

Аннотация: этот документ является практическим руководством к Coding Bootcamp.

Week 1 Day 1

**Академия Ковалевского**

**Содержание**

[**1. Таблицы кодировок ASCII**](#_bqgx0r9r876r) **3**

[**2. Управление символами в строке**](#_r1lb8nhtbwje) **3**

[**3. Практическая работа**](#_tk4ai0qsq3uu) **4**

[Пакет](#_6fx42t66hfqz) 4

[Класс StdString1](#_w18re4v4cqr6) 4

[StdString1(char[] base)](#_y3cdbf1626l1) 4

[StdString1()](#_65pl5x4f8kfa) 4

[tdString1(StdString stdString)](#_vqn6x6i3x4db) 5

[length](#_qkzkaevgaki8) 5

[append](#_z9rqwle18ycs) 5

[toCharArray](#_hgnva4njexx3) 5

[charAt](#_l1y5ymm6k5pf) 6

[indexOf](#_g4i7pxhm02a8) 6

[equals](#_n46byk6o429) 6

[hashCode](#_rhxokbqniztd) 6

[toString](#_b0anhxtxnwrr) 7

[iterator](#_yt0zqai0yinx) 7

[forEach](#_igcdyybwpr8n) 7

[Класс StringUtils](#_z729pcyrfn8x) 8

[isAsciiUppercase](#_megbm44lnq1a) 8

[isAsciiLowercase](#_xa9kmonsmuti) 8

[isAsciiNumeric](#_bedzxkik6vs1) 9

[isAsciiAlphabetic](#_ybrvl2knaabl) 9

[toAsciiUppercase](#_vvn3505ao7mq) 10

[toAsciiLowercase](#_fdtwazi3yn54) 10

[makeUppercase](#_meh9z1j5d1gp) 11

[makeLowercase](#_cux4ru14w9y1) 11

[makeCamel](#_skuff9ogruee) 12

[isStringAlphaNumerical](#_mi0m0ca9vgno) 12

[concatStrings](#_1ccbb65k3wl3) 13

[toInt](#_bk86imv0fqo8) 13

# 1. Таблицы кодировок ASCII

В вычислительных машинах символы не могут храниться иначе, как в виде последовательностей бит. Для передачи символа и его корректного отображения ему должна соответствовать уникальная последовательность нулей и единиц. Для этого были разработаны таблицы кодировок. ASCII представляет собой кодировку для представления десятичных цифр, латинского и национального алфавитов, знаков препинания и управляющих символов.

**Рекомендации**

Найти информацию о таблицах кодировок. Гуглить “ASCII symbol” или **man** ascii в терминале.

# 2. Управление символами в строке

Для **извлечения символов** по индексу в классе **String** определен метод ***charAt(int index)***. Он принимает индекс, по которому надо получить символ и возвращает извлеченный символ. Как и в массивах индексация начинается с нуля.

**Рекомендации**

* Найти [информацию](https://docs.oracle.com/javase/tutorial/java/data/manipstrings.html) об управлении символами в строке в Java
* Повторить пройденную информацию о классе StringBuilder

# 3. Практическая работа

## Пакет

Все классы этого задания должны быть в пакете:

|  |
| --- |
| package com.kovalevskyi.academy.codingbootcamp.week1.day1 |

**Запрещено** использовать любой импорт и класс String (если в описании метода не сказано иное).

## Класс StdString1

В этом задании мы начнем создавать свой собственный аналог класса **String**!

