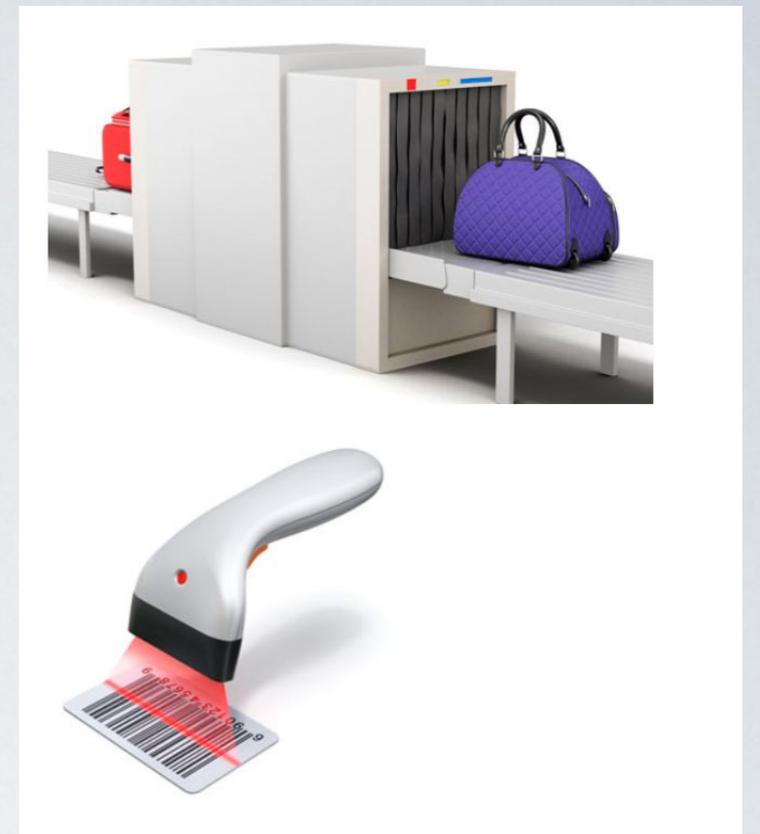


SCANNER & RANDOM

SCANNER

- `java.util.Scanner`
- Пользователь ввёл в консоли какое-то число. А программа должна считать с консоли, какое же число ввёл пользователь.
- Для работы с потоком ввода необходимо создать объект класса `Scanner`, при создании указав, с каким потоком ввода он будет связан. Стандартный поток ввода (клавиатура) в Java представлен объектом — `System.in`. А стандартный поток вывода (дисплей) — уже знакомым вам объектом `System.out`.



- Метод **nextInt()** считывает и возвращает введенное число.
- Метод **nextLine()** считывает и возвращает введенную строку.

