МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ УПРАВЛІННЯ»

ЗВІТ

з лабораторної роботи № 1

з навчальної дисципліни

«основи програмування Java (двв)»

ВИКОНАВ

Студент групи

ПЕРЕВІРИВ

доц каф. ПІІТУ к.т.н.

доцент Валерій ВОЛОВЩИКОВ

Харків 2022

**Тема:** Архітектура та синтаксис Java

**Завдання 1. Індивідуальне завдання**

Заповнити двовимірний масив цілих випадковими додатними значеннями відповідно до правила, наведеного в таблиці. Заповнити одновимірний масив рядків повторенням певного символу кількістю, вказаною в таблиці. Здійснити сортування масиву рядків за правилом, вказаним в таблиці.

Вивести отриманий масив рядків.

Реалізувати два підходи: традиційний, побудований на циклах і роботі з окремими елементами і через функції класу Arrays (без циклів).

Додати до класу та окремих функцій коментарі Javadoc.

Номер варіанту: 12

Правило заповнення двовимірного масиву: випадкове парне число від 8 до 26

Кількість елементів у масиву рядків: мінімальний елемент стовпця

Критерій сортування масиву рядків: за збільшенням довжини

**Тестові дані для завдання №1:**

Кількість рядків: 2

Кількість стовпців: 5

Символ для повторення: «#»

**Програмний код до завдання №1:**

package task1;  
  
*/\*\*  
 \* Клас Main<br>  
 \* Використовується для тестування програми.  
 \*/*public class Main {  
 */\*\*  
 \* Метод main<br>  
 \* У методі main викликаються методи для тестування роботи класів:  
 \* LoopsImplementation.test() традиційний підхід з циклами та роботі з окремими елементами;  
 \* ArraysClassImplementation.test - через функції класу Arrays (без циклів).  
 \*  
 \** ***@param*** *args аргументи командної строки  
 \*/* public static void main(String[] args) {  
 LoopsImplementation.*test*();  
 System.*out*.print("\n+-------------------------+\n\n");  
 ArraysClassImplementation.*test*();  
 }  
}

package task1;  
  
import java.util.Random;  
import java.util.Scanner;  
  
*/\*\*  
 \* Клас LoopsImplementation<br>  
 \* У класі реалізовані методи для виконання завдання через традиційний підхід з циклами.  
 \*/*class LoopsImplementation {  
 */\*\*  
 \* Метод LoopsImplementation.test<br>  
 \* Використовується для демонстрації роботи реалізації через традиційний підхід через цикли та роботу з окремими  
 \* елементами.<br>  
 \* У методі створюється двовимірний масив цілих чисел, який заповнюється випадковими додатними значеннями відповідно  
 \* до варіанту та виводиться на екран. Потім створюється масив рядків, який заповнюється повторенням певного символу  
 \* кількістю, вказаною у таблиці та виводиться, після чого сортується за правилом, вказаним в таблиці та знов виводиться.  
 \*/* public static void test() {  
 int[][] intMatrix = Utils.*createIntMatrix*();  
 *fillRandomNumbers*(intMatrix);  
 System.*out*.println("Integer matrix:");  
 *printIntMatrix*(intMatrix);  
 String[] stringArray = *createStringArray*(intMatrix);  
 System.*out*.println("String array before sorting:");  
 *printStringArray*(stringArray);  
 *sortStringArrayByLengthAscending*(stringArray);  
 System.*out*.println("String array after sorting:");  
 *printStringArray*(stringArray);  
 }  
  
 */\*\*  
 \* Метод fillRandomNumbers<br>  
 \* Заповнює двовимірний масив цілих чисел випадковими додатними значеннями відповідно до завдання (парними числами  
 \* від 8 включно до 26 невключно).  
 \*  
 \** ***@param*** *intMatrix двовимірний масив цілих чисел  
 \*/* public static void fillRandomNumbers(int[][] intMatrix) {  
 for (int i = 0; i < intMatrix.length; i++) {  
 for (int j = 0; j < intMatrix[i].length; j++) {  
 intMatrix[i][j] = (new Random().nextInt(26 / 2 - 8 / 2) + 8 / 2) \* 2;  
 }  
 }  
 }  
  
 */\*\*  
 \* Метод createStringArray<br>  
 \* Створює масив строк довжиною як кількість стовпців двовимірного масиву цілих чисел. Строки заповнюються  
 \* повторенням певного символу, який задає користувач, кількістю вказаною у таблиці (мінімальний елемент стовпця).  
 \*  
 \** ***@param*** *intMatrix двовимірний масив цілих чисел  
 \** ***@return*** *масив строк  
 \*/* public static String[] createStringArray(int[][] intMatrix) {  
 String[] stringArray = new String[intMatrix[0].length];  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Character: ");  
 char ch = in.next().charAt(0);  
 for (int i = 0; i < stringArray.length; i++) {  
 stringArray[i] = "";  
 for (int j = 0; j < Utils.*minItemInColumn*(intMatrix, i); j++) {  
 stringArray[i] += ch;  
 }  
 }  
 return stringArray;  
 }  
  
