Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
Федерального государственного бюджетного образовательного учреждения
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет ИТР

Кафедра ПИн

ЛАБОРАТОРНАЯ РАБОТА №2

По Сравнительный анализ языков программирования

Тема Работа с массивами.

Руководитель

Кульков Я.Ю.

(фамилия, инициалы)

(подпись) (дата)

Студент ПИН - 121

(группа)

Мочалин Н.А.

(фамилия, инициалы)

(подпись) (дата)

Муром 2023

Лабораторная работа №2

Тема: Работа с массивами.

Цель работы: Изучение принципов работы массивов на примере алгоритмов сортировки.

Задачи:

1.Изучение интерфейса среды программирования IntelliJ IDEA, структуры проекта.

2. Создание простейших консольных приложений.

3. Изучение способов отладки программ.

Ход работы:

Задание на лабораторную работу:

I. Изучение времени работы алгоритмов

1. Вынести каждый из рассмотренных методов в отдельные функции.
2. Сгенерировать три массива с количеством элементов не менее 10 000:
   - Полностью отсортированный массив значений
   - Полностью случайных набор значений
   - Отсортированный массив, в котором первые 10% от общего числа элементов случайные
3. Замерить время сортировки для каждого из массивов используя подготовленные функции.
4. Замерить время сортировки массивов методом sort() класса Arrays 5. Сравнить результаты и сделать выводы.

II. Реализация алгоритмов работы с массивами:

1. Напишите функцию int[] removeDuplicates(int[] array), которая возвращает массив, в котором удалены повторяющиеся элементы из массива.
2. Напишите функцию int[] getFirst(int[] array, int n), которая возвращает фрагмент массива, содержащий первые ‘n’ элементов массива.
3. Напишите функцию int[] getLast(int[] array, int n), которая возвращает фрагмент массива, содержащий последние ‘n’ элементов массива.
4. Напишите функцию int countIdentic(int[] array), которая возвращает количество повторяющихся элементов в массиве.
   - При реализации методов не использовать классы коллекций и их методов, а также сторонних библиотек классов.
   - Продемонстрировать работу разработанных методов.

