



# CodingBootcamp

Week 0 Day 2

Аннотация: Этот документ является практическим руководством к Week 0 Day 2 в CodingBootcamp.

Академия Ковалевского



# Содержание

1. Сортировка .....	2
2. Интерфейсы .....	3
3. Обобщения .....	4
4. Строки .....	5
5. Практическая работа .....	6





# 1. Сортировка

## Справка

### Определение

Сортировка — это любой процесс, который включает в себя упорядочивание данных в некотором значимом порядке, чтобы упростить понимание, анализ или визуализацию.

### Сортировка массива

Сортировка массива — это процесс распределения всех элементов массива в определенном порядке.

### Сортировка пузырьком

Сортировка пузырьком — это самый простой алгоритм сортировки. Он проходит по массиву несколько раз от начала и до конца, сравнивая попарно соседние элементы и на каждом этапе перемещает самое большое значение из неотсортированных в конец массива. В итоге, наименьший элемент постепенно перемещается к началу массива — “всплывает” до нужной позиции как пузырёк в воде.

## Рекомендации

- Найти информацию об алгоритмах сортировки в Java
- Посмотреть видео “[Сортировка пузырьком](#)”



## 2. Интерфейсы

### Справка

#### Определение

Интерфейс — это структура данных, которая может содержать поля, представленные в виде именованных констант и объявления методов. Интерфейсы определяют некоторый функционал, не имеющий конкретной реализации, который затем реализуют классы, применяющие эти интерфейсы. И один класс может реализовать множество интерфейсов.

#### Интерфейс `Comparator`

Рано или поздно появляется задача сравнения объектов по какому-либо принципу. Для этих целей существует интерфейс `Comparator`. Он содержит ряд методов, ключевым из которых является метод `compare(T o1, T o2)`, который должен быть реализован классом, реализующим компаратор. Этот метод возвращает числовое значение — если оно отрицательное, то объект `o1` предшествует объекту `o2`, иначе — наоборот. А если метод возвращает ноль, то объекты равны.

### Рекомендации

- Найти информацию об интерфейсах в Java
- Посмотреть видеоурок "[Интерфейсы в Java](#)"



## 3. Обобщения

### Справка

#### Определение

Обобщения или `generics` (обобщенные типы и методы) позволяют нам уйти от жесткого определения используемых типов. С их помощью можно объявлять классы, интерфейсы и методы, где тип данных указан в виде параметра. Обобщения — мощное дополнение к языку Java, поскольку оно облегчает работу программиста и снижает вероятность ошибок. Обобщения обеспечивают правильность типов во время компиляции и, что самое важное, позволяют реализовывать универсальные алгоритмы, не вызывая дополнительных затрат для приложений.

### Рекомендации

- Найти информацию об обобщениях
- Посмотреть видеоурок “[Что такое generic и какую проблему они решают](#)”
- Посмотреть видеоурок “[Знакомство с generic в Java](#)”



## 4. Строки

### Справка

#### Определение

Строка представляет собой последовательность символов. Для работы со строками в Java определен класс `String`, который предоставляет ряд методов для манипуляции строками. Физически объект `String` представляет собой ссылку на область в памяти, в которой размещены символы. При работе со строками важно понимать, что объект `String` является неизменяемым (immutable). То есть при любых операциях над строкой, которые изменяют эту строку, фактически будет создаваться новая строка.

#### Класс `StringBuilder`

Создание новых строк при операциях над строкой сказывается на производительности приложения. Для решения этой проблемы, чтобы работа со строками проходила с меньшими издержками в Java был добавлен класс `StringBuilder`. По сути, он напоминает расширяемую строку, которую можно изменять без ущерба для производительности. Но цена скорости - небезопасное поведение в многопоточной среде.

### Рекомендации

- Найти информацию классах `String` и `StringBuilder`
- Посмотреть видеоурок "[Составные строки в Java](#)"
- Посмотреть видеоурок "[Манипуляции символами](#)"



## 5. Практическая работа

### Класс StringTools

#### print

Написать функцию `println(Character[] target, Writer writer)`, которая выводит на экран переданный массив символов (`target`) с новой строки при помощи второго аргумента (`writer`).

Прототип функции:

```
1. public static void println(Character[] target, Writer writer);
```

#### reverse

Написать функцию `reverse(String target)`, которая принимает на вход строку и возвращает ее в обратном порядке (`"Hello world!"` → `"!dlrow olleH"`).

Прототип функции:

```
1. public static String reverse(String target);
```

#### toInt

Написать функцию `toInt(String target)`, которая принимает на вход строку и переводит ее в число — имитация работы `Integer.valueOf(target)`. Пример: `"456"` → `456`.

Прототип функции:

```
1. public static int toInt(String target);
```

### Класс Numbers

#### sort

Написать функцию `sort(int[] target)`, которая принимает на вход массив чисел (`int`) и сортирует его в порядке возрастания (ascending order).

Прототип функции:

```
1. public static void sort(int[] target);
```



## Класс Sorting

### sort

Написать функцию `sort(T[] target, Comparator<T> comparator)`, которая сортирует входящий массив в порядке возрастания (ascending order) с помощью компаратора, который передается вторым аргументом.

Прототип функции:

```
1. public static <T> void sort(T[] target, Comparator<T> comparator);
```

### sortReverseOrder

Написать функцию `sortReversedOrder(T[] target, Comparator<T> comparator)`, которая сортирует входящий массив в порядке понижения (descending order) с помощью компаратора, который передается вторым аргументом.

Прототип функции:

```
1. public static <T> void sortReversedOrder(T[] target, Comparator<T> comparator);
```

## ⚠ Требования

### Можно использовать

Классы: `java.util.Comparator`,  
`com.kovalevskiyi.academy.codingbootcamp.misc.Writer`.  
Структуры: `for`, `while`, `var`, `arrays`.

### Запрещено использовать

Любые методы, которые сортируют: `Arrays.sort`, `Collections.sort`.