 */\*\*  
 \* Метод printIntMatrix<br>  
 \* Виводить в консоль двовимірний масив цілих чисел.  
 \*  
 \** ***@param*** *intMatrix двовимірний масив цілих чисел  
 \*/* public static void printIntMatrix(int[][] intMatrix) {  
 for (int i = 0; i < intMatrix.length; i++) {  
 for (int j = 0; j < intMatrix[i].length; j++) {  
 System.*out*.print(intMatrix[i][j] + "\t");  
 }  
 System.*out*.println();  
 }  
 }  
  
 */\*\*  
 \* Метод printStringArray<br>  
 \* Виводить в консоль масив строк.  
 \*  
 \** ***@param*** *stringArray масив строк  
 \*/* public static void printStringArray(String[] stringArray) {  
 for (int i = 0; i < stringArray.length; i++) {  
 System.*out*.println(stringArray[i]);  
 }  
 }  
  
 */\*\*  
 \* Метод sortStringArrayByLengthAscending<br>  
 \* Сортує масив строк за збільшенням довжини.  
 \*  
 \** ***@param*** *stringArray масив строк  
 \*/* public static void sortStringArrayByLengthAscending(String[] stringArray) {  
 for (int i = 0; i < stringArray.length - 1; i++) {  
 for (int j = i + 1; j < stringArray.length; j++) {  
 if (stringArray[i].length() > stringArray[j].length()) {  
 String tempString = stringArray[i];  
 stringArray[i] = stringArray[j];  
 stringArray[j] = tempString;  
 }  
 }  
 }  
 }  
}

package task1;  
  
import java.util.\*;  
  
*/\*\*  
 \* Клас ArraysClassImplementation<br>  
 \* У класі реалізовані методи для виконання завдання через методи класу Arrays.  
 \*/*class ArraysClassImplementation {  
 */\*\*  
 \* Метод ArraysClassImplementation.test<br>  
 \* Використовується для демонстрації роботи реалізації через функції Arrays (без циклів).<br>  
 \* У методі створюється двовимірний масив цілих чисел, який заповнюється випадковими додатними значеннями відповідно  
 \* до варіанту та виводиться на екран. Потім створюється масив рядків, який заповнюється повторенням певного символу  
 \* кількістю, вказаною у таблиці та виводиться, після чого сортується за правилом, вказаним в таблиці та знов виводиться.  
 \*/* public static void test() {  
 int[][] intMatrix = Utils.*createIntMatrix*();  
 Arrays.*setAll*(intMatrix, i -> *getRandomIntArray*(intMatrix[0].length));  
 System.*out*.println("Integer matrix:");  
 System.*out*.println(Arrays.*deepToString*(intMatrix));  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Character: ");  
 char ch = in.next().charAt(0);  
 String[] stringArray = new String[intMatrix[0].length];  
 Arrays.*setAll*(stringArray, i -> String.*valueOf*(ch).repeat(Utils.*minItemInColumn*(intMatrix, i)));  
 System.*out*.println("String array before sorting:");  
 System.*out*.println(Arrays.*toString*(stringArray));  
 Arrays.*sort*(stringArray, Comparator.*comparingInt*(String::length));  
 System.*out*.println("String array after sorting:");  
 System.*out*.println(Arrays.*toString*(stringArray));  
 }  
  
 */\*\*  
 \* Метод getRandomIntArray<br>  
 \* Повертає одновимірний масив цілих чисел, що заповнений випадковими додатними значеннями відповідно до завдання  
 \* (парними числами від 8 включно до 26 невключно).  
 \*  
 \** ***@param*** *length довжина одновимірного масиву цілих чисел  
 \** ***@return*** *одновимірний масив цілих чисел, заповнений відповідними випадковими значеннями  
 \*/* public static int[] getRandomIntArray(int length) {  
 int[] randomIntArray = new int[length];  
 Arrays.*setAll*(randomIntArray, i -> (new Random().nextInt(26 / 2 - 8 / 2) + 8 / 2) \* 2);  
 return randomIntArray;  
 }  
}

package task1;  
  
import java.util.Scanner;  
  