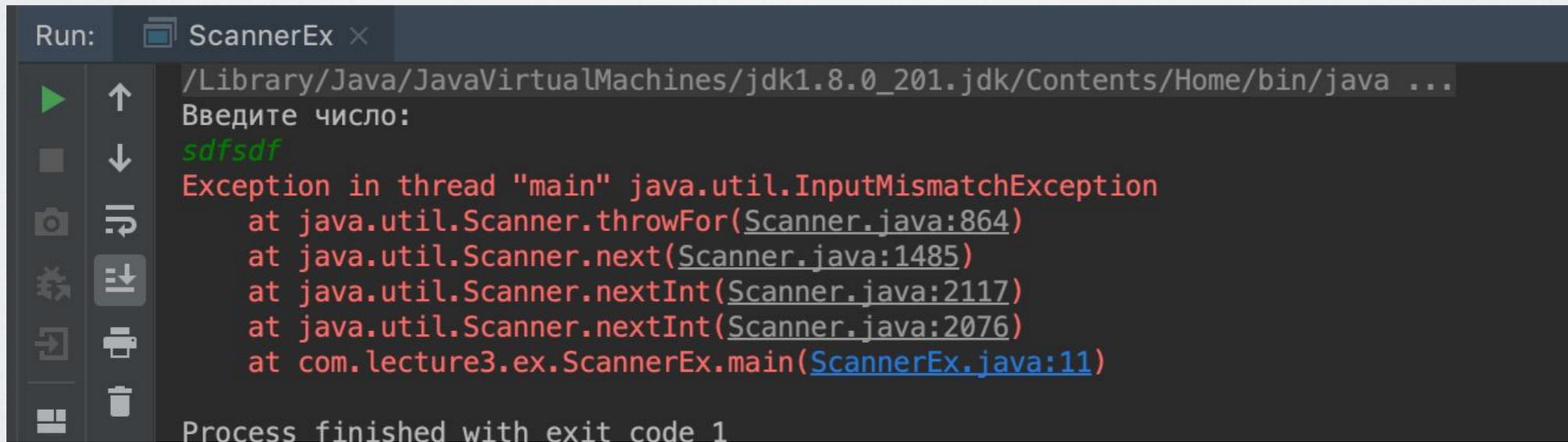
- **hasNextInt()** — метод проверяет, является ли следующая порция введенных данных числом, или нет (возвращает, соответственно, true или false).
- **hasNextLine()** — проверяет, является ли следующая порция данных строкой.
- **hasNextByte(), hasNextShort(), hasNextLong(), hasNextFloat(), hasNextDouble()** — все эти методы делают то же для остальных типов данных.

```
public class ScannerEx {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("Введите число:");

        int number = sc.nextInt();

        System.out.println("Спасибо! Вы ввели число " + number);
    }
}
```



The screenshot shows the Run console of an IDE. The title bar indicates the application is 'ScannerEx'. The console output shows the command to run the program, followed by the prompt 'Введите число:' and the user input 'sdfsdf'. This results in a 'java.util.InputMismatchException' being thrown. The stack trace shows the exception originates from the user's code in 'ScannerEx.java' at line 11, specifically during the 'nextInt()' call. The process ends with an exit code of 1.

```
Run: ScannerEx x
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java ...
Введите число:
sdfsdf
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
    at java.util.Scanner.nextInt(Scanner.java:2117)
    at java.util.Scanner.nextInt(Scanner.java:2076)
    at com.lecture3.ex.ScannerEx.main(ScannerEx.java:11)
Process finished with exit code 1
```

ЗАДАЧА

- Написать программу, которая проверяет, является ли следующий введенный символ числом или нет.
- Попросите пользователя ввести **2 любых целых числа**. После этого необходимо вывести в консоль сумму этих 2 чисел

RANDOM



- В **библиотеке классов Java** есть пакет `java.lang`, у которого есть класс `Math`, а у класса `Math` есть метод `random()`
- По умолчанию `Math.random()` генерирует случайные вещественные числа из промежутка $[0; 1)$, то есть от нуля включительно до 1 исключительно.

- Допустим, нам необходимо получить число с плавающей точкой в интервале $[0; 3)$ (**3** **ИСКЛЮЧИТЕЛЬНО**)
- 0 (от) + (`Math.random()` *(3 (до) - 0 (от)) \Rightarrow `Math.random()` * 3
- `(int)((Math.random() * (b - a + 1) + a)`

```
public class RandomEx {  
    public static void main(String[] args) {  
        double a = Math.random() * 3;  
  
        System.out.println(a);  
    }  
}
```

- $[0;2]$ - это значит, что диапазон **включает в себя число 2**.
- $\text{Math.random()} * ((2 - 0) + 1) + 0 =$
 $\text{Math.random()} * 3$

ЗАДАЧА

- Написать программу, которая генерирует целочисленное значение (Диапазон выбираете сами)
- Написать программу, которая генерирует число в диапазоне $[-100; +100)$

МАССИВЫ

Одномерные массивы

Алгоритмы сортировки массива

Алгоритмы поиска

Массив — это структура данных, в которой хранятся элементы одного типа.

