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

- The first lines of all source files must be comments containing names & IDs of all members. Also create file readme.txt containing names & IDs of all members

- Put all files (source, input, readme.txt) in folder Ex6_xxx where xxx = ID of the group representative, i.e. your source files must be in package Ex6_xxx (assumedly in Maven's src/main/java). Input files must be read from this path

- The group representative zips Ex6_xxx & submits it to Google Classroom. The other members submit only readme.txt. Email submission is not accepted

==========

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

```
class ShuffleSortThread extends Thread {
   private PrintWriter        out;          // each thread writes to a separate file
   private ArrayList<Integer> myNumbers;    // numbers to be sorted
   private boolean            increasing;   // sorting direction: increasing ?

   public ShuffleSortThread(String n, ArrayList<Integer> org, boolean in) {
      super(n);
      myNumbers  = new ArrayList<Integer>(org);
      increasing = in;
   }

   public void run() {
      // 1. Print initial order and sorting direction to file
      // 2. Keep shuffling the numbers until they are sorted
      // 3. Print round number & the new order to file
      // 4. Once the numbers are sorted, report #rounds to screen
   }
}
```

Note : To shuffle ArrayList, use Collections.shuffle(myNumbers)

2. Write another class that acts as the main class. In its main method

   2.1 Ask user for #values (e.g. n) to be sorted
   Create an ArrayList containing 1, 2, …, n. Shuffle it to get an initial order

   2.2 Ask user for #threads
   Create ShuffleSortThreads, sending the ArrayList in (2.1) to all threads. Alternate sorting directions (increasing & decreasing) between threads

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ solutions ---
Number of values to sort =
5
Initial order = 5  2  3  1  4

Number of threads =
4
Thread_0 gets increasing order in 36 rounds
Thread_2 gets increasing order in 31 rounds
Thread_3 gets decreasing order in 60 rounds
Thread_1 gets decreasing order in 74 rounds
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

In different runs, the finishing order between threads should be different. If it is always Thread_0, Thread_1, Thread_2, …, then you may not do multithreaded program properly

Threads also have to compete for System.out. If #rounds are close, the one who finishes first may get System.out later

Thread_0.txt

```
round   0 : numbers = 5  2  3  1  4
=== Shuffle to increasing order ===
Round   1 : numbers = 4  3  2  1  5
Round   2 : numbers = 2  1  4  5  3
Round   3 : numbers = 2  5  3  1  4
...
Round  35 : numbers = 5  1  3  2  4
Round  36 : numbers = 1  2  3  4  5
```

Thread_1.txt

```
round   0 : numbers = 5  2  3  1  4
=== Shuffle to decreasing order ===
Round   1 : numbers = 5  4  3  1  2
Round   2 : numbers = 3  2  5  1  4
Round   3 : numbers = 4  5  2  1  3
...
Round  73 : numbers = 1  3  2  5  4
Round  74 : numbers = 5  4  3  2  1
```

Thread_2.txt

```
round   0 : numbers = 5  2  3  1  4
=== Shuffle to increasing order ===
Round   1 : numbers = 3  5  2  1  4
Round   2 : numbers = 3  2  5  4  1
Round   3 : numbers = 1  4  3  2  5
...
Round  30 : numbers = 4  5  3  2  1
Round  31 : numbers = 1  2  3  4  5
```

Thread_3.txt

```
round   0 : numbers = 5  2  3  1  4
=== Shuffle to decreasing order ===
Round   1 : numbers = 4  5  3  1  2
Round   2 : numbers = 3  4  5  2  1
Round   3 : numbers = 2  5  3  4  1
...
Round  59 : numbers = 1  5  3  4  2
Round  60 : numbers = 5  4  3  2  1
```