Учтите, что многие методы взаимосвязаны.

|  |
| --- |
| public class StdString1 implements Iterable<Character> {  *// TO DO* } |

### **StdString1(char[] base)**

Простой конструктор, который создает строку из входного массива символов.

|  |
| --- |
| *// сигнатура конструктора* public StdString1(char[] base){  *// TO DO* } |

### **StdString1()**

Пустой конструктор, который создает пустую строку длиной *0*.

|  |
| --- |
| *// сигнатура конструктора* public StdString1(){  *// TO DO* } |

### **tdString1(StdString stdString)**

Конструктор, который создает копию входящей строки. Новая строка должна быть создана из точно таких же символов, как и входящая.

|  |
| --- |
| *// сигнатура конструктора* public StdString1(StdString stdString) {  *// TO DO* } |

### **length**

Написать метод, который возвращает количество символов в строке. Например, “*cat* ” вернет *3*, “*cat in the box*” вернет *14*.

|  |
| --- |
| *// сигнатура метода* public int length() {  *// TO DO* } |

### **append**

Написать метод, который принимает строку, добавляет ее к текущей строке и возвращает софрмированную новою строку. Например, если есть строка “*cat*” и другая строка “*dog*”, ваш метод должен вернуть новую строку “*catdog*”. Если на вход передают *NULL,* кинуть *NullPointerException*.

|  |
| --- |
| *// сигнатура метода* public StdString append(StdString that) {  *// TO DO* } |

### **toCharArray**

Написать метод, который возвращает массив символов из текущей строки.

|  |
| --- |
| *// сигнатура метода* public char[] toCharArray() {  *// TO DO* } |

### **charAt**

Написать метод, который возвращает символ по определенному индексу. Например, в строке “*cat*”, на позиции “*0*” находится символ “*с*”, в строке “*dog*”, на позиции “*2*” находится символ “*g*”, а в строке “*dogcat*”, на позиции “*7*” не символа — *indexOutOfBoundsException*.

|  |
| --- |
| *// сигнатура метода* public char charAt(int index) {  *// TO DO* } |

### **indexOf**

Написать метод, который находит первый нужный символ в строке и возвращает его индекс. Если такого символа нет, вернуть *-1*. Например, в строке “*cat*” индекс символа “*t*” →  *2*, в строке “*dog*” индекс символа “*c*” → (*-1*), а в строке “*catanddog*” индекс символа “*a*” → *1*.

|  |
| --- |
| *// сигнатура метода* public int indexOf(char target) {  *// TO DO* } |

### **equals**

Написать метод, который сравнивает каждый символ текущей строки с каждым символом входящей строки. Вернуть *true,* если строки равные и *false,* если нет.

|  |
| --- |
| *// сигнатура метода*  @Override  public boolean equals(final Object otherObj) {   *// TO DO* } |
|  |

### **hashCode**

Написать метод, который возвращает хеш код строки. Хеш код подсчитывается путем суммирования целочисленных представлений каждого символа в строке.

“*Cat*” → *C: 67 a: 97 t: 116 → 67 + 97 + 116 = 280*. Если строка нулевой длины — вернуть *0*.

|  |
| --- |
| *// сигнатура метода*  @Override  public int hashCode() {   *// TO DO* } |

### **toString**

Написать метод, который конвертирует объект StdString в объект **String**. Тут позволено использовать *java.lang.String*.

|  |
| --- |
| *// сигнатура метода*  @Override  public String toString() {   *// TO DO* } |

### **iterator**

Написать метод ***iterator()***, который работает точно так же, как и оригинальный [***Iterator из документации***](https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html) (придется импортировать этот интерфейс и класс исключение).

|  |
| --- |
| *// сигнатура метода* @Override  public Iterator<Character> iterator() {   *// TO DO* } |

### **forEach**

Написать метод ***forEach()***, который проходит по каждому символу в строке и применяет к нему *action*. Тут тоже придется импортировать один интерфейс.

|  |
| --- |
| *// сигнатура метода*  @Override  public void forEach(final Consumer<? super Character> action) {   *// TO DO* } |

## Класс StringUtils

**Разрешено:** java.util.stream.IntStream.

**Запрещено:** методы: ***Integer.valueOf****,* классы: String (но можно использовать StringBuilder).

### **isAsciiUppercase**

Написать метод ***isAsciiUppercase(char ch)***, который принимает на вход **char** и возвращает *true*, если символ — это буква от *A* до *Z* (верхний регистр) и *false* — любой другой ASCII символ. Ожидается, что метод кинет *IllegalArgumentException*, если входящий символ не является ASCII символом.

