Exercise 6 (10 points) – can be done individually or in pair

- The first lines of all source files must be comment containing names & IDs of all members. Also create file readme.txt containing names & IDs of all members.
- Put all files (source, input, output) in folder Ex6_xxx where xxx = your full ID. That is, your source files must be in package Ex6_xxx and input/output files (if there is any) must be read from/write to this folder. From now on, you'll get point deduction for wrong package & folder structure.
- The group representative zips Ex6_xxx & submits it to Google Classroom. The other members submit only readme.txt. Email submission is not accepted.
- The exercise is graded only once, and after graded, members can't be added.

==========

1. Complete class **FactorThread**. Modify it as needed. You can add more variables & methods, but do not change the visibility of existing ones

```
class FactorThread extends Thread {
  private PrintWriter        out;
  private ArrayList<Integer> allPrimes;
  private int                target;

  public void run() {
    // Create PrintWriter object to write result to a separate file

    // Execute steps 1-3 in loop until #primes = target:
    //   1. Random a value v in range [100, 500] and find all its positive factors (i.e.
    //      integers that divide it with 0 remainder.
    //
    //   2. If v is a prime:     print round number, v, current #primes to file as in demo.
    //   3. If v is not a prime: print round number, v, its factors to file as in demo.

    // After the loop, print thread name, #rounds, all primes (sorted in increasing order)
    // to the screen.
  }
}
```

** The output file must be placed in the same folder as your source file

2. Write another class that acts as the main class. In its main method
   2.1 Ask user for target number of primes.
   2.2 Ask user for number of threads.
   2.3 Create FactorThreads to perform the task in (1).

```
--- exec:3.1.0:exec (default-cli) @ solutions ---
Target #primes =
8
Number of threads =
3                                    Create the whole String & call System.out only once
Thread T2 finishes in  32 rounds, primes = [181, 211, 223, 229, 271, 311, 347, 499]
Thread T1 finishes in  54 rounds, primes = [211, 223, 269, 347, 379, 383, 463, 487]
Thread T0 finishes in  51 rounds, primes = [137, 151, 193, 223, 223, 239, 263, 313]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

- In different runs, the finishing order between threads should be different. If it is always T0, T1, T2, …, then you may not do multithreaded program properly.
- Threads compete for System.out. If #rounds are close, the one who finishes first may get System.out later. And to prevent mixed-up outputs from >1 threads, make each thread call System.out only once.

T0.txt

Random value [100, 500]

```
Round   1 >> 341           factors = [1, 11, 31, 341]
Round   2 >> 223 = prime   #primes = 1
Round   3 >> 484           factors = [1, 2, 4, 11, 22, 44, 121, 242, 484]
Round   4 >> 223 = prime   #primes = 2
Round   5 >> 144           factors = [1, 2, 3, 4, 6, 8, 9, 12, 16, 18, 24, 36, 48, 72, 144]
...
Round  50 >> 475           factors = [1, 5, 19, 25, 95, 475]
Round  51 >> 313 = prime   #primes = 8
```

T2.txt

```
Round   1 >> 217           factors = [1, 7, 31, 217]
Round   2 >> 309           factors = [1, 3, 103, 309]
Round   3 >> 211 = prime   #primes = 1
Round   4 >> 387           factors = [1, 3, 9, 43, 129, 387]
Round   5 >> 295           factors = [1, 5, 59, 295]
...
Round  31 >> 369           factors = [1, 3, 9, 41, 123, 369]
Round  32 >> 347 = prime   #primes = 8
```