

Домашнее задание №2

- Deadline:
 - мягкий 15.10.2019, 23:59
 - жёсткий 22.10.2019, 23:59
- [Ссылка на контекст](#).
- В задачах нужно реализовать JavaDoc для нетривиальных методов и классов.

Технические подробности

Для сдачи задания в контекст требуется загрузить не код, а собранный Jar-файл. В этом Jar должен также иметься файл MANIFEST.MF, в котором должен быть указан класс с методом `main()`. Например:

```
Manifest-Version: 1.0
Main-Class: org.onces.lesson1.Main
```

Задача №1. Расстояния. (40%)

Даны координаты 4 точек (A, B, C, D) на плоскости: $X_A, Y_A, X_B, Y_B, X_C, Y_C, X_D, Y_D$. Рассчитайте такие расстояния:

- от точки A до точки C
- от точки A до отрезка CD
- от точки A до луча CD
- от точки A до прямой CD
- от отрезка AB до точки C
- от отрезка AB до отрезка CD
- от отрезка AB до луча CD
- от отрезка AB до прямой CD
- от луча AB до точки C
- от луча AB до отрезка CD
- от луча AB до луча CD
- от луча AB до прямой CD
- от прямой AB до точки C
- от прямой AB до отрезка CD
- от прямой AB до луча CD
- от прямой AB до прямой CD

Для точки, отрезка, луча и прямой нужно реализовать классы (схема наследования - на ваше усмотрение).

Формат входных данных

4 пары целых чисел (по паре в строке). По модулю числа ≤ 10000 .

Формат выходных данных

16 чисел, по 1 в строке. Числа должны иметь 6 знаков после запятой. Допускается погрешность $1E-4$.

Примеры входных данных

in	out
1 2	5.6568542495
7 1	5.6000000000

5 6	5.6000000000
8 2	5.6000000000
	4.6031716446
	1.4142135624
	1.4000000000
	1.4000000000
	4.6031716446
	1.1507929111
	0.0000000000
	0.0000000000
	4.6031716446
	1.1507929111
	0.0000000000
	0.0000000000

Задача №2. Длинная арифметика. (60%)

Напишите класс **VeryLong** для работы с длинными целыми числами. Должны поддерживаться операции:

1. сложение, вычитание, умножение, деление, остаток по модулю, работающие так же, как и для обычных чисел;
2. составное присваивание с этими операциями (+=, *=, -=, /=);
3. унарный минус,
4. инкремент и декремент,
5. операторы сравнения (== != < > <= >=),
6. метод toString(), toBoolean(), toInteger(), toLong(). В двух последних случаях нужны проверки, не выходит ли число за рамки типа; если выходит - бросаем Exception. Желательно бросать не один из существующих Exception'ов, а создать своего наследника.
7. конструкторы из int, long, String, boolean

Асимптотические ограничения на время работы операторов в констесте не проверяются, но реализация должна работать за:

- Сложение, вычитание, унарный минус, операторы сравнения – $O(n)$,
- Умножение, деление, остаток по модулю – $O(n^2)$,

где n – количество разрядов большего числа (по модулю). Это будет выявляться на code review.

В данной задаче не нужно следить за порядком действий в выражении (например, выражение $a + b * c$ будет вычисляться как $(a+b) * c$). Если в выражении содержатся операции из п. 2,4,5, то других операций в выражении нет.

Использовать готовый класс BigInteger из стандартной библиотеки Java **не разрешается**.

На вход подаётся выражение. Нужно:

- 1) Привести числа к типу VeryLong.
- 2) Произвести вычисления.
- 3) Результат вывести в консоль.

Формат входных данных

Строка выражения. Числа и знаки действий (кроме унарного минуса) отделены пробелами.

Формат выходных данных

Результат вычислений в виде строки.

Примеры входных данных

in	out
937 + -13761376179864981487987	-13761376179864981487050
123448134 += 4	123448138
13423243345678908765 >= 233456789	true
2347205 * 123 + 233	288706448
1234++	1235

Задача №3 [Бонус]. Поиск по шаблону (50%)

Шаблон поиска задан строкой длины m , в которой кроме обычных символов (латиница, нижний регистр) могут встречаться символы “?”. Найти позиции всех вхождений шаблона в тексте длины n . Каждое вхождение шаблона предполагает, что все обычные символы совпадают с соответствующими из текста, а вместо символа “?” в тексте встречается произвольный символ.

Время работы - $O(n + m + Z)$, где Z - общее число вхождений подстрок шаблона между wildcard'ами (знаками вопроса) в исходном тексте. $m \leq 5000$, $n \leq 2000000$.

Интерфейс для задачи:

```
public interface PatternFinder {
    int findConjunction(String pattern, String example);
}
public interface WildCard {
    List<String> getAllConjunctions(String pattern, String example);
}
```

Примеры входных данных

in	out
ab??aba ababacaba	2