Квадратные скобки

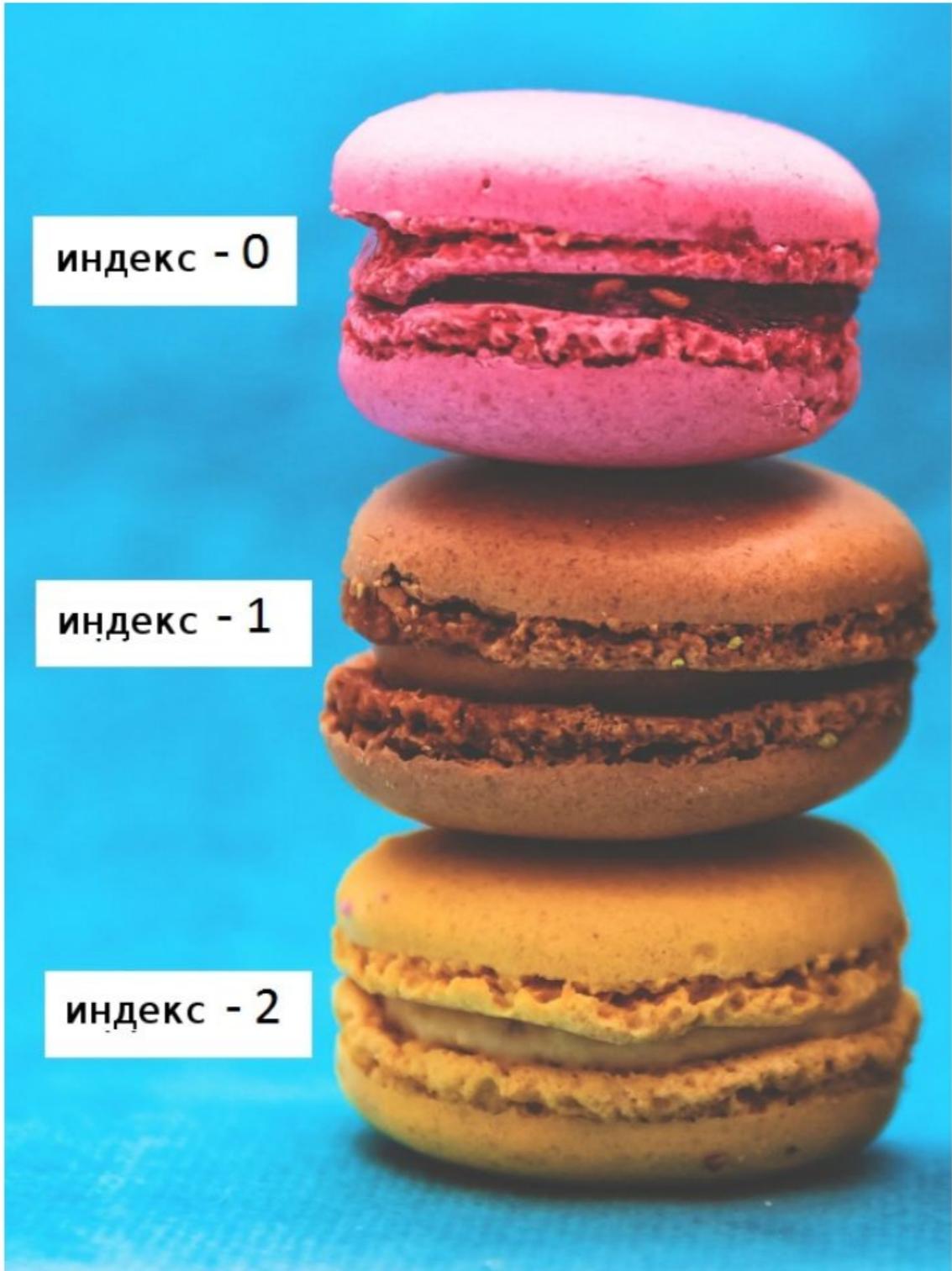
указывают,

Что это массив $\text{int arr} = \{1, 2, 3, 4, 5\};$

↑
Тип данных массивы

↑
0

Номер элемента в массиве также называют **ИНДЕКСОМ**



**массив из 3
макарон**

первый
макарон

второй
макарон

третий
макарон

ОБЪЯВЛЕНИЕ МАССИВА

- `dataType[] arrayName` (Желательно объявлять массив именно таким способом, это Java-стиль)
- `dataType arrayName[]` (Унаследованный от C/C++ способ объявления массивов, который работает и в Java)
- **`int[] myArray;`**