*/\*\*  
 \* Клас Utils<br>  
 \* Містить спільні для обох класів методи (щоб не повторювався код).  
 \*/*class Utils {  
 */\*\*  
 \* Метод createIntMatrix<br>  
 \* Створює двовимірний масив цілих чисел, кількість рядків та стовпців якого встановлює користувач.  
 \*  
 \** ***@return*** *двовимірний масив цілих  
 \*/* public static int[][] createIntMatrix() {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Rows: ");  
 int rows = in.nextInt();  
 System.*out*.print("Columns: ");  
 int columns = in.nextInt();  
 return new int[rows][columns];  
 }  
  
 */\*\*  
 \* Метод minItemInColumn<bp>  
 \* Знаходить мінімальний елемент вказаного стовпця у двовимірному масиві цілих чисел.  
 \*  
 \** ***@param*** *intMatrix двовимірний масив цілих чисел  
 \** ***@param*** *column номер стовпця  
 \** ***@return*** *мінімальний елемент вказаного стовпця  
 \*/* public static int minItemInColumn(int[][]intMatrix, int column) {  
 int min = intMatrix[0][column];  
 for (int i = 0; i < intMatrix.length; i++) {  
 if (intMatrix[i][column] < min) {  
 min = intMatrix[i][column];  
 }  
 }  
 return min;  
 }  
}

**Результат роботи програми з тестовими даними завдання №1:**



**Завдання 2. Ератосфенове решето**

Заповнити масив із трьохсот цілих чисел послідовними додатними значеннями. Замінити всі значення, що не є простими числами, деяким від'ємним значенням. Для цього послідовно виключати всі числа – дільники інших чисел. Вивести на екран додатні значення, що залишилися, (прості числа).

У програмі не застосовувати ділення та знаходження залишку від ділення.

**Тестові дані для завдання №2:**

Прості числа з послідовності з 300 чисел починаючи з 10.

Очікуваний результат: 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307.

**Програмний код до завдання №2:**

package task2;  
  
public class Main {  
 public static void main(String[] args) {  
 EratosthenesSieve.*printPrimeNumbersOf300From*(10);  
 }  
}

package task2;  
  
import java.util.Arrays;  
  
class EratosthenesSieve {  
 public static void printPrimeNumbersOf300From(int from) {  
 boolean[] boolPrimeNumbers = *getBoolPrimeNumbers*(from);  
  
 int[] numbers = new int[from + 301];  
 *setNumbers*(numbers);  
  
 *setCompositeNumbersToNegative*(numbers, boolPrimeNumbers);  
  
 *printPositiveNumbersFrom*(numbers, from);  
 }  
  
 public static boolean[] getBoolPrimeNumbers(int n) {  
 boolean[] boolPrimeNumbers = new boolean[n + 301];  
 Arrays.*fill*(boolPrimeNumbers, true);  
 boolPrimeNumbers[0] = false;  
 boolPrimeNumbers[1] = false;  
 for (int i = 2; i \* i < n + 301; i++) {  
 if (boolPrimeNumbers[i] == true) {  
 for (int j = i \* i; j < n + 301; j += i) {  
 boolPrimeNumbers[j] = false;  
 }  
 }  
 }  
 return boolPrimeNumbers;  
 }  
  
 public static void setNumbers(int[] numbers) {  
 for (int i = 0; i < numbers.length; i++) {  
 numbers[i] = i;  
 }  
 }  
  
 public static void setCompositeNumbersToNegative(int[] numbers, boolean[] boolPrimeNumbers) {  
 for (int i = 0; i < numbers.length; i++) {  
 if (boolPrimeNumbers[i] == false) {  
 numbers[i] \*= -1;  
 }  
 }  
 }  
  
 public static void printPositiveNumbersFrom(int[] numbers, int from) {  
 for (int i = from; i < numbers.length; i++) {  
 if (numbers[i] > 0) {  
 System.*out*.print(numbers[i] + " ");  
 }  
 }  
 }  
}

**Результат роботи програми з тестовими даними завдання №2:**



**Завдання 3: Знаходження чисел Фібоначчі**

Реалізувати функцію обчислення чисел Фібоначчі (до 92-го числа включно) з використанням допоміжного масиву (статичного поля). Параметр функції – номер числа Фібоначчі. Пошук чисел Фібоначчі здійснюється за таким правилом:

F(1) = F(2) = 1; F(n) = F(n - 2) + F(n - 1)

Під час першого виклику функції масив заповнюється до необхідного числа. Під час наступних викликів число або повертається з масиву, або обчислюється з використанням останніх двох чисел, що зберігаються у масиві з подальшим заповненням масиву. Використовувати тип long для представлення чисел.

Здійснити тестування функції для різних значень номерів, що вводяться у довільному порядку.

