



Аннотация: Этот документ является практическим руководством к Week 1 Day 2 в Coding Bootcamp.

Академия Ковалевского



Теория	3
Задания: работа с аргументами	3
Вывод на экран аргументов программы	4
Вывод на экран аргументов программы в обратном порядке	4
Вывод на экран аргументов в сортированном порядке	5
Задание: StdString	6
StdString(char[] base)	7
StdString()	7
StdString(StdString stdString)	7
int length()	7
StdString append(StdString that)	7
char[] toCharArray()	8
char charAt(int index)	8
int indexOf(int target)	8
boolean equals(final Object otherObj)	8
int hashCode()	9
String toString()	9
Iterator<Character> iterator()	10
void forEach(final Consumer<? super Character> action)	10



Теория

В данном уроке рекомендуется ознакомиться со следующими теоретическими понятиями:

- Как запускать java программу из консоли:
 - [Компиляция и выполнение java программы с командной строки](#)
 - [Сборка и выполнение Java программ](#)
- Что такое main метод в Java ([пример статьи](#))
- Что такое Iterator ([пример статьи](#))
- Что такое forEach метод ([пример статьи](#))

Напоминаем, что ссылки на внешние источники просто примеры, крайне рекомендуем самостоятельно поискать материалы по интересующим темам, чтобы все в группе смотрели разные источники по теме.





Задания: работа с аргументами

Вывод на экран аргументов программы

Вам необходимо реализовать консольную программу в следующем классе:

```
package com.kovalevskyi.academy.codingbootcamp.week1.day2;

public class MainPrintParam {

    public static void main(String[] args) {
        // TODO
    }
}
```

Программа должна выводить на экран все аргументы переданные ей при вызове. Например, если на вход переданы аргументы:

```
arg1 arg2 arg3
```

то программа должна вывести на экран:

```
arg1 arg2 arg3\n
```

Если не было передано ни одного аргумента, то программа должна вывести следующее сообщение:

```
Please specify at least one argument!\n
```

Вывод на экран аргументов программы в обратном порядке

Вам необходимо реализовать консольную программу в следующем классе:

```
package com.kovalevskyi.academy.codingbootcamp.week1.day2;

public class MainPrintReversedParam {

    public static void main(String[] args) {
        // TODO
    }
}
```



Программа должна выводить на экран все аргумент, переданные ей при вызове, в обратном порядке. Например, если на вход переданы аргументы:

```
arg1 arg2 arg3
```

то программа должна вывести на экран:

```
arg3 arg2 arg1\n
```

Если не было передано ни одного аргумента, то программа должна вывести следующее сообщение:

```
Please specify at least one argument!\n
```

Вывод на экран аргументов в сортированном порядке

Вам необходимо реализовать консольную программу в следующем классе:

```
package com.kovalevskyi.academy.codingbootcamp.week1.day2;

public class MainPrintSortedParam {

    public static void main(String[] args) {
        // TODO
    }
}
```

Программа должна выводить на экран все аргументы, переданные ей при вызове, в сортированном порядке. Например, если на вход переданы аргументы:

```
cat abba zoo yield
```

то программа должна вывести на экран:

```
abba cat yield zoo\n
```

Если не было передано ни одного аргумента, то программа должна вывести следующее сообщение:

```
Please specify at least one argument!\n
```





Задание: StdString

В этом задании мы начнем создавать свой собственный аналог класса String! Для начала давайте обозначим сигнатуру класса:

```
package com.kovalevskiy.academy.codingbootcamp.week1.day2;
```

```
import java.util.Iterator;
```

```
public class StdString implements Iterable<Character> {
```

```
    public StdString(char[] base) {  
        // TODO  
    }
```

```
    public StdString() {  
        // TODO  
    }
```

```
    public StdString(StdString stdString) {  
        // TODO  
    }
```

```
    public int length() {  
        // TODO  
    }
```

```
    public StdString append(StdString that) {  
        // TODO  
    }
```

```
    public char[] toCharArray() {  
        // TODO  
    }
```

```
    public char charAt(int index) {  
        // TODO  
    }
```

```
    public int indexOf(char target) {  
        // TODO  
    }
```



```
@Override
public boolean equals(final Object otherObj) {
    // TODO
}

@Override
public int hashCode() {
    // TODO
}

@Override
public String toString() {
    // TODO
}

@Override
public Iterator<Character> iterator() {
    // TODO
}

@Override
public void forEach(final Consumer<? super Character> action) {
    // TODO
}
}
```

Рассмотрим каждый метод по отдельности:

StdString(char[] base)

```
public StdString(char[] base) {
    // TODO
}
```

Простой конструктор, который создает строку на базе входящего массива символов.

StdString()

```
public StdString() {
    // TODO
}
```

Пустой конструктор, который создает пустую строку.





StdString(StdString stdString)

```
public StdString(StdString stdString) {  
    // TODO  
}
```

Конструктор копирования - создает строку идентичной той, которая передана на вход.

int length()

```
public int length() {  
    // TODO  
}
```

Возвращает длину строки.

StdString append(StdString that)

```
public StdString append(StdString that) {  
    // TODO  
}
```

Создает новый класс StdString, который включает в себя обе строки. Например, если строка "cat" и другая строка "dog", то метод append вернет новую строку "catdog".

char[] toCharArray()

```
public char[] toCharArray() {  
    // TODO  
}
```

Возвращает массив символов текущей строки.

char charAt(int index)

```
public char charAt(int index) {  
    // TODO  
}
```

Возвращает символ строки по определенному индексу. Если индекс указан не корректно, то кидает `IndexOutOfBoundsException`



int indexOf(int target)

```
public int indexOf(char target) {  
    // TODO  
}
```

Находит индекс первого вхождения заданного символа в текущей строке, или -1 если символ не найден. Например:

```
"cat", 'a' => 1
```

```
"asdef23q4asdfasdf", 'd' => 2
```

```
"dog", 'q' => -1
```

boolean equals(final Object otherObj)

```
@Override  
public boolean equals(final Object otherObj) {  
    // TODO  
}
```

Метод equals() должен проверить соответствие всех символов входящей строки с текущей строкой (помимо стандартных проверок - по типу, по ссылке и т.д.)

int hashCode()

```
@Override  
public int hashCode() {  
    // TODO  
}
```

Хеш код подсчитывается путем суммирования целочисленных представлений каждого символа в строке. "Cat" → C: 67 a: 97 t: 116 → 67 + 97 + 116 = 280

String toString()

```
@Override  
public String toString() {  
    // TODO  
}
```

Единственный метод, где можно использовать java.lang.String. Конвертирует объект StdString в объект String.



Iterator<Character> iterator()

```
@Override
public Iterator<Character> iterator() {
    // TODO
}
```

Метод `iterator()` создает новый итератор, который проходит по всем символам строки.

void forEach(final Consumer<? super Character> action)

```
@Override
public void forEach(final Consumer<? super Character> action) {
    // TODO
}
```

Метод `forEach()` проходит по каждому символу в строке и применяет к нему `action`.