СОЗДАНИЕ МАССИВА

- Массив создается с помощью оператора new
`Double [] myList = new Double [10];`

```
Integer[] arr = {1, 2, 3, 4, 5};
```

JAVA.LANG.ARRAYINDEXOU TOFBOUNDSEXCEPTION

```
public class ArrayEx {  
    public static void main(String[] args) {  
  
        Double[] myList = new Double[10];  
        myList[0] = 5.6;  
        myList[1] = 4.5;  
        myList[2] = 3.3;  
        myList[3] = 13.2;  
        myList[4] = 4.00;  
        //...  
  
        myList[11] = 4.00;  
  
    }  
}
```

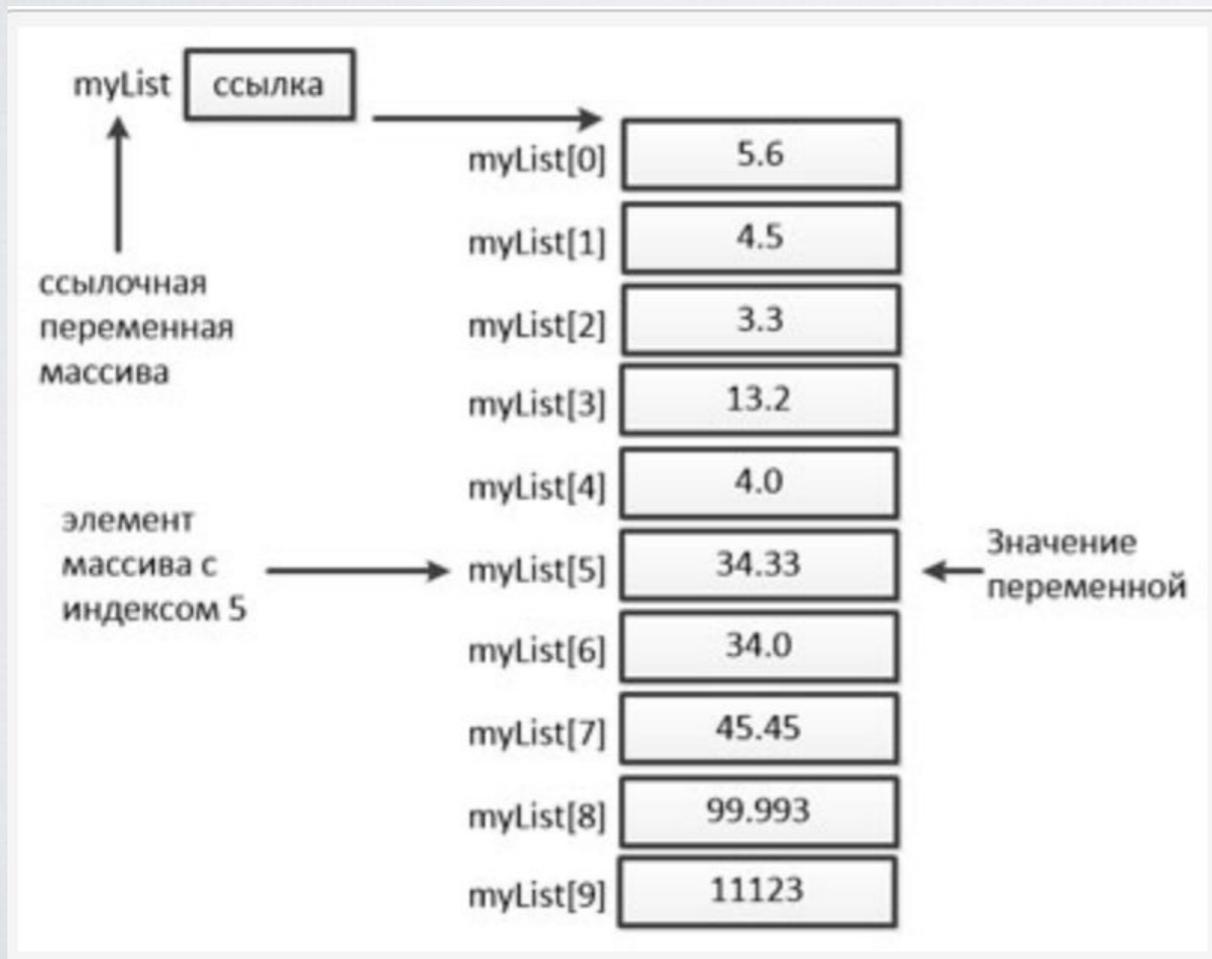
```
Run: ArrayEx x  
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java ...  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 11  
    at com.lecture3.example.ArrayEx.main(ArrayEx.java:14)
```

