2)

```
public static List<Integer> getRemainingElements  
    (List<Integer> longerList, List<Integer> shorterList) {  
    List<Integer> nums = new ArrayList<>();  
    for (int i = shorterList.size(); i < longerList.size(); i++)  
        nums.add(longerList.get(i));  
    return nums;  
}
```

Check your solution here: <https://judge.softuni.bg/Contests/1295/>



Live Exercises

Reading and Manipulating Lists



Sorting Lists and Arrays

- Sorting a list == reorder its elements incrementally: **Sort()**
 - List items should be **comparable**, e.g. numbers, strings, dates, ...

```
List<String> names = new ArrayList<>(Arrays.asList(
    "Peter", "Michael", "George", "Victor", "John"));
Collections.sort(names);
System.out.println(String.join(", ", names));
// George, John, Michael, Peter, Victor
Collections.sort(names);
Collections.reverse(names);
System.out.println(String.join(", ", names));
// Victor, Peter, Michael, John, George
```

Sort in natural
(ascending) order

Reverse the sorted result

Problem: List of Products

- Read a number **n** and **n** lines of products. Print a numbered list of all the products ordered by name
- Examples:

4
Potatoes
Tomatoes
Onions
Apples



1.Apples
2.Onions
3.Potatoes
4.Tomatoes



Check your solution here: <https://judge.softuni.bg/Contests/1295/>

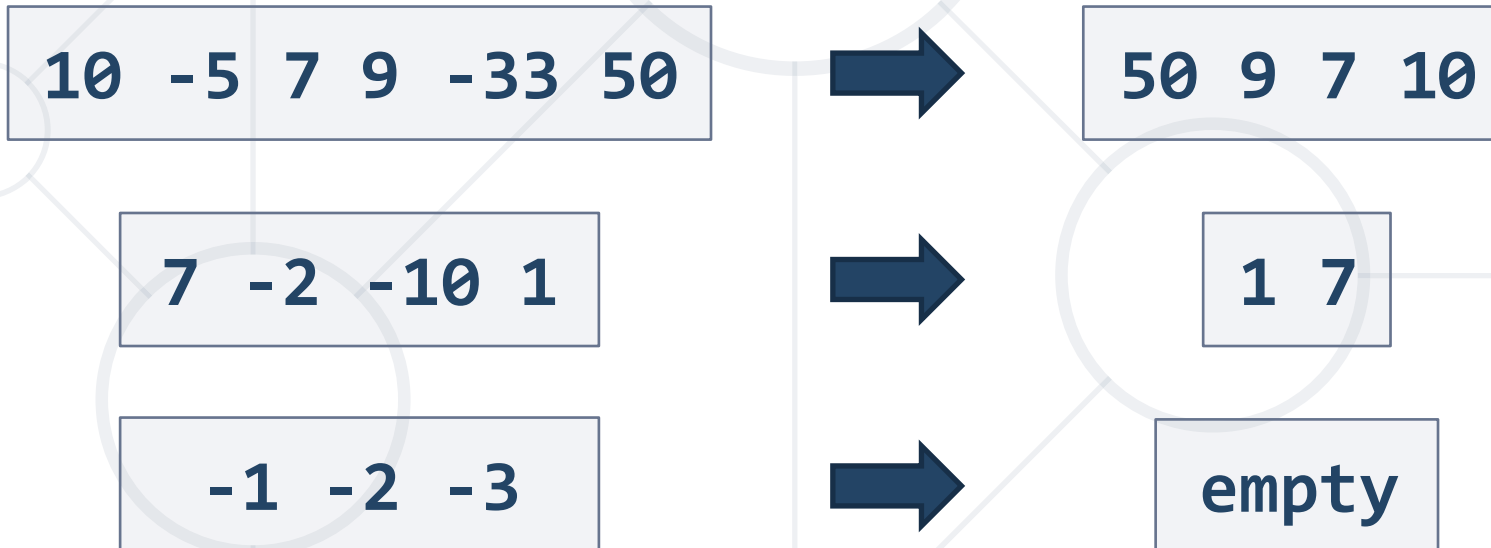
Solution: List of Products

```
int n = Integer.parseInt(sc.nextLine());
List<String> products = new ArrayList<>();
for (int i = 0; i < n; i++) {
    String currentProduct = sc.nextLine();
    products.add(currentProduct);
}
Collections.sort(products);
for (int i = 0; i < products.size(); i++)
    System.out.printf("%d.%s\n", i + 1, products.get(i));
```

Check your solution here: <https://judge.softuni.bg/Contests/1295/>

Problem: Remove Negatives and Reverse

- Read a list of integers, remove all negative numbers from it
 - Print the remaining elements in reversed order
 - In case of no elements left in the list, print "empty"



