6 Лабораторная работа №6 **Многопоточность в Java**

6.1. Цель работы

Изучить основные приемы работы с потоками в Java и получить практические навыки написания многопоточных Java-приложений. приложений.

6.2. Постановка задачи

Был выдан вариант 2.

В главном классе реализовать статический метод сортировки массива (направление и метод сортировки по варианту) и вывода результата сортировки на экран с помощью System.out.println(...), формат вывода: "имя_текущего_потока: [отсортированный_массив]". Сигнатура функции сортировки: printSorted(int[] array).

В методе main реализовать цикл генерации 3 массивов случайных чисел длины N и для каждого сгенерированного массива вызывать метод сортировки в новом потоке. Установить для потоков приоритеты в соответствии с вариантом и после этого запустит потоки в произвольном порядке. Запустить программу несколько раз и пронаблюдать за соблюдением приоритетов.

В методе сортировки заменить вывод всего массива на поэлементный вывод с помощью вызова System.out.print(...), причем массив должен выводиться по 100 элементов в строке и с двух сторон отделяться строками, состоящими из символов «*». Перед строками содержимого массива выводить имя текущего потока. Проверить корректность вывода при запуске этой программы.

Реализовать синхронизацию потоков в момент записи значений в поток вывод. Запустить исправленную программу и проверить корректность вывода.

6.3. Ход выполнения работы

6.3.1. Текст программы

Программа составлена на языке Java.

Листинг 1 — Содержимое файла AbstractApplication.java

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Random;
import java.util.concurrent.CompletableFuture;
import java.util.stream.Collectors;
public class AbstractApplication
  protected static final int ARRAY_SIZE = 10000;
  protected static final int THREAD_COUNT = 3;
  protected static final int PRINT_STEP = 100;
  protected static final int[] THREAD_PRIORITIES = { Thread.MAX_PRIORITY, Thread.NORM_PRIORITY,
Thread.MIN_PRIORITY };
  private static final Random gen = new Random();
  public static void main (String... args) throws InterruptedException
    System.out.println("Создание");
    Integer[][] arrays = generateArrays(THREAD_COUNT, ARRAY_SIZE);
    for (int i = 0; i < THREAD_COUNT; ++i)</pre>
    {
      var thread = new Thread(new MyTask(arrays[i]));
      thread.setName("Sorter_" + (i + 1));
      thread.setPriority(THREAD_PRIORITIES[i % THREAD_PRIORITIES.length]);
      thread.start();
    }
  }
//
    public static void printArray (Class<?> who, Integer[] array)
      Logger.toLog(who, Arrays.stream(array).map(x -> x.toString()).collect(Collectors.joining("; ", "{",
"}")));
// public static void printArray (Class<?> who, Integer[] array)
//
//
      for (i = 0; i + PRINT_STEP < array.length; i += PRINT_STEP + 1)</pre>
//
//
//
//
        Logger
          .toLog(
//
//
//
//
//
//
//
//
            who,
            Arravs
              .stream(Arrays.copyOfRange(array, i, i + PRINT_STEP))
              .map(x -> x.toString())
              .collect(Collectors.joining("; ", "* ", " *")));
      }
      Logger
        .toLog(
            .stream(Arrays.copyOfRange(array, i, array.length))
            .map(x -> x.toString())
            .collect(Collectors.joining("; ", "* ", " *")));
   }
  public synchronized static void printArray (Class<?> who, Integer[] array)
```

