EGCO 213
Group Project 2 – Factory Simulation

The project can be done in a group of <= 5 students. Each group must do the project by themselves:

- **Everyone involved in cheating, either as a source or copier, will get ZERO point.**
- If late submitting group copies code from a graded group, the graded group will still be penalized.
- **In case of AI usage, you must write your own prompts and perform your own acquisition with the AI.** Due to the dynamic nature of generative AI, it is very unlikely that any 2 groups will get identical generated code even when using identical prompts.
    - Therefore, submitting identical code will be counted as cheating.
    - Don't use generated content obtained by other groups as your own.
    - Don't share generated content you get from the AI with other groups.
    - If any suspicious arises, I may ask both groups to show their chat history with the AI. Failure to do so will result in cheating penalty.
- And remember, you must be able to orally explain every line of the code you submit, regardless of its origin.

1. This project uses only 1 input file (config.txt).
First column of each line = descriptions of the following columns.

```
days,                   5
warehouse_num,          3
freight_num_max,        2, 100
supplier_num_min_max,   3, 50, 100
factory_num_max,        3, 80
```

   1.1 Line "days" ➔ next cols = #days of simulation.
   1.2 Line "warehouse_num" ➔ next cols = #warehouses of materials.
   1.3 Line "freight_num_max" ➔ next cols = #freights of shipping products, max capacity of each freight.
   1.4 Line "supplier_num_min_max" ➔ next cols = #SupplierThreads, min daily material supply of each thread, max daily material supply of each thread.
   1.5 Line "factory_num_max" ➔ next cols = #FactoryThreads, max daily production of each thread.

** Don't hard code these values. I may change some of them to check whether your calculation is correct.

- There are always 5 lines with columns as stated above, but numbers may be changed.
- There won't be any input error (e.g. invalid input, negative number, wrong format, missing columns) in this file. But the program must still handle the case of missing file. Don't let it crash.

2. Implement **class Warehouse** that represents an individual warehouse to keep materials, and **class Freight** that represents an individual freight for shipping products.

- Class Warehouse should have methods put and get for material handling (see 3 and 4).
- Class Freight should have methods ship and reset for product handling (see 4).
- All threads in the program must see the same list of warehouses and the same list of freights.

3. Implement **class SupplierThread** that represents an individual supplier as thread. Thread activities are done in loop. Each iteration of a loop = 1 day. In each day:

   3.1 Wait until 1 thread (main, SupplierThread, or FactoryThread) prints day number, all warehouse balances (accumulated from previous days), all freight capacities (reset every day to max capacity).

   3.2 Put materials in 1 random warehouse. The number of materials is randomed between min and max daily supply. Print thread activities as in the demo.

4. Implement **class FactoryThread** that represents an individual factory as thread. Thread activities are done in loop. Each iteration of a loop = 1 day. In each day:

   4.1 Wait until 1 thread (main, SupplierThread, or FactoryThread) finishes activities in 3.1.

   4.2 Also wait until all SupplierThreads finish putting materials in warehouses.

   4.3 Get materials from 1 random warehouse to create products, supposing that 1 material is converted to 1 product. Try to get as many materials as possible without exceeding max daily production. For simplicity, each thread will contact only 1 warehouse in each day – if it can't get materials from this warehouse, it will not try any other. Print thread activities as in the demo & wait until all threads complete this step.

   4.4 Check total products to ship (= products created today + unshipped products from previous days). Print thread activities as in the demo & wait until all threads complete this step.

   4.5 Send products to 1 random freight. Try to ship as many products as possible without exceeding max freight capacity. For simplicity, each thread will contact only 1 freight in each day – if it can't ship products to this freight, it will not try any other. Print thread activities as in the demo & wait until all threads complete this step.

   4.6 Check unshipped products. Print thread activities as in the demo.

   4.7 After all days of simulation, calculate the percentage of products created by this thread that are successfully shipped.

5. Implement main class with main method.

   5.1 Read simulation parameters from config.txt.

   5.2 Create Warehouses, Freights, SupplierThreads, FactoryThreads. Start all threads. You are recommended to use ArrayLists to keep objects for flexibility.

   5.3 After all threads complete all days of simulation, let main thread report FactoryThreads' performance as in the demo. The report must be sorted in decreasing order of total created products, and by thread names (if total products are equal).