|  |
| --- |
| *// сигнатура метода* public static boolean isAsciiUppercase(char ch) {  *// TO DO* } |

**Пример:**

|  |
| --- |
| isAsciiUppercase('A'); // → true; isAsciiUppercase('d'); // → false; isAsciiUppercase((char)257); // → IllegalArgumentException |

### **isAsciiLowercase**

Написать метод ***isAsciiLowercase(char ch)***, который принимает на вход **char** и возвращает *true*, если символ — это буква от *a* до *z* (нижний регистр) и *false* — любой другой ASCII символ. Ожидается, что метод кинет *IllegalArgumentException*, если входящий символ не является ASCII символом.

|  |
| --- |
| *// сигнатура метода* public static boolean isAsciiLowercase(char ch) {  *// TO DO* } |

**Пример:**

|  |
| --- |
| isAsciiLowercase('A'); // → false isAsciiLowercase('d'); // → true isAsciiLowercase((char)257); // → IllegalArgumentException |

### **isAsciiNumeric**

Написать метод ***isAsciiNumeric(char ch)***, который принимает на вход **char** и возвращает *true*, если символ — это цифра от *0* до *9* и *false* — любой другой ASCII символ. Ожидается, что метод кинет *IllegalArgumentException*, если входящий символ не является ASCII символом.

|  |
| --- |
| *// сигнатура метода* public static boolean isAsciiNumeric(char ch) {  *// TO DO* } |

**Пример:**

|  |
| --- |
| isAsciiNumeric('Z'); // → false isAsciiNumeric('1'); // → true isAsciiNumeric((char)257); // → IllegalArgumentException |

### 

### **isAsciiAlphabetic**

Написать метод ***isAsciiAlphabetic(char ch)***, который принимает на вход **char** и возвращает *true*, если символ — это буква от *a* до *z* (нижний регистр) или от *A* до *Z* (верхний регистр) и *false* — любой другой ASCII символ. Ожидается, что метод кинет *IllegalArgumentException*, если входящий символ не является ASCII символом.

|  |
| --- |
| *// сигнатура метода* public static boolean isAsciiAlphabetic(char ch) {  *// TO DO* } |

**Пример:**

|  |
| --- |
| isAsciiAlphabetic('Z'); // → true isAsciiAlphabetic('a'); // → true isAsciiAlphabetic('1'); // → false isAsciiAlphabetic((char)257); // → IllegalArgumentException |

### 

### **toAsciiUppercase**

Написать метод ***toAsciiUppercase(char ch)***, который принимает на вход **char** и возвращает символ в верхнем регистре. Ожидается, что метод кинет *IllegalArgumentException*, если входящий символ не является ASCII символом.

|  |
| --- |
| *// сигнатура метода* public static char toAsciiUppercase(char ch) {  *// TO DO* } |

**Пример:**

|  |
| --- |
| toAsciiUppercase('a'); // → 'A' toAsciiUppercase('B'); // → 'B' toAsciiUppercase('1'); // → '1' toAsciiUppercase('@'); // → '@' toAsciiUppercase((char)257); // → IllegalArgumentException |

### **toAsciiLowercase**

Написать метод ***toAsciiLowercase(char ch)***, который принимает на вход **char** и возвращает символ в нижнем регистре. Ожидается, что метод кинет *IllegalArgumentException*, если входящий символ не является ASCII символом.

|  |
| --- |
| *// сигнатура метода* public static char toAsciiLowercase(char ch) {  *// TO DO* } |

**Пример:**

|  |
| --- |
| toAsciiLowercase('a'); // → 'a' toAsciiLowercase('B'); // → 'b' toAsciiLowercase('1'); // → '1' toAsciiLowercase('@'); // → '@' toAsciiLowercase((char)257); // → IllegalArgumentException |

### 

### **makeUppercase**

Написать метод ***makeUppercase(char[] input)***, который принимает на вход массив символов и возвращает строку символов в верхнем регистре. Ожидается, что метод кинет *IllegalArgumentException*, если хоть один из входящих символов не является ASCII.