I. Изучение времени работы алгоритмов

Листинг **App.java**

```
import java.util.Arrays;
import java.util.Random;

public class App {
    public static void main(String[] args) throws Exception {
        int count = 70000;
        int[] array = new int[count];
        Random random = new Random();

        for (int i = 0; i < count; i++) {
            array[i] = random.nextInt();
        }

        int[] arrayClone = array.clone();
        Arrays.sort(arrayClone);

        int[] arrayTenPercentRandom = arrayClone.clone();
        for (int i = 0; i < count * 0.1; i++) {
            arrayTenPercentRandom[i] = random.nextInt();
        }
        System.out.println("");
        System.out.println("Not sort array");
        System.out.println("");

        double time = System.currentTimeMillis();
        bubbleSort(array.clone());
        System.out.print("BubbleSort: ");
        System.out.println(System.currentTimeMillis() - time);

        time = System.currentTimeMillis();
        insertionSort(array.clone());
        System.out.print("Insertion Sort: ");
        System.out.println(System.currentTimeMillis() - time);

        time = System.currentTimeMillis();
        selectionSort(array.clone());
        System.out.print("Selection Sort: ");
        System.out.println(System.currentTimeMillis() - time);

        time = System.currentTimeMillis();
        mergeSort(array.clone(), 0, count - 1);
        System.out.print("Merge Sort: ");
        System.out.println(System.currentTimeMillis() - time);

        time = System.currentTimeMillis();
        quickSort(array.clone(), 0, count - 1);
        System.out.print("Quick Sort: ");
        System.out.println(System.currentTimeMillis() - time);

        time = System.currentTimeMillis();
        Arrays.sort(array.clone());
        System.out.print("Arrays.sort: ");
        System.out.println(System.currentTimeMillis() - time);
```

```
System.out.println("");
System.out.println("Sort array");
System.out.println("");
time = System.currentTimeMillis();
bubbleSort(arrayClone);
System.out.print("BubbleSort: ");
System.out.println(System.currentTimeMillis() - time);

time = System.currentTimeMillis();
insertionSort(arrayClone);
System.out.print("Insertion Sort: ");
System.out.println(System.currentTimeMillis() - time);

time = System.currentTimeMillis();
selectionSort(arrayClone);
System.out.print("Selection Sort: ");
System.out.println(System.currentTimeMillis() - time);
time = System.currentTimeMillis();
mergeSort(arrayClone, 0, count - 1);
System.out.print("Merge Sort: ");
System.out.println(System.currentTimeMillis() - time);

time = System.currentTimeMillis();
quickSort(arrayClone, 0, count - 1);
System.out.print("Quick Sort: ");
System.out.println(System.currentTimeMillis() - time);

time = System.currentTimeMillis();
Arrays.sort(arrayClone);
System.out.print("Arrays.sort: ");
System.out.println(System.currentTimeMillis() - time);

System.out.println("");
System.out.println("Ten percent random array");
System.out.println("");
time = System.currentTimeMillis();
bubbleSort(arrayTenPercentRandom.clone());
System.out.print("BubbleSort: ");
System.out.println(System.currentTimeMillis() - time);

time = System.currentTimeMillis();
insertionSort(arrayTenPercentRandom.clone());
System.out.print("Insertion Sort: ");
System.out.println(System.currentTimeMillis() - time);

time = System.currentTimeMillis();
selectionSort(arrayTenPercentRandom.clone());
System.out.print("Selection Sort: ");
System.out.println(System.currentTimeMillis() - time);

time = System.currentTimeMillis();
mergeSort(arrayTenPercentRandom.clone(), 0, count - 1);
System.out.print("Merge Sort: ");
System.out.println(System.currentTimeMillis() - time);

time = System.currentTimeMillis();
quickSort(arrayTenPercentRandom.clone(), 0, count - 1);
System.out.print("Quick Sort: ");
System.out.println(System.currentTimeMillis() - time);
```

```
        time = System.currentTimeMillis();
        Arrays.sort(arrayTenPercentRandom.clone());
        System.out.print("Arrays.sort: ");
        System.out.println(System.currentTimeMillis() - time);
    }

    private static void bubbleSort(int[] array) {
        boolean sorted = false;
        int temp;
        while (!sorted) {
            sorted = true;
            for (int i = 0; i < array.length - 1; i++) {
                if (array[i] > array[i + 1]) {
                    temp = array[i];
                    array[i] = array[i + 1];
                    array[i + 1] = temp;
                    sorted = false;
                }
            }
        }
    }

    private static void insertionSort(int[] array) {
        for (int i = 1; i < array.length; i++) {
            int current = array[i];
            int j = i - 1;
            while (j >= 0 && current < array[j]) {
                array[j + 1] = array[j];
                j--;
            }
            array[j + 1] = current;
        }
    }

    private static void selectionSort(int[] array) {
        for (int i = 0; i < array.length; i++) {
            int min = array[i];
            int minId = i;
            for (int j = i + 1; j < array.length; j++) {
                if (array[j] < min) {
                    min = array[j];
                    minId = j;
                }
            }
            // замена
            int temp = array[i];
            array[i] = min;
            array[minId] = temp;
        }
    }

    public static void mergeSort(int[] array, int left, int right) {
        if (right <= left)
            return;
        int mid = (left + right) / 2;
        mergeSort(array, left, mid);
        mergeSort(array, mid + 1, right);
        merge(array, left, mid, right);
    }
```

```
private static void merge(int[] array, int left, int mid, int right) {
    int lengthLeft = mid - left + 1;
    int lengthRight = right - mid;
    int leftArray[] = new int[lengthLeft];
    int rightArray[] = new int[lengthRight];
    // копируем отсортированные массивы во временные
    for (int i = 0; i < lengthLeft; i++)
        leftArray[i] = array[left + i];
    for (int i = 0; i < lengthRight; i++)
        rightArray[i] = array[mid + i + 1];
    // итераторы содержат текущий индекс временного подмассива
    int leftIndex = 0;
    int rightIndex = 0;
    // копируем из leftArray и rightArray обратно в массив
    for (int i = left; i < right + 1; i++) {
        // если остаются нескопированные элементы в R и L, копируем минимальный
        if (leftIndex < lengthLeft && rightIndex < lengthRight) {
            if (leftArray[leftIndex] < rightArray[rightIndex]) {
                array[i] = leftArray[leftIndex];
                leftIndex++;
            } else {
                array[i] = rightArray[rightIndex];
                rightIndex++;
            }
        }
        // если все элементы были скопированы из rightArray, скопировать остальные из
        // leftArray
        else if (leftIndex < lengthLeft) {
            array[i] = leftArray[leftIndex];
            leftIndex++;
        }
        // если все элементы были скопированы из leftArray, то скопировать остальные из
        // rightArray
        else if (rightIndex < lengthRight) {
            array[i] = rightArray[rightIndex];
            rightIndex++;
        }
    }
}

private static int partition(int[] array, int begin, int end) {
    int pivot = end;
    int counter = begin;
    for (int i = begin; i < end; i++) {
        if (array[i] < array[pivot]) {
            int temp = array[counter];
            array[counter] = array[i];
            array[i] = temp;
            counter++;
        }
    }
    int temp = array[pivot];
    array[pivot] = array[counter];
    array[counter] = temp;
    return counter;
}
```

```
    private static void quickSort(int[] array, int begin, int end) {
        if (end <= begin)
            return;
        int pivot = partition(array, begin, end - 1);
        quickSort(array, begin, pivot - 1);
        quickSort(array, pivot + 1, end);
    }

}
```

