



Аннотация: этот документ является практическим руководством к Coding Bootcamp.

Week 2 Day 0

**Академия Ковалевского**

**Содержание**

[**1. Теория**](#_t6lx1gefs5vu) **3**

[Рекурсия](#_6icpi1lntesi) 3

[Рекомендации](#_5vvukt5ncmvw) 3

[**2. Практическая работа**](#_x0e5z3m82g6s) **4**

[Пакет](#_pom8kfz6xn97) 4

[Принтер параметров](#_d2iufm5386ru) 4

[Принтер параметров на выворот](#_xc9hk5dlkcei) 4

[Принтер параметров, отсортированных](#_ottidmey5kqh) 5

[Калькулятор](#_6fx42t66hfqz) 5

[Копаем дальше математику](#_kqdszw11kxrr) 5

[Обычное вычисление факториала](#_cax2rpwbaic0) 6

[Рекурсивное вычисление факториала](#_da9mfuy5fzu2) 6

[Обычное вычисление степени](#_myall27gub9v) 6

[Рекурсивное вычисление степени](#_ylw01m55vwh0) 7

[Вычисление числа фибоначчи](#_iv4jzr518ico) 7

[Вычисление ряда фибоначчи](#_fka88qss173u) 7

[Корень квадратный](#_r5ug1yxydvxf) 7

[Простое число](#_e9k3n5w3qyrw) 8

[Поиск простого числа](#_y2sd5fsflv6) 8

# 

# 1. Теория

С чего начинается любая java-программа?

Точкой входа в любую java-программу является main-метод. А в этом методе мы уже начинаем дергать собственную магию классов и методов.

* Поинтересуйтесь о входящих аргументах main-метода, что они из себя представляют.
* Выясните какие ключевые моменты присутствуют в сигнатуре main-метода, чем он отличается от других методов, кроме названия.
* Ответьте на вопрос: что такое переполнение типа?
* Выясните для себя как можно больше методов для вывода на экран чего либо.

## Рекурсия

Шуточным объяснением рекурсии является: для того чтобы понять рекурсию — нужно понять рекурсию.

Программирование с применением рекурсии подразумевает применение рекурсивных вызовов. Это требует немного иного взгляда на решение той или иной задачи. Рекурсивные вызовы могут оказаться проще чем, например, написание хитрых циклов и условий. Но рекурсивные вызовы хранят в себе опасность переполнения стека методов в JVM, что приводит к выбросу [StackOverflowError](https://www.codeflow.site/ru/article/java-stack-overflow-error). По легендам — эта ошибка помогла назвать соответствующий сайт с вопрос-ответами.

## Рекомендации

* Мойте руки, клавиатура не должна быть залапана.
* Найдите примеры рекурсии в java:
  + Изучите их внимательно, осознайте
  + Поиграйтесь с ними у себя в любом тестовом классе
  + Задавайте вопросы о непонятном друг-другу, перед началом выполнения заданий
* **Вопрос**: какой предел стека в JVM?

# 

# 2. Практическая работа

В этом задании придется опять реализовывать не только математику, но и возится со строками. Сигнатуры методов будут даны не для всех заданий текущего дня. **Разрешено:** использовать только свои реализации различных методов из этого и предыдущих дней. **Запрещено:** использовать готовые парсеры чисел и строк из библиотек java.

## Пакет

Все классы текущего задания должны находиться в

|  |
| --- |
| package com.kovalevskyi.academy.codingbootcamp.week2.day0; |

## Принтер параметров

Это класс-игрушка для вывода всех входящих в программу аргументов на экран. Имя класса: MainPrintParam

Единственный *публичный* метод этого класса принимает неопределенное количество аргументов. Метод должен принимать аргументы в момент запуска программы. **Задача метода** вывести на экран в столбик все входящие аргументы. Если аргументов нет — вывести сообщение:

***Please specify at least one argument!***

## Принтер параметров навыворот

Этот класс делает все тоже самое, что и предыдущий класс. Но выводит аргументы на экран в обратном порядке. Имя класса: MainPrintReversedParam

Не забудьте про реакцию на отсутствие аргументов.

## 

## Принтер параметров, отсортированных

Этот класс тоже повторяет предыдущие, но с элементом сортировки. Имя класса должно быть: MainPrintSortedParam

Единственный метод этого класса принимает неопределенное количество аргументов, во время запуска программы, сортирует их и выводит на экран поочередно.