|  |
| --- |
| *// сигнатура метода* public static StringBuilder makeUppercase(char[] input) {  *// TO DO* } |

**Пример**:

|  |
| --- |
| makeUppercase({'h', 'e', 'l', 'L', 'o'}); // → "HELLO" makeUppercase({'@','h', 'e', 'l', 'L', 'o', '1', '2'}); // → "@HELLO12" makeUppercase({'h', 'e', 'l', 'L', 'o', (char)257}); // → IllegalArgumentException |

### 

### **makeLowercase**

Написать метод ***makeLowercase(char[] input)***, который принимает на вход массив символов и возвращает строку символов в нижнем регистре. Ожидается, что метод кинет *IllegalArgumentException*, если хоть один из входящих символов не является ASCII.

|  |
| --- |
| *// сигнатура метода* public static StringBuilder makeLowercase(char[] input) {  *// TO DO* } |

**Пример:**

|  |
| --- |
| makeLowercase({'h', 'e', 'l', 'L', 'o'}); // → "hello" makeLowercase({'@','h', 'e', 'l', 'L', 'o', '1', '2'}); // → "@hello12" makeLowercase({'h', 'e', 'l', 'L', 'o', (char)257}); // → IllegalArgumentException |

### **makeCamel**

Написать метод ***makeCamel(char[] input)***, который принимает на вход массив символов и возвращает строку символов в нижнем и ВЕРХНЕМ регистре последовательно (начиная с нижнего). Ожидается, что метод кинет *IllegalArgumentException*, если хоть один из входящих символов не является ASCII.

|  |
| --- |
| *// сигнатура метода* public static StringBuilder makeCamel(char[] input) {  *// TO DO* } |

**Пример:**

|  |
| --- |
| makeCamel({'h', 'e', 'l', 'L', 'o'}); // → "hЕlLo" makeCamel({'@','h', 'e', 'l', 'L', 'o', '0', '1', '2'}); // → "@HeLlO012" makeCamel({'h', 'e', 'l', 'L', 'o', (char)257}); // → IllegalArgumentException |

### 

### **isStringAlphaNumerical**

Написать метод ***isStringAlphaNumerical(char[] input)***, который принимает на вход массив символов и возвращает *true*, если все символы — это буквы или цифры, или пробелы. Ожидается, что метод кинет *IllegalArgumentException*, если хоть один из входящих символов не является ASCII.

|  |
| --- |
| *// сигнатура метода* public static boolean isStringAlphaNumerical(char[] input) {  *// TO DO* } |

**Пример:**

|  |
| --- |
| isStringAlphaNumerical({'a', 'b', '1', ' '}); // → true isStringAlphaNumerical({'a', ',', '1', ' '}); // → false isStringAlphaNumerical({'a', (char)257, '1', ' '}); // → IllegalArgumentException |

### **concatStrings**

Написать метод ***concatStrings(char[][] input)***, который принимает на вход двумерный массив символов, соединяет их и возвращает новый одномерный результирующий массив. Ожидается, что метод кинет *IllegalArgumentException*, если хоть один из входящих символов не является ASCII.

|  |
| --- |
| *// сигнатура метода* public static char[] concatStrings(char[][] input) {  *// TO DO* } |

**Пример:**

|  |
| --- |
| concatStrings({'a', 'b'}, {'c', 'd'}); // → {'a', 'b', 'c', 'd'} concatStrings({'a', 'b', (char)257}, {'c', 'd'}); // → IllegalArgumentException |

### 

### **toInt**

Написать метод ***toInt(char[] input)***, который принимает на вход массив символов и возвращает число **int**. Ожидается, что метод кинет *IllegalArgumentException*, если хоть один из входящих символов не является ASCII.

|  |
| --- |
| *// сигнатура метода* public static int toInt(char[] input) {  *// TO DO* } |

**Пример:**

|  |
| --- |
| toInt({'1', '2'}); // → 12 toInt({'0'}); // → 0 toInt({'-', '1', '1'}); // → -11 toInt({(char)257, '2'}); // → IllegalArgumentException |