Welcome to the

Java Course

Module 2 – Day 02

Content of the course

- **Functions and procedures**
- Arrays and lists
- Search and sorting algorithms
- Data structures
- Computational complexity

Project Students - Step 6

```
How many students do you want to register? 3
>>> Student 1 <<<
Enter first name: Ana
Enter last name: Gaggero
Enter birthday (day of month): 22
Enter birth month: 10
Enter birth year: 1982
Enter course registered: Java
>>> Student 2 <<<
Enter first name: Carol
Enter last name: Muller
Enter birthday (day of month): 12
...
```

Project Students - Step 6

```
...
>>> Student 3 <<<
Enter first name: Tom
Enter last name: Grass
Enter birthday (day of month): 7
Enter birth month: 1
Enter birth year: 1980
Enter course registered: Java

List of registered students:
Ana Gaggero born the 22 of October 1982. Registered to Java
Carol Muller born the 12 of April 1990. Registered to Python
Tom Grass born the 7 of January 1980. Registered to Java
```

Project Students - Step 6

Modify the program:

- Split the code into methods (functions and procedures). At least one to convert the month from number to text, one to request each student's information and one to print the students information.

Arrays and Lists

Both used to store collections of elements but differ in:

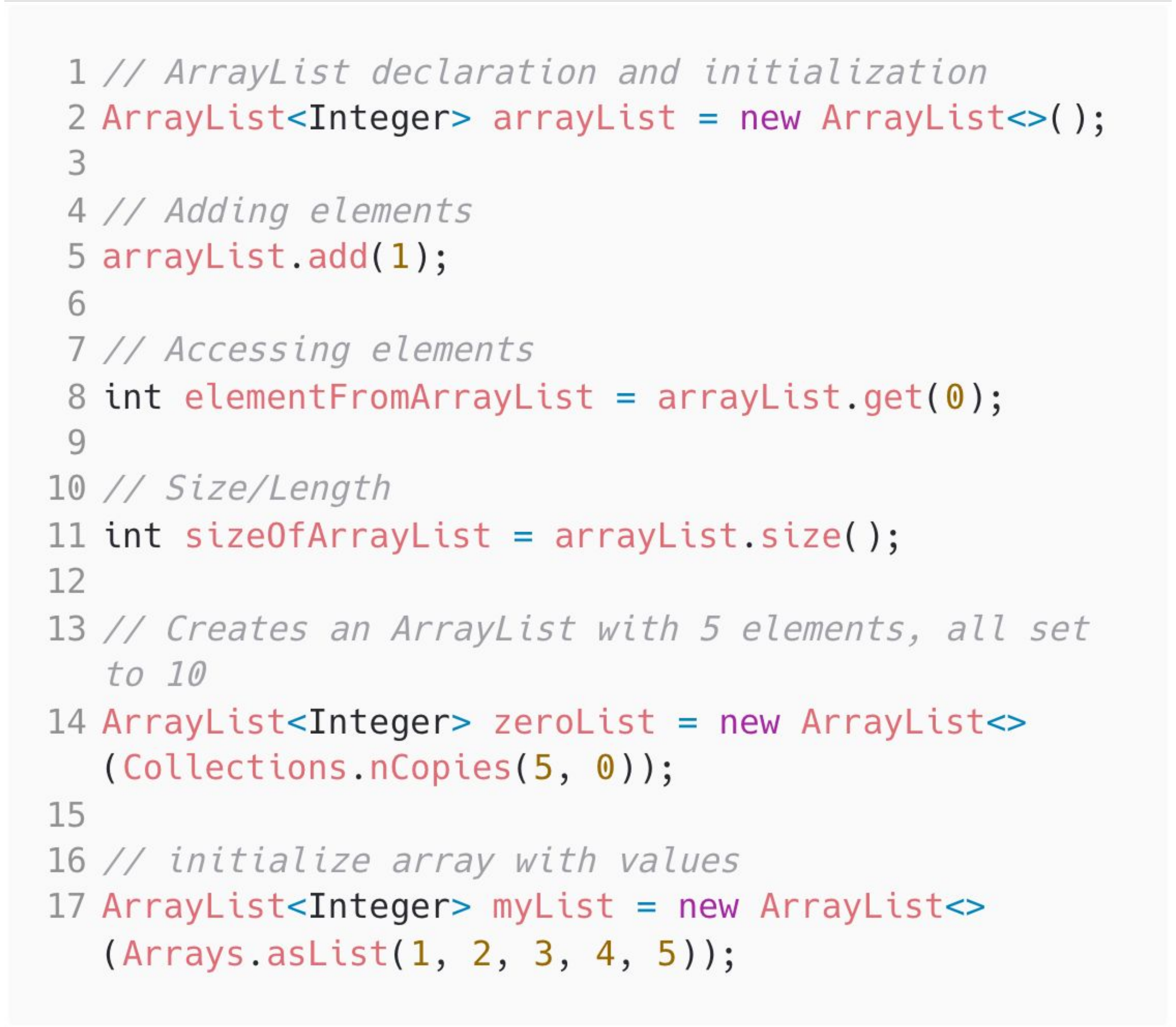
- Fixed Size vs. Dynamic Size
- Primitives vs. Objects
- Direct vs. Indirect Access
- Performance
- Length vs. Size

