Extra Задача №2

Создать массив строк из любой строчки песни/текста (длина массива не менее 5 элементов). Любыми доступными знаниями выстроить элементы массива

в алфавитном порядке. Вывести на печать

Extra Задача №3

Создать массив строк из любой строчки песни/текста (длина массива не менее 10 элементов). Любыми доступными знаниями подсчитать и вывести

на печать каждую букву что встречается в массиве. (Пример а = 5, b = 4, если = 0 можно не выводить на печать).

Написать метод, который принимает на вход необходимые параметры и возвращает строку ведомости выдачи зарплаты сотрудникам.

Например:

“Смирнова Мария Ивановна 70000 руб 00 коп”

если имя сотрудника придет в виде пустой строки или в виде null (String - это референсный тип данных и может принимать значение null), то в

ведомости должна быть распечатана пустая строка "".

Если же на вход придет невалидное данное по зарплате, или зарплата будет рассчитана, как 0, то в ведомости тоже должна быть распечатана пустая

строка ""

Создать метод, который печатает ведомость для нескольких сотрудников, используя метод из задачи 8, например:

Смирнова Мария Ивановна 70000 руб 00 коп

Серебров Иван Петрович 128059 руб 00 коп

Метод в задании 9 может принимать разные параметры, в зависимости от того, что вам нужно для печати ведомости.

Смысл этого метода - распечатать несколько строк ведомости одновременно. Каждая из строк должна формироваться с помощью метода, написанного

в задании 8.

Написать метод, который принимает на вход десятичное число (например, 10.75), и возвращает строку “10 руб 75 коп”.

Написать метод, который принимает на вход десятичное число (например, 10,075) и возвращает строку “10 кг 75 гр”.

Написать метод, который принимает на вход необходимые параметры, и печатает чек. Этот метод тестировать не нужно, но необходимо использовать

все раннее написанные методы (в задачах из Part 1 тоже могут пригодиться некоторые методы)

Пример чека:

Яблоки

Цена за 1 кг 50 руб 13 коп

Количество товара 3 кг 400 гр

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Сумма к оплате 170 руб 44 коп

или

Хлеб

Цена за 1 шт 30 руб 50 коп

Количество товара 5 шт

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Сумма к оплате 152 руб 50 коп

Написать метод, который принимает на вход год рождения и возвращает ваше счастливое число. Счастливое число рассчитывается по формуле: сумма всех чисел, если результат больше 9, снова считается сумма всех чисел.

Например:

год рождения 1999

1 + 9 + 9 + 9 = 28

2 + 8 = 10

1 + 0 = 1

Счастливое число – 1

а) Дано 3 числа. Необходимо рассчитать разницу между средним значением и медианой (median) значением.

Если разница больше 2, необходимо показать сообщение: “Среднее значение … отличается от медианы … на … “.

Иначе показать сообщение: “Среднее значение = …, медиана = … ”.

b) Посчитать все то же самое с помощью методов класса Math, где возможно их использовать

\*Медиана - это серединное число в отсортированном наборе чисел.

1, 3, 9 → медиана 3

12, 13, 101 → медиана 13

14, 2, 12 → медиана 12

ВСе параметры приходят в int, расчеты производим в int

Написать метод, который использует методы класса Math, принимает на вход сумму к оплате (например, 10.75) и округляет сумму в

пользу покупателя. Метод возвращает новую сумму к оплате в виде строки, например “10 руб 00 коп”.

Написать метод, который возвращает целочисленное случайное число в пределах от 1 до 99 включительно. Затем написать метод,

который возвращает случайное число в пределах от -1 до -99 включительно

**Задача №3**

Необходимо создать класс **Employee** с полями как у **Person** (из предыдущего задания) и полем зарплата. Класс должен иметь метод **isSameName(Employee employee)** который возвращает true, если у сотрудника у которого был вызван метод и сотрудника который был передан как параметр, одинаковое имя.

**Задача №4**

Необходимо создать класс **SalaryUtils** с единственным методом - **getSum(Employee[] employeeArray)**, метод должен возвращать сумму зарплат всех сотрудников из массива переданного в качестве аргумента вызова метода.