** Everything printed to the screen must be labelled by the name of the thread who prints it. Don’t hard code thread’s name but use Thread.currentThread().getName().

6. Package and folder structure must be correct

   6.1 Your source files (.java) must be in folder Project2_XXX where XXX = full ID of the group representative, assuming that this folder is under Maven’s "src/main/java" structure. The first lines of all source files must be comments containing names & IDs of all members.

   6.2 Input files must be read from Project2_XXX. Don’t use absolute path that is valid only on your PC.

   6.3 Add readme.txt containing names & IDs of all members in Project2_XXX.

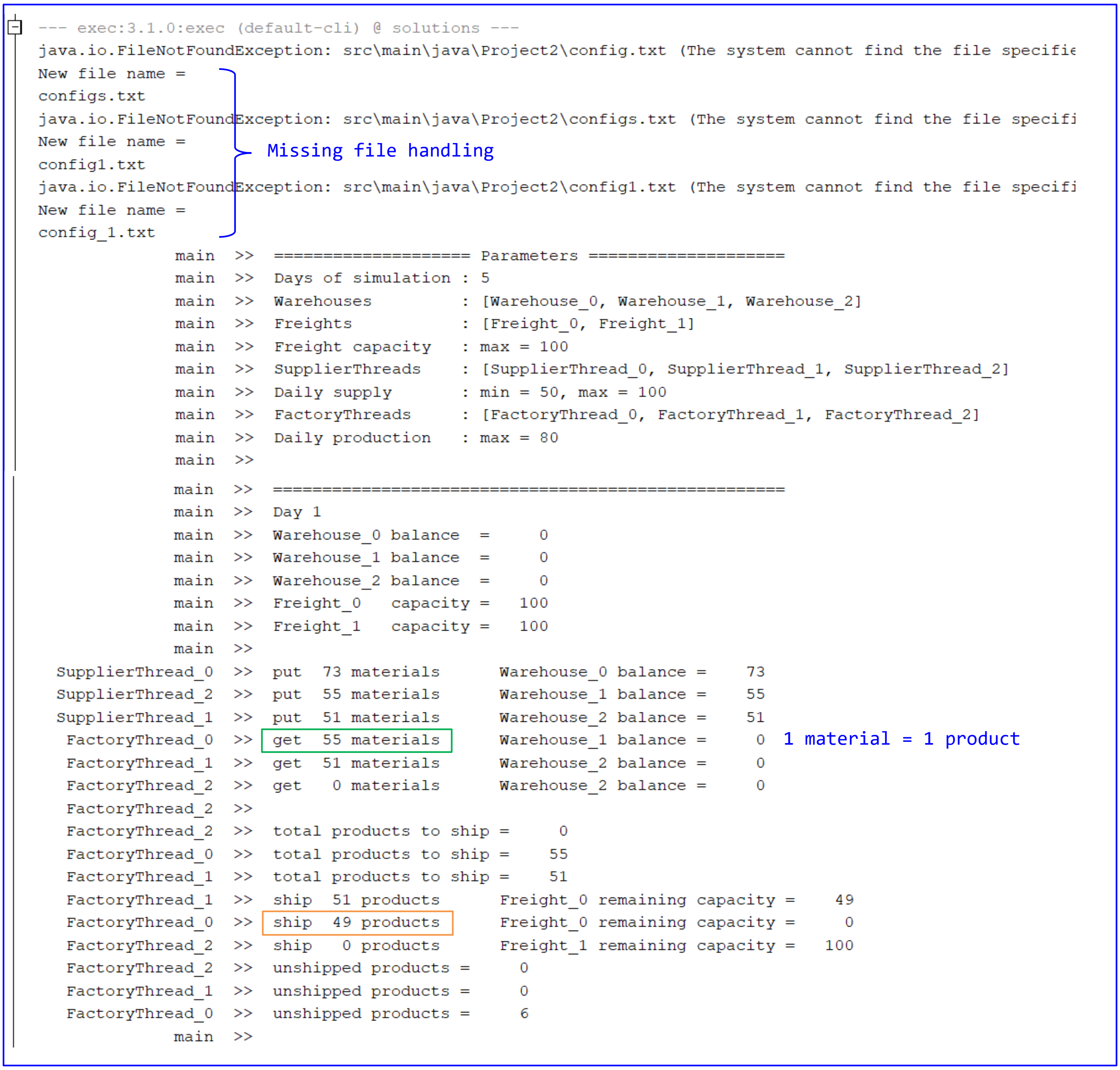
Submission

1. Group representative zips and submits Project2_XXX to Google classroom
2. Other members submit only readme.txt to Google classroom