Arrays

```java
// Array declaration and initialization
int[] array = new int[5];

// Adding elements
array[0] = 1;

// Accessing elements
int elementFromArray = array[0];

// Size/Length
int lengthOfArray = array.length;

// create an array with all same values
int[] zeroArray = new int[5];
Arrays.fill(zeroArray, 0);

// initialize array with values
int[] myArray = {1, 2, 3, 4, 5};
```

Lists

ArrayList<>



```java
// ArrayList declaration and initialization
ArrayList<Integer> arrayList = new ArrayList<>();

// Adding elements
arrayList.add(1);

// Accessing elements
int elementFromArrayList = arrayList.get(0);

// Size/Length
int sizeOfArrayList = arrayList.size();

// Creates an ArrayList with 5 elements, all set to 10
ArrayList<Integer> zeroList = new ArrayList<>(Collections.nCopies(5, 0));

// initialize array with values
ArrayList<Integer> myList = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));
```

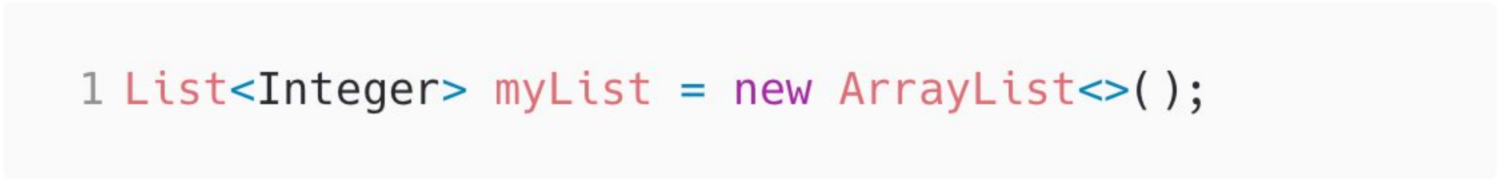
Polymorphic variables

Ability to hold different data types into a variable

```java
List<String> myList = new ArrayList<>();
```

Wrapper variables

- int → Integer
- char → Character
- Double → Double



```java
List<Integer> myList = new ArrayList<>();
```

Now YOUR TURN !

Let’s do exercises 1.1 and 1.2

Linear search

Linear search

Linear search is a simple search algorithm that checks every element in a list or array until the desired element is found or the list ends.

Now YOUR TURN !

Let’s do exercise 2

Sorting algorithms

Sorting can only be applied to items that can be compared to one another.

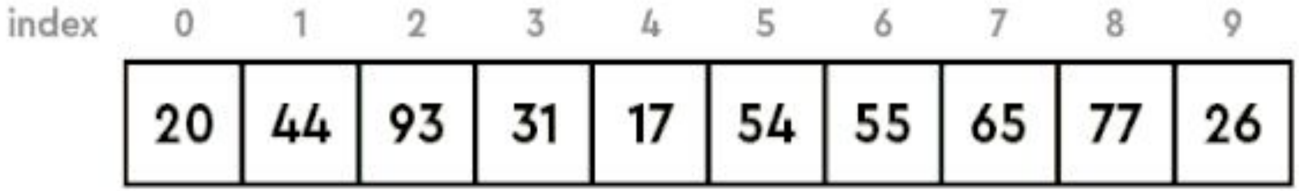
```
checklist = [56, 2, 0, -3, 123]
```
✓

```
checklist = ["k", "s", "a", "b", "y", "e"]
```
✓

```
checklist = ["k", 3, "n", -4, "y", 153, -9, "g"]
```
✗