**Задача №5**

Необходимо создать утилитарный класс **EmployeeUtils** со следующими методами:

* поиск сотрудника в массиве по его имени
* поиск сотрудника в массиве по вхождению указанной строки в его имени
* подсчет зарплатного бюджета для всех сотрудников в массиве
* поиск наименьшей зарплаты в массиве
* поиск наибольшей зарплаты в массиве

**Задача от Алисы**  
Создать класс в котором будут поля Имя, Возраст, Страна, Зарплата (цифрами просто), образование (Bachelor, PhD и тд). Создать во ВТОРОМ классе около 10 человек (минимум 8 ) и использовать методы первого класса. Сделать сортировки по всем параметрам(например сначала выводит тех кто из США, потом кто из Японии и тд).

На основе данной задачи, будут еще по мере продвижения по темам. Опционально к выполнению.

**ЗАДАНИЯ 7**

**Задача №1**

Необходимо создать класс Employee со следующими методами:

* getBaseSalary - получить базовую ставку
* set Base Salary
* getName - получить имя
* setName
* getSalary - получить зарплату

**Задача №2**

Необходимо создать класс Worker где метод getSalary будет возвращать базовую ставку.

Необходимо создать класс Manager в который нужно добавить следующие методы:

* getNumberOfSubordinates - получить количество подчиненных
* setNumberOfSubordinates

 в классе, метод getSalary будет возвращать значение по формуле - <базовая ставка> \* (<количество подчиненных> / 100 \* 3). Если количество подчиненных 0, то результат как у обычного рабочего.

Необходимо создать класс Director с теми же методами, что и Manager, но метод getSalary должен возвращать результат по формуле - <базовая ставка> \* (<количество подчиненных> / 100 \* 9). Если количество подчиненных 0, то результат как у обычного рабочего.

**Задача №3**

Создать класс Month с полями: имя месяца, количеством дней и количеством рабочих дней. Создать класс MonthUtils который бы хранил подготовленные месяцы или их массивы для использования (объекты класса Month).

**Задача №4**

Необходимо изменить класс Employee с полями: имя, возраст, пол и ЗП в день. Все поля сделать приватными и для каждого поля добавить методы set и get. Класс должен иметь метод - getSalary(Month[] monthArray), метод возвращает зарплату за те месяцы которые были переданы в качестве аргумента.

**Задача №5**

Необходимо изменить класс Manager с полями: имя, возраст, пол, ЗП в день и количество подчиненных. Все поля сделать приватными и для каждого поля добавить методы set и get. Класс должен иметь метод - getSalary(Month[] monthArray), метод возвращает зарплату за те месяцы которые были переданы в качестве аргумента.

К рассчитанной ЗП должно прибавляться по 1% за каждого подчиненного.

**Задача №6**

Необходимо расширить утилитарный класс **EmployeeUtils** из прошлого задания следующими методами:

* поиск наименьшего количества подчиненных в массиве менеджеров
* поиск наибольшего количества подчиненных в массиве менеджеров
* поиск наибольшей надбавки (разнице между базовой зарплатой и выплатой) в массиве менеджеров
* поиск наименьшей надбавки (разнице между базовой ставки и зарплатой) в массиве менеджеров

**Задача №7**

<https://www.codewars.com/kata/57eadb7ecd143f4c9c0000a3/train/java>

<https://www.codewars.com/kata/5aa736a455f906981800360d/train/java>

<https://www.codewars.com/kata/5704aea738428f4d30000914/train/java>

<https://www.codewars.com/kata/5808e2006b65bff35500008f/train/java>

<https://www.codewars.com/kata/5a2be17aee1aaefe2a000151/train/java>

ЗАДАНИЕ 7 для бегинеров

**Part 1**

ВСе задания выполнять в классе HW7  в проекте JavaForBeginners (без Maven)

1. Написать универсальный метод для построения последовательности, которая начинается с числа start, заканчивается числом end (оба включительно). Все числа в этой последовательности должны быть кратны числу step. Этот метод должен высчитывать первое кратное число в диапазоне [start, end] и строить последовательность (возрастающую или убывающую) от вычисленного первого кратного числа с шагом step.