Grading

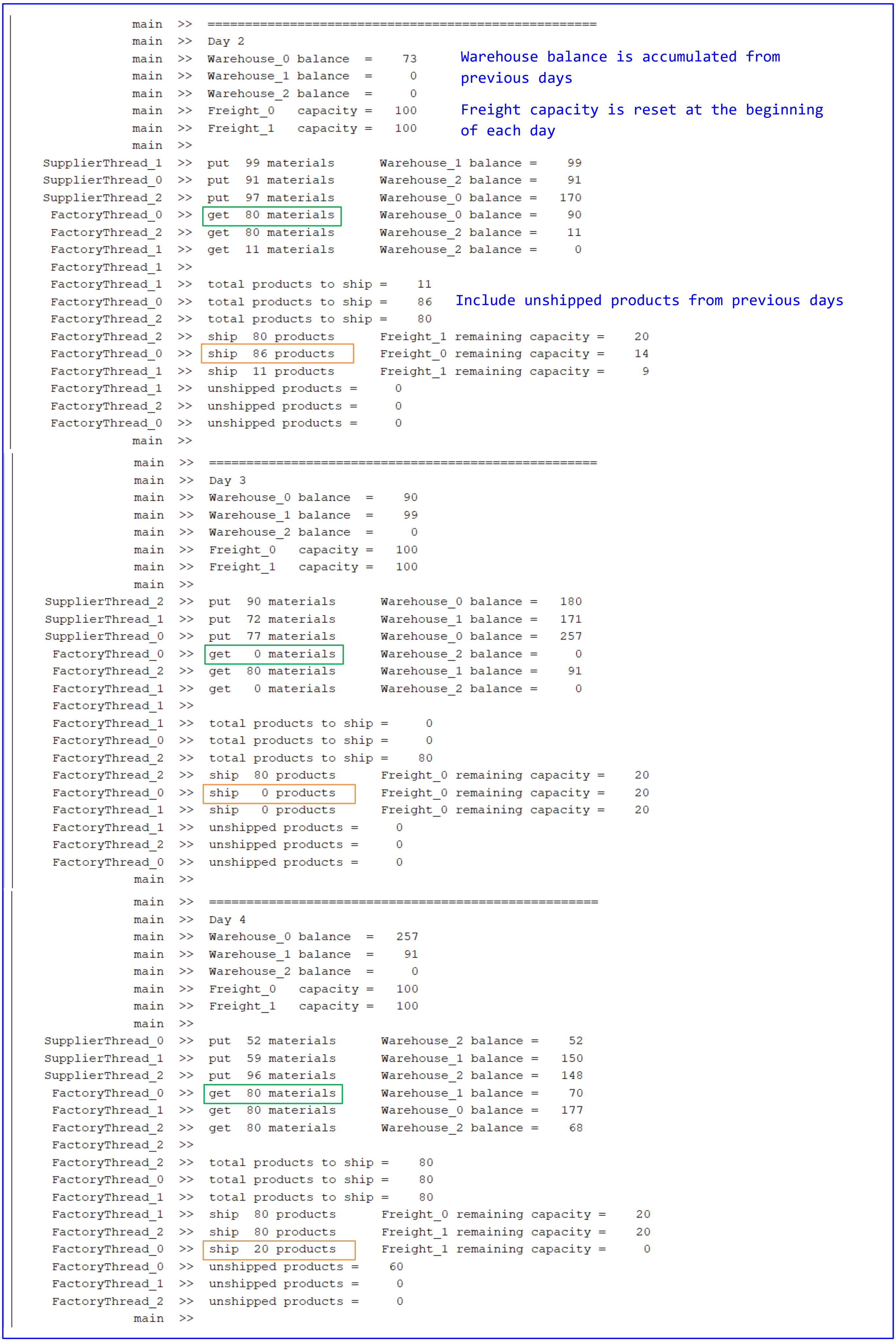
1	point	daily processing by main + SupplierThread	(requirements 3.1, 3.2)
2.5	points	daily processing by FactoryThread	(requirements 4.1-4.2, 4.3, 4.4, 4.5, 4.6)
1.5	points	summary by main thread	(requirements 4.7, 5.3)
1	point	others e.g. thread name, missing file handling	
4	points	design & programming in proper OOP and multithreading style Maximum & safe concurrency is expected. Don’t enforce sequential execution when threads can work in parallel. For example, if 2 SupplierThreads want to put materials in 2 different warehouses, they should be able to do it concurrently.	