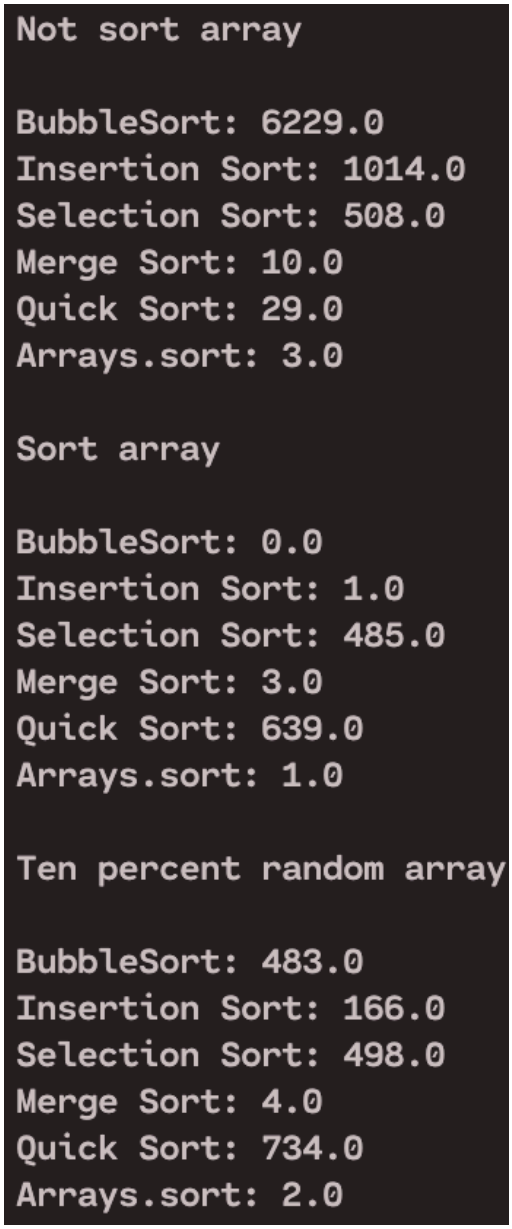

Рисунок 1 - Замеры времени
сортировки разными методами

Из выше представленных результатов замера можно сделать вывод что встроенный метод Arrays.sort самый быстрый среди всех представленных, а Merge.sort самый быстрый среди реализованных на языке java.

II. Реализация алгоритмов работы с массивами

Листинг **Methods.java**

```
import java.util.Arrays;

public class Methods {
    public static void main(String[] args) throws Exception {
        int[] arrayConsistDuplicates = {1, 1, 1, 5, 5, 2, 2, 6, 7, 8, 7, 6, 5, 4, 5, 6, 7, 2, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9, 8,
7, 6, 5, 4, 3, 2, 1};
        System.out.println("Изночальный массив: " + Arrays.toString(arrayConsistDuplicates));
        System.out.println("Массив с удалёнными дубликатами: " +
Arrays.toString(removeDuplicates(arrayConsistDuplicates)));
        System.out.println("Первые 5 элементов массива: " + Arrays.toString(getFirst(arrayConsistDuplicates,
5)));
        System.out.println("Последнии 10 элементов массива: " +
Arrays.toString(getLast(arrayConsistDuplicates, 10)));
        System.out.println("Количество уникальных элементов массива: " +
countIdentic(arrayConsistDuplicates));
    }
    private static int[] removeDuplicates(int[] array) {
        int[] numbers = new int[array.length];
        int currenIndex = 0;
        for (int indexArray = 0; indexArray < array.length; indexArray++) {
            boolean isDuplicate = false;
            for (int indexNumbers = 0; indexNumbers < numbers.length; indexNumbers++) {
                if (numbers[indexNumbers] == array[indexArray]) {
                    isDuplicate = true;
                    break;
                }
            }
            if (!isDuplicate) {
                numbers[currenIndex] = array[indexArray];
                currenIndex++;
            }
        }
        int[] result = new int[currenIndex];
        result = Arrays.copyOf(numbers, currenIndex);
        return result;
    }
    private static int[] getFirst(int[] array, int n){
        return Arrays.copyOf(array, n);
    }
    private static int[] getLast(int[] array, int n){
        return Arrays.copyOfRange(array, array.length - n, array.length);
    }
    private static int countIdentic(int[] array){
        return array.length - removeDuplicates(array).length;
    }
}
```

```
Изночальный массив: [1, 1, 1, 5, 5, 2, 2, 6, 7, 8, 7, 6, 5, 4, 5, 6, 7, 2, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9, 8, 7, 6, 5, 4, 3, 2, 1]
Массив с удалёнными дубликатами: [1, 5, 2, 6, 7, 8, 4, 3, 9]
Первые 5 элементов массива: [1, 1, 1, 5, 5]
Последнии 10 элементов массива: [9, 9, 8, 7, 6, 5, 4, 3, 2, 1]
Количество уникальных элементов массива: 27
```

Рисунок 2 - результат работы Methods.java

Вывод: В ходе работы были изучены принципы работы массивов на примере алгоритмов сортировки.