## Калькулятор

В данном классе мы будем реализовывать простой консольный калькулятор! И “переваривать” он должен различные случайные числа из диапазона Integer. Имя класса должно быть: Calculator. Содержит этот класс в себе всего один публичный метод ***main***, который способен принимать определенное количество аргументов, в момент запуска программы.

Наш калькулятор на вход ожидает **три аргумента.** Если пользователь передал другое число аргументов — на экран должно быть выведено следующее:

***Please provide 3 input arguments, example: 2 + 3***

Три аргумента, которые ожидает программа:

* Целое число
* Математическая операция: **+**, **-**, **/**, **\*** или **%**
* Целое число

Если любой из аргументов не соответствует этим условиям — нужно кинуть *IllegalArgumentException*.

Если на вход пришел делитель равный нулю — на экран должно выйти следующее сообщение:

***Division by zero is impossible!***

Если все в порядке — программа должна вывести ответ в формате: ***result: trololo***

Например:

* **in**: *3 + 2* **out**: *result: 5*
* **in**: *-5 + -10* **out**: *result: -15*
* **in**: abc + 2 **out**: IllegalArgumentException

## Копаем дальше математику

Имя класса: NumberUtils. Все методы этого класса статические. Некоторые из них повторяют поведение друг-друга, но должны отличаться реализацией.

Фактически в каждом методе нужно проверять аргументы на валидность, это зависит от конкретных математических ограничений для каждого метода. Если аргумент не валидный — кидать *IllegalArgumentException*

### **Обычное вычисление факториала**

Вы уже писали данный метод. Можете переиспользовать его. Но для складности задания — он должен быть в этом классе.

|  |
| --- |
| *// сигнатура метода*  public static int getFactorial(final int number) {  // TODO } |

### **Рекурсивное вычисление факториала**

Обязательно проверяйте друг у друга способ реализации этого метода. Этот метод вычисляет факториал входящего числа.

|  |
| --- |
| *// сигнатура метода*  public static int factorialRecursive(int number) {  // TODO  } |

Предусмотрите перехват нелегальных аргументов.

### **Обычное вычисление степени**

Напишите метод, который принимает два числа: основу и степень. Неожиданно: основу нужно возвести в степень!

|  |
| --- |
| *// сигнатура метода*  public static long power(int base, int power) {  // TODO } |

### **Рекурсивное вычисление степени**

Этот метод делает то же самое, что и предыдущий, но реализован с помощью рекурсии. Проверяйте способ реализации друг у друга!

|  |
| --- |
| *// сигнатура метода*  public static int powerRecursive(int base, int power) {  // TODO } |

### **Вычисление числа фибоначчи**

[Что это?!](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%B0_%D0%A4%D0%B8%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D0%B8#:~:text=0%2C%201%2C%201%2C%202,%D1%80%D0%B0%D0%B2%D0%BD%D0%BE%20%D1%81%D1%83%D0%BC%D0%BC%D0%B5%20%D0%B4%D0%B2%D1%83%D1%85%20%D0%BF%D1%80%D0%B5%D0%B4%D1%8B%D0%B4%D1%83%D1%89%D0%B8%D1%85%20%D1%87%D0%B8%D1%81%D0%B5%D0%BB.) Сам метод должен находить число фибоначчи по заданному индексу.  
Метод сразу должен иметь рекурсивную реализацию. Ищем только позитивные числа фибоначчи.

|  |
| --- |
| *// сигнатура метода*  public static int fibRecursive(int index) {  // TODO } |

### **Вычисление ряда фибоначчи**

Этот метод должен вычислить и вернуть весь ряд чисел фибоначчи. Длина ряда задается входящим в метод аргументом.

|  |
| --- |
| *// сигнатура метода*  public static int[] fibSequence(int length) {  // TODO } |

### 

### **Корень квадратный**

Пишем метод, который вычисляет квадратный корень для входящего числа. Находит он только целочисленные варианты квадратного корня из числа. Если из числа невозможно найти целочисленный квадратный корень — возвращаем ***-1***.

|  |
| --- |
| *// сигнатура метода*public static int sqrt(int target) {  // TODO } |

### **Простое число**

Этот метод определяет простоту входящего числа. Он отвечает на вопрос: *число простое?*

|  |
| --- |
| *// сигнатура метода*  public static boolean isPrime(int target) {  // TODO } |

### **Поиск простого числа**

Данный метод должен находить ближайшее простое число в большую сторону от входящего числа.

Если входящее число уже является простым — его и возвращаем. Не забываем проверять легальность аргумента.

|  |
| --- |
| *// сигнатура метода*  public static int findNextPrime(int target) {  // TODO } |