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. That is, your source files must be in package Ex5_xxx and 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

===

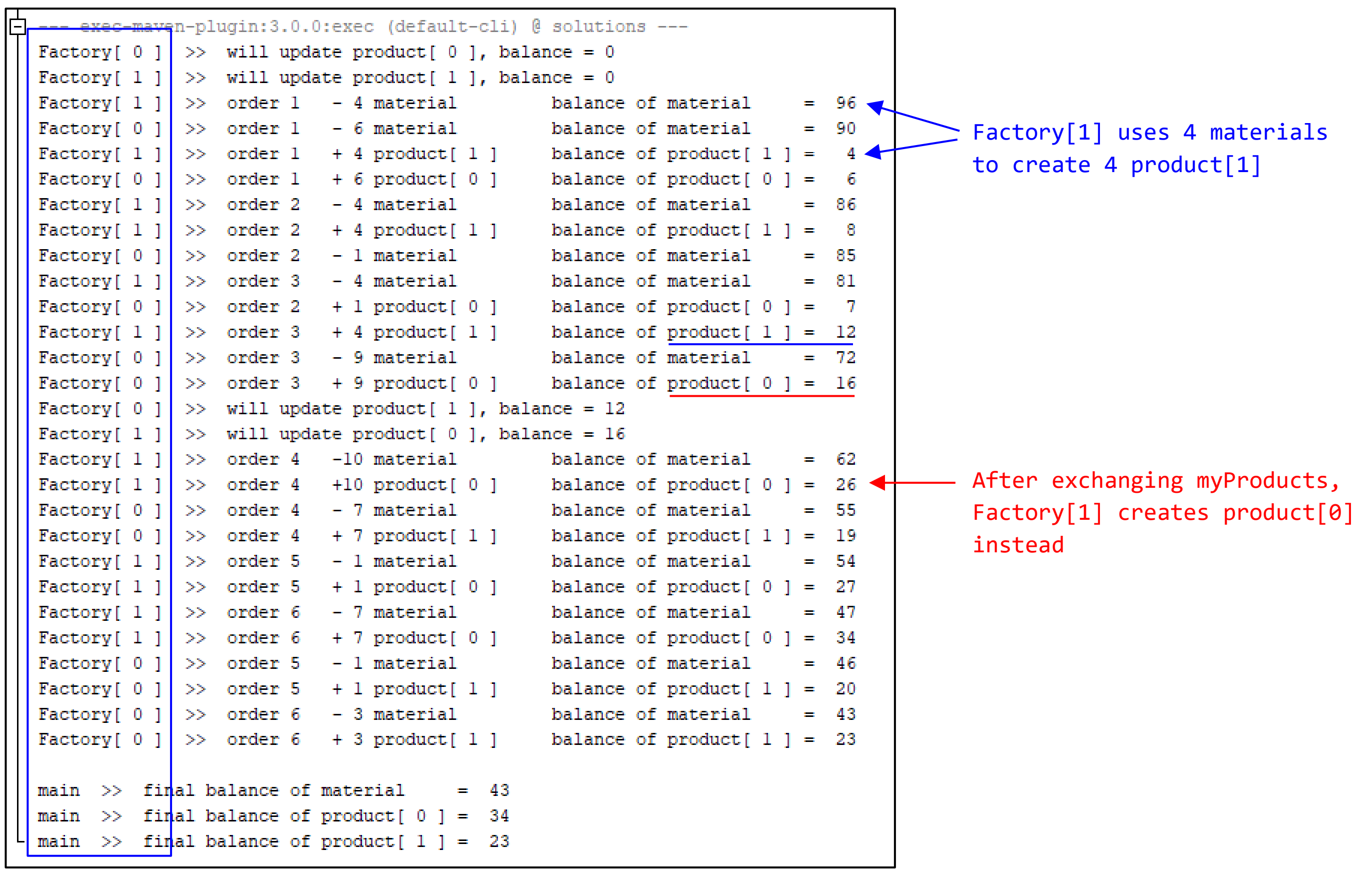
Complete the given source file to make the program work as follows:

1. Complete class FactoryThread. You can add more variables & methods, change method headers, but don't change the visibility of existing ones
    - Use Exchanger to exchange myProduct between FactoryThreads
    - Use CyclicBarrier to make both FactoryThreads start order processing at the same time

2. Complete classes Item, Material, and Product. You can add more variables & methods, change method headers, but don't change the visibility of existing ones and class Item must remain abstract
    - Use Semaphore or monitor to let only 1 thread update balance and print to System.out at a time. To get correct result, balance & System.out should be protected together

3. Complete method runSimulation for main thread's activies
    - Use Join to make main thread wait until all FactoryThreads complete their works before printing final balances

4. Each output line must be labelled by the name of the thread who prints it. Don't hard code thread name, but use Thread.currentThread() to get the printing thread



```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ solutions ---
Factory[ 0 ]  >>  will update product[ 0 ], balance = 0
Factory[ 1 ]  >>  will update product[ 1 ], balance = 0
Factory[ 1 ]  >>  order 1   - 4 material           balance of material     =  96
Factory[ 0 ]  >>  order 1   - 6 material           balance of material     =  90
Factory[ 1 ]  >>  order 1   + 4 product[ 1 ]       balance of product[ 1 ] =   4
Factory[ 0 ]  >>  order 1   + 6 product[ 0 ]       balance of product[ 0 ] =   6
Factory[ 1 ]  >>  order 2   - 4 material           balance of material     =  86
Factory[ 1 ]  >>  order 2   + 4 product[ 1 ]       balance of product[ 1 ] =   8
Factory[ 0 ]  >>  order 2   - 1 material           balance of material     =  85
Factory[ 1 ]  >>  order 3   - 4 material           balance of material     =  81
Factory[ 0 ]  >>  order 2   + 1 product[ 0 ]       balance of product[ 0 ] =   7
Factory[ 1 ]  >>  order 3   + 4 product[ 1 ]       balance of product[ 1 ] =  12
Factory[ 0 ]  >>  order 3   - 9 material           balance of material     =  72
Factory[ 0 ]  >>  order 3   + 9 product[ 0 ]       balance of product[ 0 ] =  16
Factory[ 0 ]  >>  will update product[ 1 ], balance = 12
Factory[ 1 ]  >>  will update product[ 0 ], balance = 16
Factory[ 1 ]  >>  order 4   -10 material           balance of material     =  62
Factory[ 1 ]  >>  order 4   +10 product[ 0 ]       balance of product[ 0 ] =  26
Factory[ 0 ]  >>  order 4   - 7 material           balance of material     =  55
Factory[ 0 ]  >>  order 4   + 7 product[ 1 ]       balance of product[ 1 ] =  19
Factory[ 1 ]  >>  order 5   - 1 material           balance of material     =  54
Factory[ 1 ]  >>  order 5   + 1 product[ 0 ]       balance of product[ 0 ] =  27
Factory[ 1 ]  >>  order 6   - 7 material           balance of material     =  47
Factory[ 1 ]  >>  order 6   + 7 product[ 0 ]       balance of product[ 0 ] =  34
Factory[ 0 ]  >>  order 5   - 1 material           balance of material     =  46
Factory[ 0 ]  >>  order 5   + 1 product[ 1 ]       balance of product[ 1 ] =  20
Factory[ 0 ]  >>  order 6   - 3 material           balance of material     =  43
Factory[ 0 ]  >>  order 6   + 3 product[ 1 ]       balance of product[ 1 ] =  23

main  >>  final balance of material      =  43
main  >>  final balance of product[ 0 ] =  34
main  >>  final balance of product[ 1 ] =  23
```

Factory[1] uses 4 materials to create 4 product[1]

After exchanging myProducts, Factory[1] creates product[0] instead

Thread who prints each line (by Thread.currentThread())