Late submission: -0.5 points for <1 week late; -1 point for each 1 full week late



```
--- exec:3.1.0:exec (default-cli) @ solutions ---
java.io.FileNotFoundException: src\main\java\Project2\config.txt (The system cannot find the file specifie
New file name =
configs.txt
java.io.FileNotFoundException: src\main\java\Project2\configs.txt (The system cannot find the file specifi
New file name =
config1.txt
java.io.FileNotFoundException: src\main\java\Project2\config1.txt (The system cannot find the file specifi
New file name =
config_1.txt
              main  >>  ==================== Parameters ====================
              main  >>  Days of simulation : 5
              main  >>  Warehouses         : [Warehouse_0, Warehouse_1, Warehouse_2]
              main  >>  Freights           : [Freight_0, Freight_1]
              main  >>  Freight capacity   : max = 100
              main  >>  SupplierThreads    : [SupplierThread_0, SupplierThread_1, SupplierThread_2]
              main  >>  Daily supply       : min = 50, max = 100
              main  >>  FactoryThreads     : [FactoryThread_0, FactoryThread_1, FactoryThread_2]
              main  >>  Daily production   : max = 80
              main  >>

              main  >>  ====================================================
              main  >>  Day 1
              main  >>  Warehouse_0 balance  =     0
              main  >>  Warehouse_1 balance  =     0
              main  >>  Warehouse_2 balance  =     0
              main  >>  Freight_0   capacity =   100
              main  >>  Freight_1   capacity =   100
              main  >>
  SupplierThread_0  >>  put  73 materials     Warehouse_0 balance =    73
  SupplierThread_2  >>  put  55 materials     Warehouse_1 balance =    55
  SupplierThread_1  >>  put  51 materials     Warehouse_2 balance =    51
   FactoryThread_0  >>  get  55 materials     Warehouse_1 balance =     0
   FactoryThread_1  >>  get  51 materials     Warehouse_2 balance =     0
   FactoryThread_2  >>  get   0 materials     Warehouse_2 balance =     0
   FactoryThread_2  >>
   FactoryThread_2  >>  total products to ship =     0
   FactoryThread_0  >>  total products to ship =    55
   FactoryThread_1  >>  total products to ship =    51
   FactoryThread_1  >>  ship  51 products     Freight_0 remaining capacity =    49
   FactoryThread_0  >>  ship  49 products     Freight_0 remaining capacity =     0
   FactoryThread_2  >>  ship   0 products     Freight_1 remaining capacity =   100
   FactoryThread_2  >>  unshipped products =    0
   FactoryThread_1  >>  unshipped products =    0
   FactoryThread_0  >>  unshipped products =    6
              main  >>
```



```
            main  >>  ===================================================
            main  >>  Day 2
            main  >>  Warehouse_0 balance  =    73
            main  >>  Warehouse_1 balance  =     0
            main  >>  Warehouse_2 balance  =     0
            main  >>  Freight_0   capacity =   100
            main  >>  Freight_1   capacity =   100
```