(Эта задача по мотивам разбора HW6,  Part 1  в воскресенье 9 октября)

1. Создать массив catsNames[], заполнить его **любыми своими** значениями (не такими, как были на уроке).
2. В массиве catsNames[] изменить значение элемента с индексом 4 на “Рыжик”,  а значение элемента с индексом 1 на “Черныш”.
3. Создать массив catsColors[] и заполнить его значениями (см картинку в презентации).
4. Создать массив catsAges[] и заполнить его любыми целочисленными валидными значениями.
5. Создать массив isCatRed[] и заполнить его соответствующими значениями (см картинку в презентации. Red = рыжий)
6. Распечатать для массивов catsNames[] и catsColors[]:

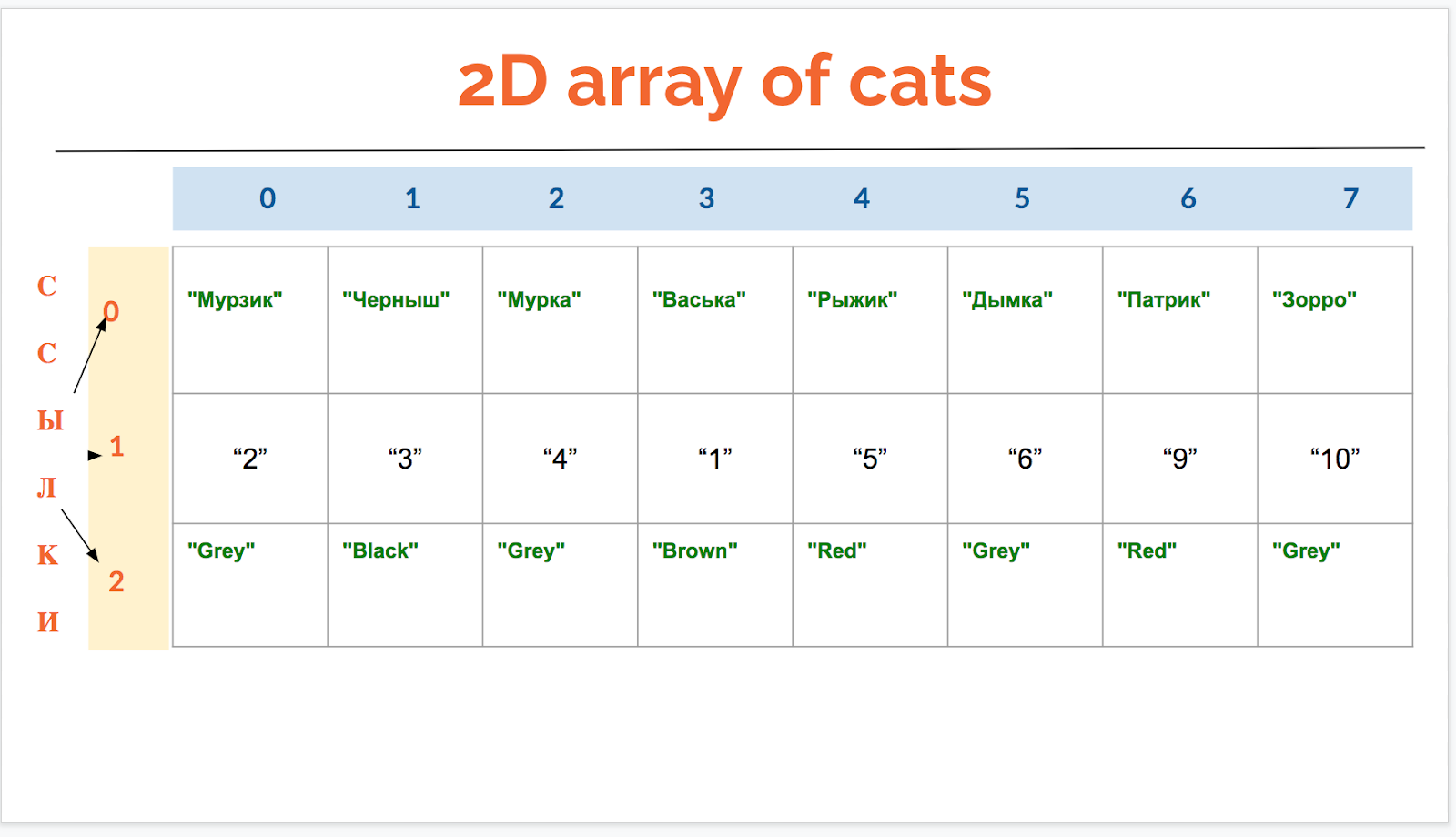
* имя кота в коробке с номером 6
* имена котов из коробок с четными индексами
* имена котов из коробок, чьи индексы кратны 4
* цвет котов из коробок с нечетными индексами
* цвет котов из коробок, чьи индексы кратны 3, но не кратны 2