```
{
 for (i = 0; i + PRINT_STEP < array.length; i += PRINT_STEP + 1)</pre>
   Logger
      .toLog(
        who,
        Arrays
          .stream(Arrays.copyOfRange(array, i, i + PRINT_STEP))
          .map(x -> x.toString())
          .collect(Collectors.joining("; ", "* ", " *")));
 }
 Logger
    .toLog(
      who,
      Arrays
        .stream(Arrays.copyOfRange(array, i, array.length))
        .map(x -> x.toString())
        .collect(Collectors.joining("; ", "* ", " *")));
}
private static Integer[] generateArray (Integer size)
 var ret = new Integer[size];
 for (int i = 0; i < ret.length; ++i)</pre>
   ret[i] = gen.nextInt();
 return ret;
private static Integer[][] generateArrays (int count, int size)
 var ret = new Integer[count][];
 List<CompletableFuture<Integer[]>> futures = new ArrayList<>(count - 1);
 for (int i = 0; i < ret.length - 1; ++i)
   futures.add(CompletableFuture.supplyAsync( () -> generateArray(size)));
 ret[0] = generateArray(size);
 for (int i = 0; i < futures.size(); ++i)</pre>
   ret[i + 1] = futures.get(i).join();
 return ret;
public static Integer[] sort (Integer[] arr)
 for (int left = 0; left < arr.length; left++)</pre>
 {
   int value = arr[left];
    int i = left - 1;
    for (; i >= 0; i--)
    {
      if (value < arr[i])</pre>
      {
        arr[i + 1] = arr[i];
      else
      {
        break;
      }
   arr[i + 1] = value;
 return arr;
```

class MyTask implements Runnable

```
{
  private final Integer[] array;
  public MyTask (Integer[] array)
  {
    this.array = array;
  }
  @Override
  public void run ()
  {
    AbstractApplication.printArray(this.getClass(), AbstractApplication.sort(array));
  }
}

class Logger
  {
  public static void toLog (Class<?> who, String message)
    {
       System.out.println(String.format("[%s/%s] %s", Thread.currentThread().getName(), who.getName(), message));
  }
}
```

6.3.2. Результаты тестирования

Программа была скомпилирована и запущена. Рисунок 1 демонстрирует вывод всех элементов каждого массива на экран.

Рисунок 2 демонстрирует вывод всех элементов каждого массива по 10 элементов в строчке на экран. После анализа сделан вывод, что выводимые данные перемешиваются вследствие отсутствия блокировок. Рисунок 3 демонстрирует вывод программы с исправленной функцией вывода данных.

COLUMN
CO

Рисунок 1 — Вывод всех элементов массива сразу

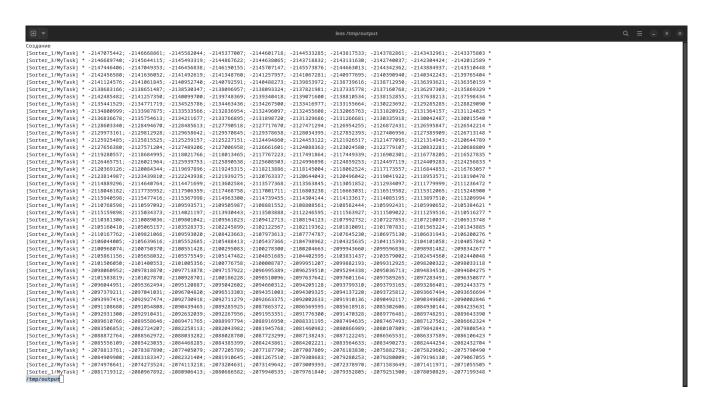


Рисунок 2 — Вывод элементов массива по 10 на строчку

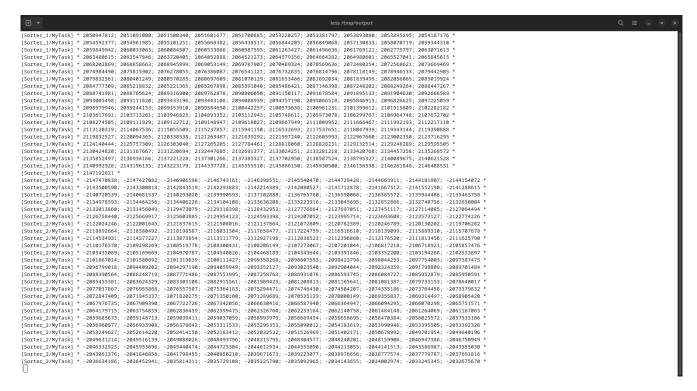


Рисунок 3 — Исправленный вывод элементов массива по 10 на строчку

Вывод

При выполнении данной работы были получанны навыки созданий многопоточных приложений с помощью средств языка Java. В процессе работы были использованы следующие средства: конструкторы потоков, создание блоков-блокировщиков с помощью ключевого слова synchronized.