**Задачи:**

1. Дан массив чисел, посчитать процент уникальных чисел.

\*процент уникальных чисел = количество уникальных чисел \* 100 / общее количество чисел в массиве.

1. Найти пересечение двух множеств

Пример:

set1= [1,2,3,4]

set2= [3,5,6,7]

Вывод в консоль:

[3]

1. Найти пересечения слов в массивах и указать их общее количество.

Пример:

Mas1= [“qwe”,”asd”,”qwe”,”x”]

Mas2=[“qwe”,”v”]

Результат:

qwe=3

1. Напишите интерфейс Converter для конвертации из градусов по Цельсию в

Кельвины, Фаренгейты. У классов наследников необходимо переопределить метод интерфейса, для валидного перевода величин. Метод для конвертации назовите "convertValue".

**Задачи со звездой:**

**Бинарное сложение:**

public static String zad4(String a, String b) {

if(a.length()<b.length()){

return zad4(b,a);

}

int carry=0;

int j = b.length()-1;

String ans="";

“1111”

“1011”

for (int i = a.length()-1; i >=0; i--) {

if(a.charAt(i)=='1'){

carry++;

}

if(j>=0 && b.charAt(j)=='1'){

carry++;

}

ans= carry % 2 + ans;

j--;

carry/=2;

}

if(carry>0){

ans= carry%2+ans;

}

return ans;

}

**Перестановки массива:**

public static void main(String[] args) {  
 int[] arr = {1, 2, 3};  
 int count = 1;  
 for (int i = 2; i <= arr.length; i++) {  
 count \*= i;  
 }  
 System.*out*.println(count + " - перестановок");  
 int maxIndex = arr.length - 1;  
 for (int i = 0; i < count; i++) {  
 int temp = arr[maxIndex];  
 arr[maxIndex] = arr[maxIndex - 1];  
 arr[maxIndex - 1] = temp;  
 System.*out*.println(Arrays.*toString*(arr));  
 if (maxIndex < 2) {  
 maxIndex = arr.length - 1;  
 } else {  
 maxIndex--;  
 }  
 }  
}

**Валидация доски:**

public static void main(String[] args) {  
 char[][] board = {  
 {'5', '3', '.', '.', '7', '.', '.', '.', '.'},  
 {'6', '.', '.', '1', '9', '5', '.', '.', '.'},  
 {'.', '9', '8', '.', '.', '.', '.', '6', '.'},  
 {'8', '.', '.', '.', '6', '.', '.', '.', '3'},  
 {'4', '.', '.', '8', '.', '3', '.', '.', '1'},  
 {'7', '.', '.', '.', '2', '.', '.', '.', '6'},  
 {'.', '6', '.', '.', '.', '.', '2', '8', '.'},  
 {'.', '.', '.', '4', '1', '9', '.', '.', '5'},  
 {'.', '.', '.', '.', '8', '.', '.', '7', '9'}};  
  
 System.*out*.println("Validete:" + *isValidSudoku*(board));  
}  
  
*/\*\*  
 \* Проверяет валидность char матрицы по условиям:  
 \* 1) For - проверяет наличие повторяющихся символов в строках  
 \* 2) For - проверяет наличие повторяющихся символов в столбцах  
 \* 3) For - проверяет наличие повторяющихся символов в блоках 3x3  
 \*/*public static boolean isValidSudoku(char[][] board) {  
 for (int i = 0; i < 9; i++) {  
 int[] masPovtor = new int[10];  
 boolean q = true;  
 for (int j = 0; j < 9; j++) {  
 int p = Character.*getNumericValue*(board[i][j]);  
 if (board[i][j] > '0' && board[i][j] <= '9')  
 if (masPovtor[p] < 1)  
 masPovtor[p]++;  
 else return false;  
 }  
 }  
 for (int i = 0; i < 9; i++) {  
 int[] masPovtor = new int[10];  
 boolean q = true;  
 for (int j = 0; j < 9; j++) {  
 int p = Character.*getNumericValue*(board[j][i]);  
 if (board[j][i] > '0' && board[j][i] <= '9')  
 if (masPovtor[p] < 1)  
 masPovtor[p]++;  
 else return false;  
 }  
 }  
  
 int[] masPovtor = new int[30];//разделил матрицу char на 3 блока по i => заводить 9 массивов  
 int[] masPovtor2 = new int[30];//думаю не очень удобно поэтому расширил объем каждого массива  
 int[] masPovtor3 = new int[30];//в индексах массивов с 0 по 9 лежат значения первых трех блоков  
 int k = -10; //а последующий ( 2-й с 10 по 19), 3-й с 20 по 29  
 boolean q = true;  
 for (int i = 0; i < 9; i++) {  
 if (i % 3 == 0)  
 k += 10;// разделение блоков по строкам  
 for (int j = 0; j < 9; j++) {  
 int p = Character.*getNumericValue*(board[j][i]);  
 if (board[j][i] > '0' && board[j][i] <= '9') {  
 if (j < 3)// разделение блоков по столбцам  
 if (masPovtor[p + k] < 1)  
 masPovtor[p + k]++;  
 else return false;  
 if (j >= 3 && j <= 5)  
 if (masPovtor2[p + k] < 1)  
 masPovtor2[p + k]++;  
 else return false;  
 if (j > 5)  
 if (masPovtor3[p + k] < 1)  
 masPovtor3[p + k]++;  
 else return false;  
 }  
 }  
 }  
 return true;  
}