Bubble sort

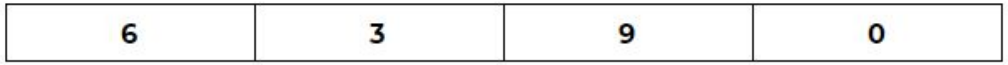
1. We start from the beginning of the list and compare each pair of adjacent elements.
2. If the elements are in the wrong order (i.e., the current element is greater than the next one), we swap them.
3. We repeat this process until no more swaps are needed, which means the list is sorted.



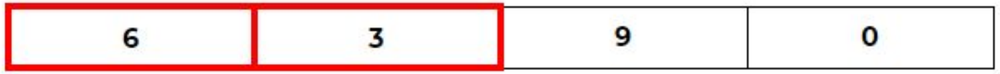
index	0	1	2	3	4	5	6	7	8	9
	20	44	93	31	17	54	55	65	77	26

Bubble sort (plan)

List given:

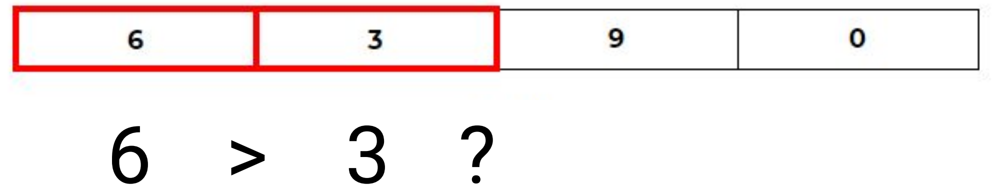


Select the first two items:

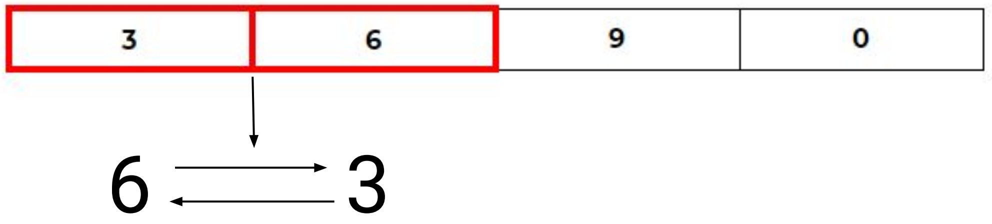


Bubble sort (plan)

Compare them to one another:

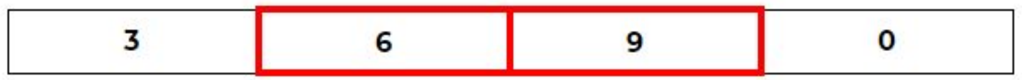


If the condition is true (6 is greater than 3), we swap the items:

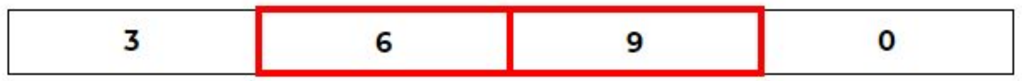


Bubble sort (plan)

Take the following two items:



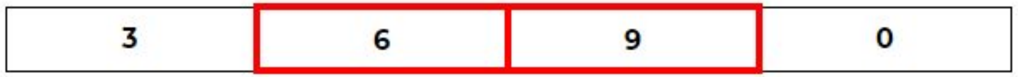
We also compare them to one another:



6 > 9 ?

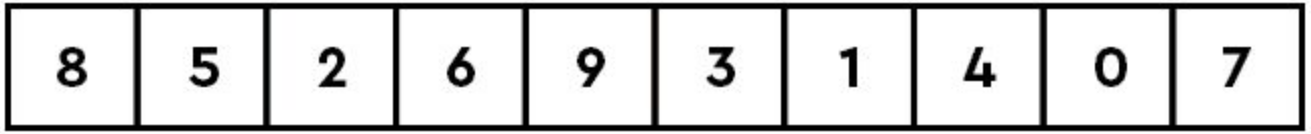
Bubble sort (plan)

The condition is false (6 is not greater than 9), which means the items remain in their places:



Selection sort

1. We select an item that by default is considered the smallest in the list
2. We compare it to the others. If there is a smaller item among them, we select it as the new smallest one and swap it with the previous one.