СОЗДАНИЕ МАССИВА

- После создания массива с помощью **new**, в его ячейках записаны значения по умолчанию. Для численных типов это будет 0, для **boolean** — **false**, для ссылочных типов — **null**.

- Длину массива нельзя изменить после его создания.
- Получить доступ к длине массива можно с помощью переменной **length**
- Массив *однороден*, то есть во всех его ячейках будут храниться элементы одного типа.
- В Java массивы являются объектами. Это значит, что имя, которое даётся каждому массиву, лишь указывает на адрес какого-то фрагмента данных в памяти.

- Инициализация массива — это заполнение его конкретными данными (не по умолчанию).
- # ИНИЦИАЛИЗАЦИЯ МАССИВА И ДОСТУП К ЕГО ЭЛЕМЕНТАМ



```
Double[] myList = new Double[10];  
myList[0] = 5.6;  
myList[1] = 4.5;  
myList[2] = 3.3;  
myList[3] = 13.2;  
myList[4] = 4.00;  
// ...
```

КАК ВЫВЕСТИ МАССИВ НА ЭКРАН?

```
3 public class ArrayPrintEx {
4     public static void main(String[] args) {
5         String[] seasons = {"Winter", "Spring", "Summer", "Autumn"};
6         for (int i = 0; i < 4; i++) {
7             System.out.println(seasons[i]);
8         }
9     }
10 }
11
```

ArrayPrintEx

ArrayPrintEx x

/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java ...

Winter
Spring
Summer
Autumn

```
System.out.println(Arrays.toString(arr));
```

ПОИСК МАКСИМАЛЬНОГО ЗНАЧЕНИЯ

```
public class MaxArrayEx {  
    public static void main(String[] args) {  
        int[] arr = {1, 2, 3, 111, 4, 5, 6, 71, 8, 9, 110};  
        int maxNumber = 0;  
        for (int i = 0; i < arr.length; i++) {  
            if (maxNumber <= arr[i]) {  
                maxNumber = arr[i];  
            }  
        }  
        System.out.println(maxNumber);  
    }  
}
```

ПОИСК МИНИМАЛЬНОГО ЗНАЧЕНИЯ

```
public class MinArrayEx {  
    public static void main(String[] args) {  
        int[] arr = {3, 2, 3, 111, 4, 0, 5, 6, 71, 8, 9, 110, 1};  
        int minNumber = arr[0];  
        for (int i = 0; i < arr.length; i++) {  
            if (arr[i] <= minNumber) {  
                minNumber = arr[i];  
            }  
        }  
        System.out.println(minNumber);  
    }  
}
```

ЗАДАЧИ

- Создайте массив, содержащий 10 первых нечетных чисел. Выведете элементы массива на консоль в одну строку, разделяя запятой.
- Найти среднее арифметическое всех элементов массива. Массив заполняете рандомно.
- Найти сумму всех элементов массива. Массив заполняете рандомно.
- Заполнить массив `short` типа числами от 100 до 0.
- * Переставить элементы массива в обратном порядке