1. Распечатать “Накорми кота!” для всех серых котов
2. Распечатать “Дай коту воды!” для всех котов со значениями true из массива isCatRed[]
3. Распечатать “Отнеси кота на прививку!”, если возраст кота меньше 2 лет
4. Для кота в последней коробке распечатать имя, цвет, возраст
5. Распечатать имя, цвет, возраст котов в двух серединных коробках
6. Распечатать имена всех котов, чей возраст больше 2 лет
7. Распечатать “Накорми … !” , где … - имя кота, а значение isCatRed == true
8. Распечатать средний возраст котов из массива catsAges[]
9. Распечатать возраст самого молодого кота
10. Распечатать возраст самого старого кота
11. Распечатать количество серых котов и количество рыжих котов
12. Распечатать имя кота, если кот находится в коробке с нечетным индексом и его возраст не больше 2 лет

**Part 2**

1. Написать метод, который создает и печатает массив четных положительных чисел, значения которых не больше 6. Заполняем значения и печатаем с помощью цикла for. Длина массива - l.
2. Создать массив всех нечетных отрицательных чисел в промежутке от (-1000 до -900).
3. Создать массив из 10 случайных положительных целых чисел в промежутке от 100 до 200 включительно.
4. Написать метод, который принимает на вход массив int[] array и число int number. Метод проверяет, содержится ли число number в массиве array. При первом нахождении числа number, метод печатает “yes”. При каждом нахождении чисел на 1 больше числа number,  или на 1 меньше числа number, метод печатает “maybe”. Метод печатает “no”, если не находит ни одного числа, соответствующего условиям.

**Part 3**

1. Создать массив четных чисел и массив нечетных чисел из элементов массива из задания 24.
2. Создать двумерный массив, который состоит из имен, возрастов, цветов котов:  
     
     
   Распечатать все данные котов в коробках с четными индексами, используя двумерный массив.
3. Создать двумерный массив целых случайных чисел от 1 до 10 включительно, размерность массива 4\*8.
4. Вывести сумму всех четных чисел массива из задания 28.

**HW 8 for Beginners**

2. Написать алгоритм OddIndices, который принимает массив целых чисел,  и возвращает массив значений нечетных индексов

Test Data:

Input = {-45, 590, 234, 985, 12, 68}

Expected Result =  {590, 985, 68}

3.

4.



**Part 2 Algorithms**

8. Написать алгоритм MinMaxAve, который принимает массив чисел int[]  и 2 значения индексов. Алгоритм возвращает массив, который содержит минимальное значение, максимальное значение,  и среднее среди всех значений между 2-мя индексами.

Test Data:

({1, 2, 3, 4, 5, 6, 7, 8}, 2, 6) →  {3, 7, 5}

9. Написать алгоритм Intersection, который принимает на вход 2 массива целых чисел и возвращает массив общих элементов.  
  
Test Data:

{1, 2, 4, 5, 89}, {8, 9, 4, 2} → {2, 4}  
{1, 2, 4, 5, 8, 9}, {8, 9, -4, -2} → {8, 9}

{1, 2, 4, 5, 89}, {8, 9, 45} → {}

10. Написать алгоритм PeakElement,  который принимает на вход массив целых чисел и возвращает значения пиковых элементов (элементы, которые больше своих соседей).

Test Data:  
{3, 2, 7, 5, 1, 9, 23, 1} → {3, 7, 23}

11. Написать алгоритм ReverseArray, который принимает на вход массив целых чисел, и возвращает “перевернутый” массив.

Test Data:  
{2, 7, 3, 10} → {10, 3, 7, 2}

12.

13.