Selection sort (plan)

Item stored (position):

0

List given:

199	185	197	203

Store the first item:

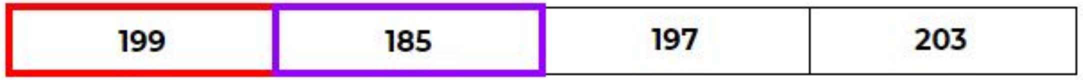
199	185	197	203

Selection sort (plan)

Item stored (position):

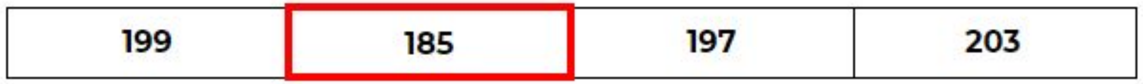
$\boxed{1}$

Compare it to the next item:



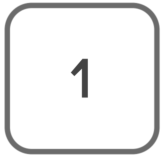
199 > 185 ?

If the condition is true (199 is greater than 185), remember the new item:

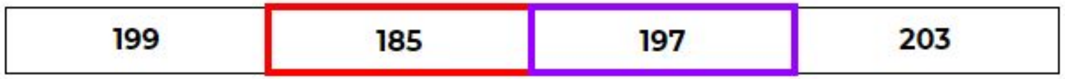


Selection sort (plan)

Item stored (position):



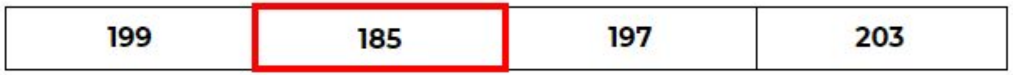
Compare it to the next item:



199	185	197	203

185 > 197 ?

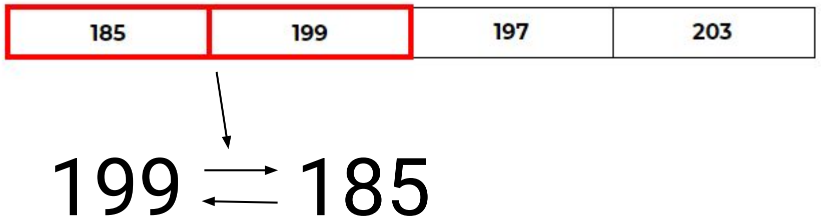
If the condition is false (185 is less than 199), the smallest item remains the same:



199	185	197	203

Selection sort (plan)

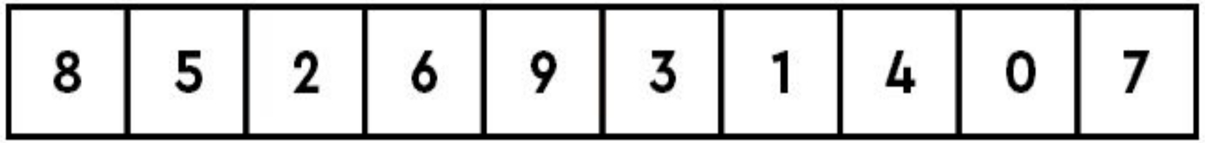
We compare all the remaining items in this way. If there is no smaller item, we put the smallest one we remembered at the top of the list. The item there goes to the place where the smallest was:



Insertion sort

1. Select the smallest list item. To make it easier let’s assume the very first item is the smallest.
2. We sort through all the remaining items and compare them to the selected one. If a smaller item is found, we put it at the top of the list, and move the other items one position forward.

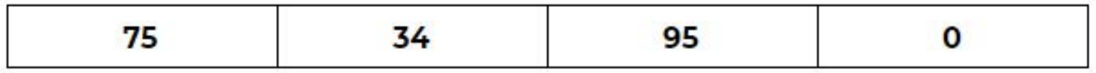
We repeat these steps until the list is sorted.