Warehouse balance is accumulated from previous days

Freight capacity is reset at the beginning of each day

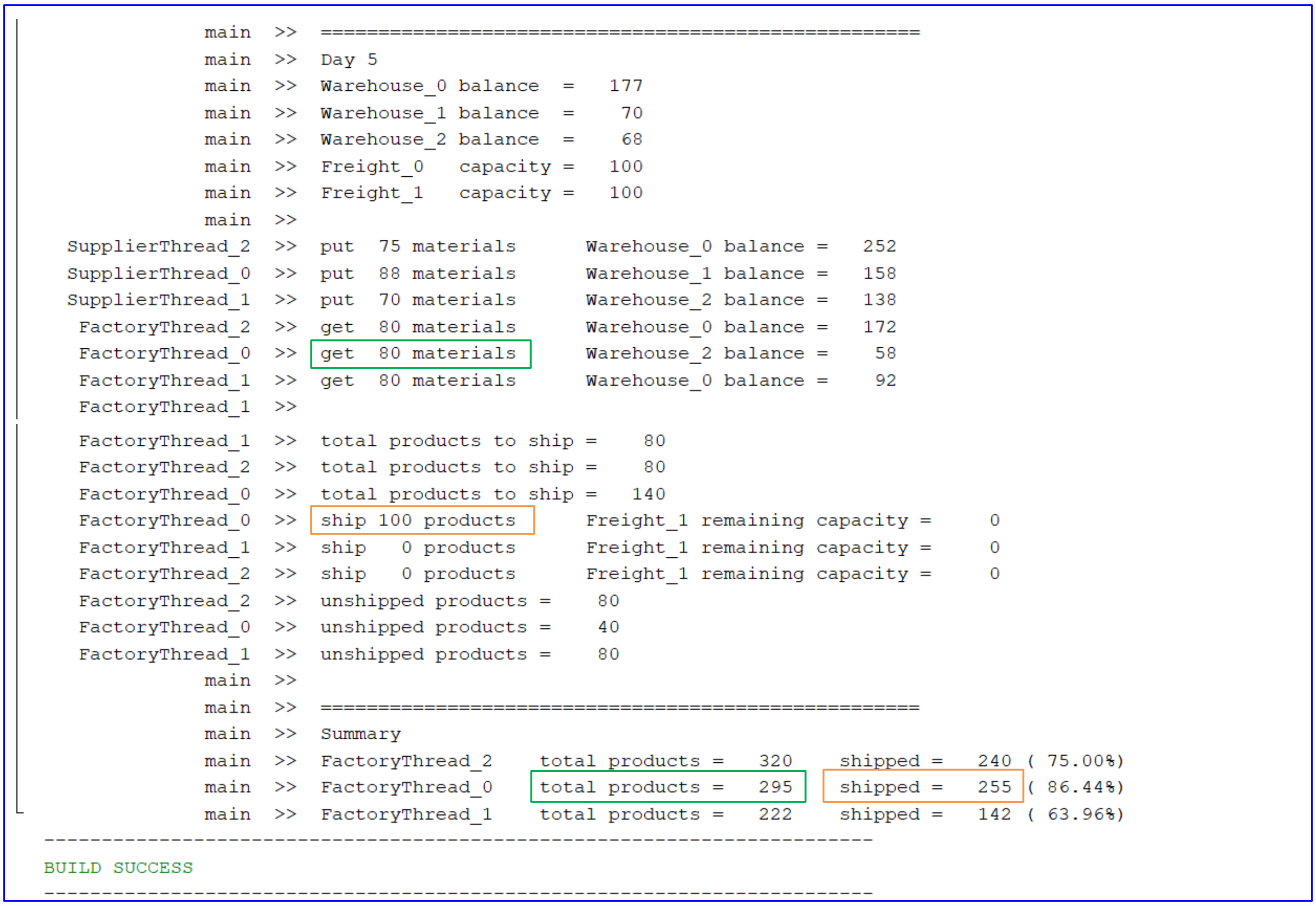
```
            main  >>
SupplierThread_1  >>  put  99 materials      Warehouse_1 balance =    99
SupplierThread_0  >>  put  91 materials      Warehouse_2 balance =    91
SupplierThread_2  >>  put  97 materials      Warehouse_0 balance =   170
 FactoryThread_0  >>  get  80 materials      Warehouse_0 balance =    90
 FactoryThread_2  >>  get  80 materials      Warehouse_2 balance =    11
 FactoryThread_1  >>  get  11 materials      Warehouse_2 balance =     0
 FactoryThread_1  >>
 FactoryThread_1  >>  total products to ship =    11
 FactoryThread_0  >>  total products to ship =    86
 FactoryThread_2  >>  total products to ship =    80
```

