Working with Arrays

Software University

- **Allocating** an array of 10 integers:

```
int[] numbers = new int[10];
```

All elements are initially == 0

- **Assigning values** to the array elements:

```
for (int i = 0; i < numbers.length; i++)
  numbers[i] = 1;
```

The length holds the number of array elements

- **Accessing** array elements by index:

```
numbers[5] = numbers[2] + numbers[7];
numbers[10] = 1; // ArrayIndexOutOfBoundsException
```

The [] operator accesses elements by index

Reading Arrays from the Console

Software University

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

```
int n = Integer.parseInt(sc.nextLine());
```

- Next, create an array of given size **n** and read its **elements**:

```
int[] arr = new int[n];
for (int i = 0; i < n; i++) {
  arr[i] = Integer.parseInt(sc.nextLine());
}
```

Reading Array Values from a Single Line

- Arrays can be read from a **single line** of **separated values**

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

```
String values = sc.nextLine();
String[] items = values.split(" ");
int[] arr = new int[items.length];

for (int i = 0; i < items.length; i++)
  arr[i] = Integer.parseInt(items[i]);
```

Shorter: Reading Array from a Single Line

- Read an array of integers using functional programming:

```
String inputLine = sc.nextLine();
String[] items = inputLine.split(" ");
int[] arr = Arrays.stream(items)
  .mapToInt(e -> Integer.parseInt(e)).toArray();
```

import java.util.Arrays;

```
int[] arr = Arrays
 .stream(sc.nextLine().split(" "))
 .mapToInt(e -> Integer.parseInt(e)).toArray();
```

You can chain methods

Printing Arrays On the Console

Software University

- To print all array elements, a for-loop can be used
  - Separate elements with white space or a new line

```
String[] arr = {"one", "two"};
// == new String [] {"one", "two"};
// Process all array elements
for (int i = 0; i < arr.length; i++) {
   System.out.printf("arr[%d] = %s%n", i, arr[i]);
}
```

Printing Arrays with for / String.join(…)

Software University

- Use for-loop:

```
String[] arr = {"one", "two"};
for (int i = 0; i < arr.length; i++)
   System.out.println(arr[i]);
```

- Use `String.join(separator, array)`:

```
String[] strings = { "one", "two" };
System.out.println(String.join(" ", strings)); // one two
int[] arr = { 1, 2, 3 };
System.out.println(String.join(" ", arr)); // Compile error
```

Works only with strings

Solution: Reverse Array of Strings


```
String[] elements = sc.nextLine().split(" ");
for (int i = 0; i < elements.length / 2; i++) {
  String oldElement = elements[i];
  elements[i] = elements[elements.length - 1 - i];
  elements[elements.length - 1 - i] = oldElement;
}
System.out.println(String.join(" ", elements));
```

Check your solution here: https://judge.softuni.org/Contests/1248/

Foreach Loop

- Iterates through all elements in a collection
- Cannot access the current index
- **Read-only**

```
for (var item : collection) {
    // Process the value here
}
```

Print an Array with Foreach

```
int[] numbers = { 1, 2, 3, 4, 5 };
for (int number : numbers) {
    System.out.println(number + " ");
}
```