Комментарии к ДЗ по модулю 4.

Урок 3.

Собственно такой результат из-за представления чисел с плавающей точкой в двоичной системе исчисления, ни одна правильная дробь вида , кроме чисел со знаменателем в виде степени двойки не может быть однозначно получена в двоичной системе исчисления, а представление чисел по спецификации  IEEE 754 подразумевает правильную дробь – то есть мантиссу, умноженную на 2^E. Почему нельзя представить, ну это достаточно очевидный момент, для этого надо просто попробовать делить в двоичной системе исчисления например 1/11, то есть 1/3, очевиден ответ 0,0111111111..

То есть точного значения, как и в случае с 1/3 для десятичной дроби получить невозможно. Числа 2.4 и 0.1 это 24/10 и 1/10 в знаменателе не степени двойки, значит ошибки округления будут, в случае со сложением берется минимальный порядок ошибки, то есть если ошибка округления a = 10^(-10) а ошибка округления b = 10^(-9) то ошибка округления a+b 10^(-9), в случае умножения чисел ошибка умножается. В принципе, могу прочитать лекцию по теории приближенных вычислений и накапливаемым ошибкам, если интересно)))

Но пока это означает, что в случае сравнения double необходимо пользоваться неким малым эпсилон, заранее определив его из условия задачи. Если же погрешность недопустима, то необходимо уходить в bigDecimal, что я собственно и сделал.

Урок 5.

- Вывод русских символов сделал, ё в алфавитном порядке готова.

- с заработком маши и Васи там отдельная папка StringExperiments вне уроков, прошу прощения за неудобства.

Урок 7.

- Задача с деньгами, магия, но работает, то есть по факту мы оставили запятую и пошли по ней, а ее до 5000 не было, а значит ничего не записалось в 0 элемент до 5000, а так как в первоначальной версии разделители были до 5000, то, получается в реализации split есть образный аккумулятор, он был пустой строкой, а так как первый символ был разделителем, то текущее состояние аккумулятора перешло в массив, как я это понял. А когда первый символ не разделитель, то есть цифра, то этот символ записывается в аккумулятор и просто читается следующий, по идее автоматная логика по каждому символу должна здесь работать.

- if(!Pattern.compile("[А-ЯЁ&&[^ЪЬЫ]]{1}[а-яё]+\\s[А-ЯЁ&&[^ЪЬЫ]]{1}[а-яё]+\\s" +

"[А-ЯЁ&&[^ЪЬЫ]]{1}[а-яё]+$").matcher(fullName).matches())

Теперь классная регулярочка, понял даже почти все, получается, класс Pattern создает объект регулярочки методом compile и для него вызывается проверка, удовлетворяет ли fullName составленному регулярному выражению. Теперь с регуляркой. Любой заглавный символ русского алфавита одновременно не ЪЬЫ, чтобы ну в русском языке так нельзя. Получается {1} говорит, что заглавная буква ровно одна, а дальше идет произвольное количество незаглавных букв, но уже + значит хотя бы один, после чего пробельный отступ, как в занятии показывали ну и все опять для имени и отчества. А не, я все понял. Да, в конце $ - конец строки. Да, кстати, {1} же здесь необязательна получается, квадратные скобки – это всегда один символ. Опять же, если не стоит + или \*.

А, регулярки классные, аж захотелось автоматик какой-нибудь построить.