Check your solution here: <https://judge.softuni.bg/Contests/1295/>

Solution: Remove Negatives and Reverse

```
List<Integer> nums = Arrays.stream(sc.nextLine().split(" "))
    .map(Integer::parseInt).collect(Collectors.toList());
for (int i = 0; i < nums.size(); i++)
    if (nums.get(i) < 0)
        nums.remove(i--);
Collections.reverse(nums);
if (nums.size() == 0)
    System.out.println("empty");
else
    System.out.println(nums.toString().replaceAll("[\\[\\]", " "));
```



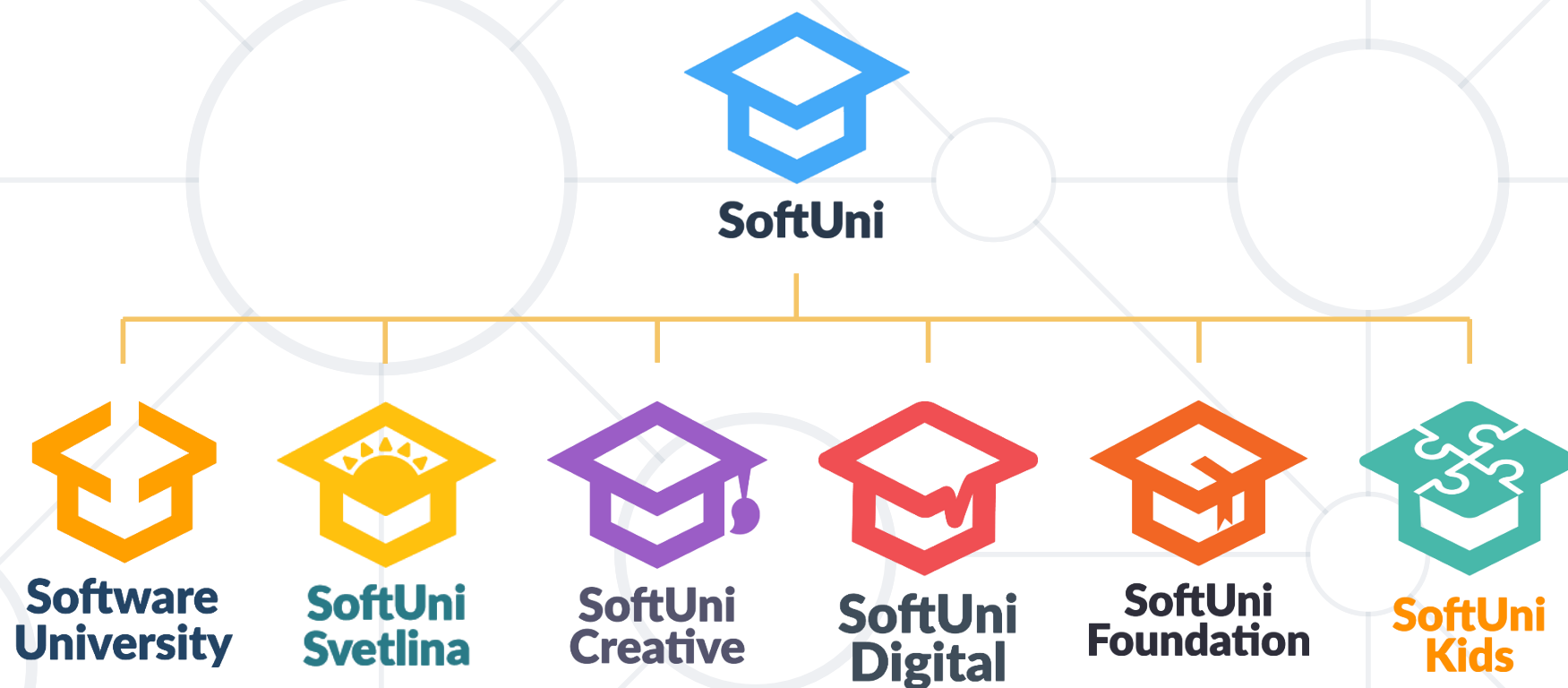
Live Exercises

Sorting Lists

- Lists hold a sequence of elements (variable-length)
- Can **add** / **remove** / **insert** elements at runtime
- Creating (allocating) a list:
`new ArrayList<E>()`
- Accessing list elements by index
- Printing list elements: **`String.join(...)`**



Questions?



SoftUni Diamond Partners



XSsoftware



SBTech
we know sports



telenor



SoftwareGroup
doing it right

NETPEAK



SmartIT



Postbank

Решения за твоето утре

**SUPER
HOSTING
.BG**

INDEAVR

Serving the high achievers



INFRAGISTICS®

LIEBHERR



aeternity



SoftUni Organizational Partners



OneBit
SOFTWARE



WORLD
OF
MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
 - softuni.bg
- Software University Foundation
 - <http://softuni.foundation/>
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University Forums
 - forum.softuni.bg



- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