8	5	2	6	9	3	1	4	0	7

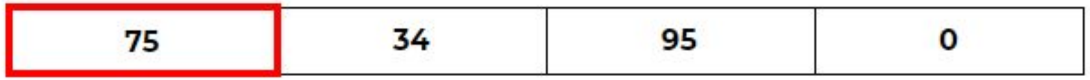
Insertion sort (plan)

List given:



75	34	95	0

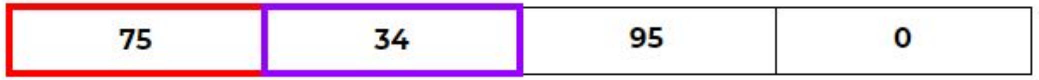
Remember the first item:



75	34	95	0

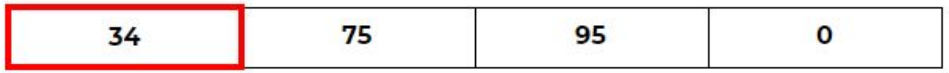
Insertion sort (plan)

Compare it to the next item:



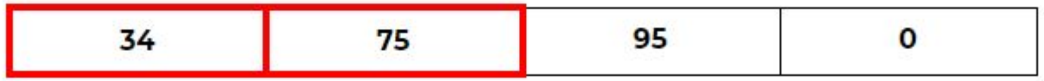
75 > 34 ?

If the condition is true (75 is greater than 34), we have found a smaller item. Put it at the top of the list and move all the other items forward 1 position:

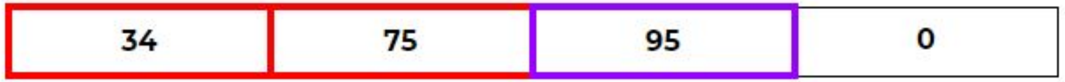


Insertion sort (plan)

Remember the first two items:



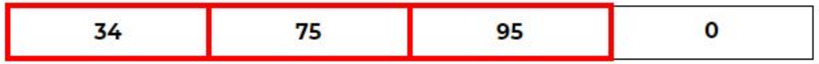
Compare them to the next item in order:



34 > 95 ? 75 > 95 ?

Insertion sort (plan)

If the item turns out to be less than one of those that we remembered, it goes in that one’s place, and all the remaining ones are shifted forward by one position. In our case, item 95 is greater than 75 and 34. So it remains in its place:



Now **YOUR TURN !**

Let’s do exercise 3

Streams

For complex data processing tasks such as

- Filtering
- Mapping
- Sorting

```
// Streams with arrays
 Integer[] numbers = {1, 2, 3, 4, 5, 6};
 Arrays.stream(numbers).filter(n -> n % 2 == 0).forEach(System.out::println);

 // Streams with Lists
 ArrayList<String> list = new ArrayList<>(Arrays.asList("java", "streams", "are", "cool"));
 list.stream().map(String::toUpperCase).forEach(System.out::println);
```

Streams on arrays

```
// Streams with arrays
Integer[] numbers = {1, 2, 3, 4, 5, 6};
```

Class wrapper

Stream converter

Array

```
Arrays.stream(numbers)
      .filter(n -> n % 2 == 0)
      .forEach(System.out::println);
```

Method reference

Streams on Lists, **deep dive**

```
// Streams with arrays
ArrayList<Integer> numbers = new ArrayList<>(Arrays.asList(1,2,3,4,5,6));
```

List

Stream converter

```
numbers.stream()
        .filter(n -> n % 2 == 0)
        .forEach(n -> System.out.println(n));
```

Now YOUR TURN !

Let’s do exercises number 4

Project Students - Step 7

Modify the program:

- To store the students in a List instead of all together in one String.

- Allow the user to search for a student by name

Project Students - Step 7

```
How many students do you want to register? 3
>>> Student 1 <<<
Enter first name: Ana
Enter last name: Gaggero
Enter birthday (day of month): 22
Enter birth month: 10
Enter birth year: 1982
Enter course registered: Java
>>> Student 2 <<<
Enter first name: Valerie
Enter last name: Muller
Enter birthday (day of month): 12
...
```

Project Students - Step 7

```
...
>>> Student 3 <<<
Enter first name: Tom
Enter last name: Grass
Enter birthday (day of month): 7
Enter birth month: 1
Enter birth year: 1980
Enter course registered: Java
```

Project Students - Step 7

```
Do you want to (a) see the list of students or (b) search for one
student? a

List of students:
Ana Gaggero born the 22 of October 1982. Registered to Java
Valerie Muller born the 12 of April 1990. Registered to Python
Tom Grass born the 7 of January 1980. Registered to Java

Do you want to (a) see the list of students or (b) search for one
student? b
Enter the student name: Tom Grass

Student:
Tom Grass born the 7 of January 1980. Registered to Java
```