Упражнение 1 (Синхронизация потоков).

- Откройте проект SyncMain в каталоге Лабы Java2/Threads/SyncMain.

- Запустите его, обратите внимание, что в программе создаётся два дополнительных потока. Каждый из них увеличивает счётчик 10 раз, а окончательный результат равен 10, вместо 20.

- Добавьте необходимый код для синхронизации счётчика «count» из класса «Data». Весь код синхронизации должен быть в методе «Add».

- Добавьте необходимый код для синхронизации статического счётчика «countSt» из класса «Data». Весь код синхронизации должен быть в методе «AddStatic».

- Запустите программу, оба счётчика равны 20? Почему?

Упражнение 2 (Последовательная работа потоков).

- Откройте проект TicTak в каталоге Лабы Java2/Threads /TicTak.

- Запустите его и обратите внимание, что первым всегда начинает работать первый поток. Затем порядок срабатывания потоков произвольный. Необходимо добавить код, который заставит потоки работать последовательно:

* откройте метод «main» и первым оператором добавьте создание объекта класса «Object»и с именем «monitor»;
* ссылку на этот объект передайте и в первый и во второй потоки;
* откройте класс «ThreadOne»;
* добавьте приватную переменную «monitor» с типом «Object»;
* измените конструктор, добавив ему входной параметр с типом «Object» и инициализируйте переменную «monitor» входным параметром;
* в цикле метода «run», поле печати добавьте следующий код

synchronized (monitor) {

monitor.notify();

if (i < TicTak.num-1)

monitor.wait();

}

* снимите комментарий с блока try-catch;
* откройте класс «ThreadTwo»и сделайте в нём аналогичные изменения;

- Запустите программу. Зачем понадобилось создавать объект «monitor»?

Упражнение 3 (Реализация паттерна читатели - писатель).

- Откройте проект RW в каталоге Лабы Java2/Threads /RW.

- Запустите его и обратите внимание на скорость выполнения и на корректность данных.

- Поставьте перед методами «read» и «write» модификатор «synchronized». Как изменилась скорость работы? Что с данными?

- Уберите «synchronized».

- В класс «Data» добавьте следующие члены класса

private ReadWriteLock lock = new ReentrantReadWriteLock();

private Lock rl = lock.readLock();

private Lock wl = lock.writeLock();

- Используйте объект «rlock» для получения блокировки на чтение в методе «read». Добавьте блок «finally», в котором и снимите блокировку на чтение.

- Используйте объект «wlock» для получения блокировки на запись в методе «write». Добавьте блок «finally», в котором и снимите блокировку на запись.

- Запустите программу. Оцените скорость работы и корректность данных.