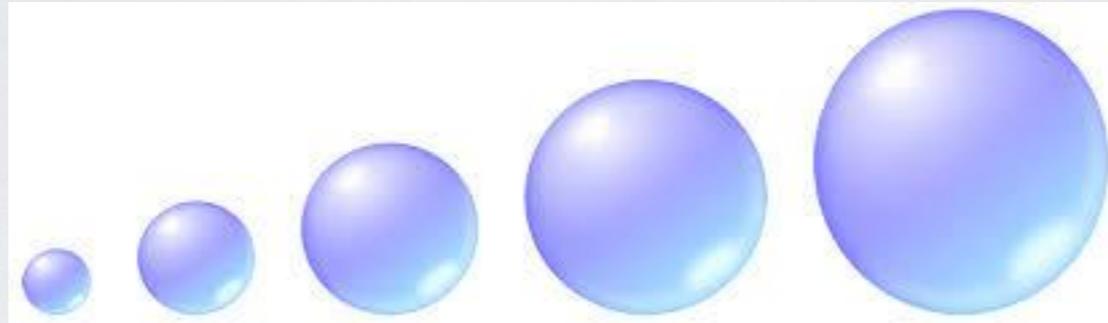
- **Алгоритм** — это последовательность действий, которая направлена на достижение окончательного решения проблемы наиболее оптимальными и эффективными способами.



ОЦЕНКА СЛОЖНОСТИ

- Сложность алгоритмов обычно оценивают по времени выполнения или по используемой памяти.
- $O(n)$ — линейная сложность
- $O(\log n)$ — логарифмическая сложность
- $O(n^2)$ — квадратичная сложность

СОРТИРОВКА ПУЗЫРЬКОМ



- Начиная с начала массива просматриваем попарно по 2 элемента (первый со вторым, второй с третьим, третий с четвертым и т.д.).
- Если второй элемент в паре меньше первого элемента – перемещаем его на место первого, а первый на место второго. Это мы делаем для всех элементов.
- После того, как мы дошли до конца массива (сравнили предпоследний и последний элементы и сделали обмен, если нужно), проверяем, был ли хотя бы один обмен. Если да, значит массив не отсортирован и начинаем все сначала.

```
public class BubbleSortEx {
    public static void main(String[] args) {
        int[] arr = {3, 5, 3, 6, 35, 3, 76};
        boolean isSorted = false;
        while (!isSorted) {
            isSorted = true;
            for (int i = 0; i < arr.length - 1; i++) {
                if (arr[i] > arr[i + 1]) {
                    int temp = arr[i];
                    arr[i] = arr[i + 1];
                    arr[i + 1] = temp;
                    isSorted = false;
                }
            }
        }
        System.out.println(Arrays.toString(arr));
    }
}
```

ЛИНЕЙНЫЙ ПОИСК

- Алгоритм ищет элемент в заданной структуре данных, пока не достигнет конца структуры.
- При нахождении элемента возвращается его позиция в структуре данных. Если элемент не найден, возвращаем -1.
- Для получения позиции искомого элемента перебирается набор из **N** элементов. В худшем сценарии для этого алгоритма искомый элемент оказывается последним в массиве - $O(n)$

```
public class LinearSearchEx {  
  
    public static void main(String[] args) {  
        int index = linearSearch(new int[]{89, 57, 91, 47, 95, 3, 27, 22, 67, 99}, elementToSearch: 89);  
        print(elementToSearch: 89, index);  
    }  
  
    private static int linearSearch(int arr[], int elementToSearch) {  
        for (int i = 0; i < arr.length; i++) {  
            if (arr[i] == elementToSearch)  
                return i;  
        }  
        return -1;  
    }  
  
    private static void print(int elementToSearch, int index) {  
        if (index == -1){  
            System.out.println(elementToSearch + " not found.");  
        }  
        else {  
            System.out.println(elementToSearch + " found at index: " + index);  
        }  
    }  
}
```