13. Написать алгоритм NegativeOnTheRight, который принимает на вход массив целых чисел, и возвращает массив,  в котором все негативные числа находятся во второй половине массива (массив не должен быть отсортирован)  
Test Data:

{4, -3, 7, -12, 5, -2, 9, 4, 12} → {4, 7, 5, 9, 4, 12, -3, -12, -2}

14. Написать алгоритм SumOfTwo, который принимает на вход массив целых чисел,  и число n. Алгоритм  возвращает пары элементов, которые в сумме дают число n.  
Test Data:  
({4, 3, 7, 12, 5, 2, 9, 4, 12}, 12)  → {{3, 9}, {7, 5}}

15. Написать алгоритм NumberOccurrences,  который принимает на вход массив целых чисел,  и возвращает массив пар   
{число, сколько раз число встречается в массиве}  
 Test Data:  
{3, 2, 5, 3, 1, 5, 4, 2, 1, 4, 5, 3, 2, 1, 4, 5, 3, 2, 1} →   
{{1, 4}, {2, 4}, {3, 4}, {4, 3}, {5, 4}}

**Part 1.1 Arrays**






4. Написать метод toStringArray(), который принимает на вход массив целых чисел,  и возвращает массив типа String[] из тех же чисел (желательно не использовать метод .toString(), нужно переводить вручную)  
   Например, toStringArray({1, 2, 3, 4, 5}) -> {“1”, “2”, “3”, “4”, “5”}
5. Перегрузить метод toStringArray() параметром double[]. Этот метод должен возвращать массив типа String[] (желательно не использовать метод .toString(), нужно переводить вручную)  
   Например, toStringArray({1.1, 2.5, 3.7, 4.0, 5.5}) -> {“1.1”, “2.5”, “3.7”, “4.0”, “5.5”}



**Part 2.1 Arrays**

1. В классе *CreateArray* написать метод createIntArrayFromText(), который принимает на вход строку чисел (н-р, “3 4 1 8 10 12.3”) и возвращает массив типа int[] из этих чисел. (vожно использовать split() или toCharArray())
2. В классе *OddEvenElementsInArray* написать метод countOddEvenValuesInArray(), который принимает на вход массив целых чисел и возвращает массив int[2], который содержит количество четных и нечетных элементов входящего массива
3. В классе *OddEvenElementsInArray* написать метод createOddEvenArray(), который принимает массив целых случайных чисел, и возвращает двумерный массив (массив четных и массив нечетных чисел)
4. В классе *ManipulationsWithArrays* написать метод getTheGreaterHalf(), который принимает массив целых чисел,  и возвращает  массив из суммарно бОльшей первой или второй половины входящего массива



**Part 3 Arrays**

1. В классе *CreateArray* написать метод getPhoneNumberAndCountry(), который принимает массив из 11 целых положительных чисел от 0 до 9, и возвращает массив, который содержит номер телефона, состоящий из этих чисел,  и название страны, которой номер принадлежит.  
   Например:   
   getPhoneNumberAndCountry({1, 8, 0, 0, 1, 2, 3, 4, 5, 6, 7}) -> {“1(800)123-45-67”, “USA”}

1. В классе *CreateArray* написать метод createUniqueArray(), который принимает массив целых положительных чисел больше 0, и возвращает массив уникальных чисел.

Например, createUniqueArray({1, 2, 1, 5, 23, 90, 3, 43, 23, 1}) ->

{1, 2, 5, 23, 90, 3, 43}

1. В классе *ManipulationsWithArrays* написать метод findNonUniqueValues(), который принимает на вход массив целых положительных чисел, и возвращает количество неуникальных элементов в виде двумерного массива {{n, c}}, где n = неуникальное число и с = сколько таких чисел найдено  
   Например,   
   findNonUniqueValues({1, 2, 1, 5, 23, 90, 3, 43, 23, 1}) -> {{1, 3}, {23, 2}}
2. В классе *CreateArrays* написать метод createArrayFollowedNumber()  
   This method having in input an array of numbers   
   and a number x   
   returns an array of elements that follows the last occurrence of x in a.

For example with input {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} and x = 6 the method must return {7, 8, 9}   
  
 meanwhile with {4, 1, 4, 2} and x = 4 the method must return {2}   
  
 and if the x is not in a, then it must return empty array {} (or array with 0 lengths)  
  
 and if x is the last number in a then it must return empty array {} (or array with 0 length)

1. В классе *ManipulationsWithArrays* написать метод createUniqueArray(), который принимает на вход 2 массива int[] и возвращает объединенный массив уникальных неповторяющихся элементов

**Part 3 Algorithms**

1. Написать метод deleteNumber, который принимает на вход массив целых чисел и число number. Удалите все вхождения числа number из массива (пропусков быть не должно).

