Задание 9.
Работа с потоками исполнения

9.1. Напишите консольное приложение, которое:
- описывает простой поток для отсчета и вывода в консоль чисел от 10 до нуля с задержкой в 1 секунду, в конце выводит строку "Bomb";
- создает и запускает описанный поток на исполнение.

ТРЕБОВАНИЯ.
1. Приложение должно быть написано на языке Java.
2. Использовать только стандартные компиляторы и библиотеки.
3. При кодировании должны быть использованы соглашения об оформлении кода для языка Java.
4. Потоки опишите через реализацию интерфейса **Runnable** с использованием лямбда-выражения.

9.2. Напишите консольное приложение, которое:

- описывает поток **Counter**, задача которого инкрементировать значение некоторой переменной от нуля до 1_000_000;

- описывает поток **Printer**, задача которого выводить в консоль значение инкрементируемой переменной;

- создает и запускает описанные потоки на исполнение.

ТРЕБОВАНИЯ.
1. Приложение должно быть написано на языке Java.
2. Использовать только стандартные компиляторы и библиотеки.
3. При кодировании должны быть использованы соглашения об оформлении кода для языка Java.
4. Потоки опишите через реализацию интерфейса **Runnable** с использованием лямбда-выражения.
5. Не используйте синхронизацию и объясните результат работы.
6. Примените механизм wait/notify так, чтобы вывод значения числа был следующим: 0 1 2 3 4 5 6 … 999999.

9.3. Напишите консольное приложение, которое:

- создает и запускает несколько потоков, которые производят запись в коллекцию типа **Map**;

- создает и запускает несколько потоков, которые выполняют чтение данных из той же коллекции типа **Map**;

- опишите коллекцию в двух вариантах:
  - типа **ConcurrentHashMap**;
  - типа **HashMap**;
- оцените время работы потоков с коллекциями.

ТРЕБОВАНИЯ.

1. Приложение должно быть написано на языке Java.
2. Использовать только стандартные компиляторы и библиотеки.
3. При кодировании должны быть использованы соглашения об оформлении кода для языка Java.
4. Потоки опишите через реализацию интерфейса **Runnable** с использованием лямбда-выражения.
5. Коллекции **HashMap** и **ConcurrentHashMap** должны иметь одинаковую начальную емкость и коэффициент загруженности.
6. Количество читающих и записывающих потоков должно быть одинаковым для этих коллекций.
7. Записываемые и читаемые данные должны быть одинаковыми для этих коллекций.

9.4. Напишите консольное приложение для вычисления суммы всех элементов массива (из 1_000_000 целых элементов, значения которых генерируются случайным образом в диапазоне от 0 до 100), используя фреймворк ForkJoin. Для этого:

- создайте и инициализируйте массив размерностью 1_000_000 элементов;

- опишите рекурсивную задачу для деления массива на две части и назначения каждой части другой рекурсивной задаче для дальнейшего деления;

- продолжайте деление массива, пока в подмассиве не останется менее 20 элементов, и тогда вычислите сумму этих элементов.

ТРЕБОВАНИЯ.

1. Приложение должно быть написано на языке Java.
2. Использовать только стандартные компиляторы и библиотеки.
3. При кодировании должны быть использованы соглашения об оформлении кода для языка Java.
4. Задачу реализуйте как **RecursiveTask**.
5. Количество рабочих потоков в пуле установите на 8.

9.5. Напишите консольное приложение Java, которое рекурсивно обрабатывает указанную директорию, запуская для каждой поддиректории отдельный поток, с использованием высокоуровневого интерфейса для доступа к ресурсам:

- ✓ найти в заданной директории текстовые файлы, и подсчитать в каждом файле количество слов, начинающихся с заданной буквы. В некоторый файл записать названия таких файлов и количество найденных слов. Вывести в консоль содержимое созданного файла.

ТРЕБОВАНИЯ.

1. Приложение должно быть написано на языке Java.
2. Использовать только стандартные компиляторы и библиотеки.
3. При кодировании должны быть использованы соглашения об оформлении кода для языка Java.
4. Обеспечить обработку произвольной директории в файловой системе компьютера, для чего ее местоположение и название ввести набором с клавиатуры.
5. При необходимости название новых директории и/или файла для записи результата поиска тоже ввести набором на клавиатуре.
6. Для обеспечения устойчивости работы предусмотреть проверку корректности ввода данных и предложить решение в случае неверного их введения для продолжения работы приложения.
7. Все действия должны фиксироваться выводом соответствующих сообщений.