**Тестові дані для завдання №3:**

Порядковий номер числа Фібоначчі: 8

Потрібний результат: 21

Порядковий номер числа Фібоначчі: 6

Потрібний результат: 8

Порядковий номер числа Фібоначчі: 12

Потрібний результат: 144

**Програмний код до завдання №3:**

package task3;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println(FibonacciNumbers.*getFibonacciNumber*(8));  
 System.*out*.println(FibonacciNumbers.*getFibonacciNumber*(6));  
 System.*out*.println(FibonacciNumbers.*getFibonacciNumber*(12));  
 }  
}

package task3;  
  
class FibonacciNumbers {  
 private static final long[] *fibonacciNumbers* = new long[92];  
 private static int *greatestCounted* = 1;  
  
 static {  
 *fibonacciNumbers*[0] = 1;  
 *fibonacciNumbers*[1] = 1;  
 }  
  
 public static long getFibonacciNumber(int n) {  
 if (n - 1 > *greatestCounted*) {  
 for (int i = *greatestCounted* + 1; i <= n - 1; i++) {  
 *fibonacciNumbers*[i] = *fibonacciNumbers*[i - 1] + *fibonacciNumbers*[i - 2];  
 }  
 *greatestCounted* = n - 1;  
 }  
  
 return *fibonacciNumbers*[n - 1];  
 }  
}

**Результат роботи програми з тестовими даними завдання №3:**



**Завдання 4. Вирівнювання рядку**

Прочитати аргумент командного рядка і додати в нього пропуски (space characters) так, щоб довжина рядка дорівнювала заданому числу. Пропуски додавати рівномірно між словами (за можливості).

**Тестові дані для завдання №4:**

1) Аргумент командного рядка: “ ”

Довжина: 3

Задана довжина: 7

Очікуваний результат: “ ”

2) Аргумент командного рядка: “argument1 argument2 argument3”

Довжина: 29

Задана довжина: 33

Очікуваний результат: “argument1 argument2 argument3”

3) Аргумент командного рядка: “ argument1 argument2 argument3”

Довжина: 35

Задана довжина: 37

Очікуваний результат: “"argument1 argument2 argument3”

**Програмний код до завдання №4:**

package task4;  
  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("\"" + args[0] + "\" length: " + args[0].length());  
 System.*out*.print("Enter needed length: ");  
 Scanner in = new Scanner(System.*in*);  
 int length = in.nextInt();  
 in.close();  
 String alignedString = AlignmentOfString.*alignString*(args[0], length);  
 System.*out*.println("\"" + alignedString + "\" length: " + alignedString.length());  
 }  
}

package task4;  
  
import java.util.StringTokenizer;  
  
class AlignmentOfString {  
 public static String alignString(String stringToAlign, int length) {  
 StringTokenizer stringTokenizer = new StringTokenizer(stringToAlign);  
 if (!stringTokenizer.hasMoreTokens()) {  
 return stringToAlign + " ".repeat(length - stringToAlign.length());  
 }  
 int spacesInString = 0;  
 for (int i = 0; i < stringToAlign.length(); i++) {  
 if (stringToAlign.charAt(i) == ' ') {  
 spacesInString++;  
 }  
 }  
 String alignedString = "";  
 int needToAddSpaces = length - (stringToAlign.length() - spacesInString);  
 if (stringTokenizer.countTokens() == 1) {  
 if (needToAddSpaces % 2 != 0)  
 alignedString = " " + alignedString;  
 alignedString += " ".repeat(needToAddSpaces / 2);  
 alignedString += stringTokenizer.nextToken();  
 alignedString += " ".repeat(needToAddSpaces / 2);  
 }  
 else {  
 int addBetweenEachWords = needToAddSpaces / (stringTokenizer.countTokens() - 1);  
 int remainderFromDivision = needToAddSpaces % (stringTokenizer.countTokens() - 1);  
 for (int i = 0; i <= stringTokenizer.countTokens(); i++) {  
 String token = stringTokenizer.nextToken();  
 alignedString += token;  
 alignedString += " ".repeat(addBetweenEachWords);  
 if (remainderFromDivision > 0) {  
 alignedString += " ";  
 remainderFromDivision--;  
 }  
 }  
 alignedString += stringTokenizer.nextToken();  
 }  
 return alignedString;  
 }  
}

**Результат роботи програми з тестовими даними завдання №4:**

1)



2)



3)



**Висновки:**

В ході виконання лабораторної роботи №1 з предмету “Основи програмування Java” я ознайомився з основними принципами архітектури та синтаксису у мові програмування Java, з її базовими засобами, пакетами та функціями, консольним введенням та виведенням, типами-посиланнями, класами, масивами, деякими стандартними класами, рядками, класами-обгортками тощо, а також з інтегрованим середовищем розробки IntelliJ IDEA.