**(**{0,1,2,2,3,0,4,2}, 2) -> {0, 1, 3, 0, 4}

1. Напишите метод removeDuplicates(int[] array) , который возвращает массив, в котором удалены повторяющиеся элементы из массива array
2. Напишите метод, который принимает на вход массив double[] array и создает новый массив такой же длины double[] dupArray.   
   Метод берет значение array[i], генерирует новый случайный индекс j для этого значения в новом массиве, и присваивает значение array[i] ячейке dupArray[j] (где j - случайно сгенерированное число).   
   Последние три элемента массива array заполняют пустые ячейки в новом массиве dupArray.
3. Напишите метод getLast(array, n) , который возвращает фрагмент массива, содержащий последние ‘n’ элементов массива. Метод должен работать с массивами типа int[], double[] и String[]
4. Напишите метод descendingSort(int[] array) , который возвращает массив, элементы которого отсортированы по убыванию их значений.

**HW 9 for Beginners**

**Методы класса String для самостоятельного изучения**

* Все задания в первой части нужно выполнять в проекте Java\_05
* Создайте класс **StringMethods**
* Смотрите видео по каждому методу и решайте задание с помощью метода из видео.
* Каждое задание должно быть решено методом, который возвращает результат (не печатать ответы, а возвращать)
* Тестировать ваш код можно в классе **StringMethodsTest** (протестируйте только то, что хотите, или на что останется время)
* **Задачи, написанные жирным шрифтом** - повышенной сложности

**Видео1**: <https://youtu.be/78ZAzmNbUYQ?t=173>**Методы isEmpty(), length(), trim(), replace()**

1. Написать метод removeSpaces(), который принимает на вход строку.   
   Если строка не пустая (проверить методом класса String), то примените метод по удалению пробелов в начале и в конце строки.   
   Догадаться, каким методом из видео можно проверить, были ли пробелы в строке.  
   Если пробелы были, то метод должен удалить их и вернуть сообщение: “Лишние пробелы удалены”.

Если пробелов не было, вернуть сообщение “Пробелов не было”.

Если строка пустая (проверить методом класса String), вернуть сообщение “Строка пустая”.

Test Data:

“    Red Rover School   “ → “Лишние пробелы удалены”

“Red Rover School“ → “Пробелов не было”  
“” → “Строка пустая”

1. Написать алгоритм removeAllAs().  
   С помощью методов из видео1,  написать алгоритм, который принимает на вход строку. Если строка валидная, то метод удаляет все буквы **a** из строки, если таковые имеются. Метод возвращает обработанную строку.

*Test Data:*

 “    Red Rover School   “ →  “Red Rover School“

“panda   “ → “pnd”

“taramasalata” → “trmslt”

1. Написать алгоритм removeAllZeros().  
   С помощью методов из видео1,  написать алгоритм, который принимает на вход строку, состоящую из цифр. Если строка валидная, то метод удаляет все пробелы из строки, если таковые имеются. Метод возвращает обработанную строку, в которой нет нулей. Если в строке не было нулей, метод возвращает сообщение “This is a valid string”.

*Test Data:*

 “3504209706040000 “ →  “35429764“

“0000000111“ → “111”

1. Написать алгоритм removeAllSpaces.  
   С помощью методов из видео1,  написать алгоритм, который принимает на вход строку. Если строка валидная, то метод удаляет все пробелы из строки, если таковые имеются. Метод возвращает обработанную строку.

*Test Data:*

 “    R e d     Ro ve    r Sc   h ool   “ →  “RedRoverSchool“

“p a     n   d a   “ → “panda”

**Видео2:** <https://www.youtube.com/watch?v=thzfsPEYXfI>**Методы**

**charAt(), concat(), contains(), substring(), toLowerCase(),  toUpperCase()**

1. Напишите метод countAs(), который принимает на вход строку и считает, сколько букв а или А содержится в строке. Метод возвращает количество букв a/A,  и количество букв/знаков в слове без букв a/A. Итоговый результат должен строится с помощью метода из видео 2.

*Test Data:*“Abracadabra” → “5, 6”  
“Homenum Revelio” → “0, 15”  
“3 tarAmasAlatA” → “6, 8”

1. Напишите метод countJava(), который принимает на вход текст и проверяет, содержится ли в тексте хотя бы одно слово Java.  
   *Test Data:  
   “As of March 2022,* [*Java 18*](https://en.wikipedia.org/wiki/Java_version_history) *is the latest version, while Java 17, 11 and 8 are the current* [*long-term support*](https://en.wikipedia.org/wiki/Long-term_support) *(LTS) versions. Oracle released the last zero-cost public update for the* [*legacy*](https://en.wikipedia.org/wiki/Legacy_system) *version* [*Java 8*](https://en.wikipedia.org/wiki/Java_8) *LTS in January 2019 for commercial use, although it will otherwise still support Java 8 with public updates for personal use indefinitely. Other vendors have begun to offer* [*zero-cost builds*](https://en.wikipedia.org/wiki/OpenJDK#OpenJDK_builds) *of OpenJDK 8 and 11 that are still receiving security and other upgrades.”* → true  
     
   “99 little bugs in a code.  
   99 little bugs in a code.  
   Take one down, and patch it around.  
   235 critical bugs in the code.” → false
2. Напишите метод insertQuotes(), который принимает слово и возвращает строку, в которой это слово “обернуто” в кавычки:  
   *Test Data:  
   “*Abracadabra” →  ““Abracadabra””
3. **Напишите метод insertQuotes(), который принимает на вход две строки, и добавляет знак :  после слова “писал”,  и оборачивает в кавычки вторую строку  
   (результат строится с помощью метода concat())  
     
   *Test Data:  
   “Федор Достоевский писал”, “Надо любить жизнь больше, чем смысл жизни.” →   
   “Федор Достоевский писал: "Надо любить жизнь больше, чем смысл жизни."”***

***Задание со звездочкой:  
“Наполеон Бонапарт писал”, “В моем словаре нет слова «невозможно».” →   
"Наполеон Бонапарт писал: "В моем словаре нет слова "невозможно".""***

1. Напишите метод, кторый принимает на вход название города и исправляет написание:

*Test Data:*  
“ташкент” → “Ташкент”  
“ЧикаГО” → “Чикаго”

Видео3: <https://www.youtube.com/watch?v=6_RDHZfygGo>**indexOf(), lastIndexOf()**

1. Напишите метод, который принимает на вход строку и букву-параметр, и возвращает все, что находится между первой и последней буквой-параметром:  
   *Test Data:*  
   “Abracadabra”, “b” → “bracadab”  
   “Whippersnapper”, “p” → “ppersnapp”
2. Напишите метод, который принимает на вход слово, и возвращает true, если слово начинается и заканчивается на одинаковую букву, и false иначе

*Test Data:*  
 “Abracadabra” → true  
 “Whippersnapper” → false

Видео4: <https://www.youtube.com/watch?v=AY5b6iIk8mM>

1. Напишите метод, который принимает на вход строку из двух слов, разделенных пробелом, и возвращает последнее слово

*Test Data:*  
 “Red Rover” → “Rover”

1. Написать метод, который принимает строку, и 2 индекса. Метод удаляет все, что находится между двумя индексами включительно

*Test Data:*  
 “Red rover”, 1, 4 → “Rover”

Видео5: <https://youtu.be/SBByYcrsXu0?t=42>**split()**

1. Напишите метод, который принимает на вход предложение и возвращает слова из этого предложения в виде массива слов  
   *Test Data:*  
   “QA for Everyone” → {“QA”, “for”, “Everyone”}

“Александр Сергеевич Пушкин” → {“Александр”, “Сергеевич”, “Пушкин”}

1. **Написать метод, который принимает на вход предложение, которое состоит из имени, фамилии, отчества и возвращает текст:  
   *Test Data:*  
   “Александр Сергеевич Пушкин” →**

**“Имя: Александр**

**Отчество: Сергеевич**

**Фамилия: Пушкин”**

Без видео, используя документацию Oracle и гугл, найти методы класса String, которые помогут выполнить задания

1. Написать метод, который принимает на вход слово и число n,  и возвращает строку, где слово повторяется n раз.

*Test Data:*  
“one”, 5 → “oneoneoneoneone”

1. Напишите метод, который принимает строку и index, и возвращает численное значение буквы, которая находится на позиции index (использовать только один метод класса String)
2. Написать метод, который принимает 3 слова, и сравнивает их методом, который считает, что прописные и заглавные буквы - это одни и те же буквы. Метод возвращает true, если все слова одинаковы, и метод возвращает false, если слова не одинаковы:

*Test Data:*  
“one”, “One”, “ONE”  → true

“one”, “Один”, “ONE” → false