Include unshipped products from previous days

```
 FactoryThread_2  >>  ship  80 products      Freight_1 remaining capacity =    20
 FactoryThread_0  >>  ship  86 products      Freight_0 remaining capacity =    14
 FactoryThread_1  >>  ship  11 products      Freight_1 remaining capacity =     9
 FactoryThread_1  >>  unshipped products =     0
 FactoryThread_2  >>  unshipped products =     0
 FactoryThread_0  >>  unshipped products =     0
            main  >>

            main  >>  ===================================================
            main  >>  Day 3
            main  >>  Warehouse_0 balance  =    90
            main  >>  Warehouse_1 balance  =    99
            main  >>  Warehouse_2 balance  =     0
            main  >>  Freight_0   capacity =   100
            main  >>  Freight_1   capacity =   100
            main  >>
SupplierThread_2  >>  put  90 materials      Warehouse_0 balance =   180
SupplierThread_1  >>  put  72 materials      Warehouse_1 balance =   171
SupplierThread_0  >>  put  77 materials      Warehouse_0 balance =   257
 FactoryThread_0  >>  get   0 materials      Warehouse_2 balance =     0
 FactoryThread_2  >>  get  80 materials      Warehouse_1 balance =    91
 FactoryThread_1  >>  get   0 materials      Warehouse_2 balance =     0
 FactoryThread_1  >>
 FactoryThread_1  >>  total products to ship =     0
 FactoryThread_0  >>  total products to ship =     0
 FactoryThread_2  >>  total products to ship =    80
 FactoryThread_2  >>  ship  80 products      Freight_0 remaining capacity =    20
 FactoryThread_0  >>  ship   0 products      Freight_0 remaining capacity =    20
 FactoryThread_1  >>  ship   0 products      Freight_0 remaining capacity =    20
 FactoryThread_1  >>  unshipped products =     0
 FactoryThread_2  >>  unshipped products =     0
 FactoryThread_0  >>  unshipped products =     0
            main  >>

            main  >>  ===================================================
            main  >>  Day 4
            main  >>  Warehouse_0 balance  =   257
            main  >>  Warehouse_1 balance  =    91
            main  >>  Warehouse_2 balance  =     0
            main  >>  Freight_0   capacity =   100
            main  >>  Freight_1   capacity =   100
            main  >>
SupplierThread_0  >>  put  52 materials      Warehouse_2 balance =    52
SupplierThread_1  >>  put  59 materials      Warehouse_1 balance =   150
SupplierThread_2  >>  put  96 materials      Warehouse_2 balance =   148
 FactoryThread_0  >>  get  80 materials      Warehouse_1 balance =    70
 FactoryThread_1  >>  get  80 materials      Warehouse_0 balance =   177
 FactoryThread_2  >>  get  80 materials      Warehouse_2 balance =    68
 FactoryThread_2  >>
 FactoryThread_2  >>  total products to ship =    80
 FactoryThread_0  >>  total products to ship =    80
 FactoryThread_1  >>  total products to ship =    80
 FactoryThread_1  >>  ship  80 products      Freight_0 remaining capacity =    20
 FactoryThread_2  >>  ship  80 products      Freight_1 remaining capacity =    20
 FactoryThread_0  >>  ship  20 products      Freight_1 remaining capacity =     0
 FactoryThread_0  >>  unshipped products =   60
 FactoryThread_1  >>  unshipped products =     0
 FactoryThread_2  >>  unshipped products =     0
            main  >>
```



```
              main  >>  ====================================================
              main  >>  Day 5
              main  >>  Warehouse_0 balance  =   177
              main  >>  Warehouse_1 balance  =    70
              main  >>  Warehouse_2 balance  =    68
              main  >>  Freight_0   capacity =   100
              main  >>  Freight_1   capacity =   100
              main  >>
  SupplierThread_2  >>  put  75 materials      Warehouse_0 balance =   252
  SupplierThread_0  >>  put  88 materials      Warehouse_1 balance =   158
  SupplierThread_1  >>  put  70 materials      Warehouse_2 balance =   138
   FactoryThread_2  >>  get  80 materials      Warehouse_0 balance =   172
   FactoryThread_0  >>  get  80 materials      Warehouse_2 balance =    58
   FactoryThread_1  >>  get  80 materials      Warehouse_0 balance =    92
   FactoryThread_1  >>
   FactoryThread_1  >>  total products to ship =    80
   FactoryThread_2  >>  total products to ship =    80
   FactoryThread_0  >>  total products to ship =   140
   FactoryThread_0  >>  ship 100 products      Freight_1 remaining capacity =     0
   FactoryThread_1  >>  ship    0 products      Freight_1 remaining capacity =     0
   FactoryThread_2  >>  ship    0 products      Freight_1 remaining capacity =     0
   FactoryThread_2  >>  unshipped products =    80
   FactoryThread_0  >>  unshipped products =    40
   FactoryThread_1  >>  unshipped products =    80
              main  >>
              main  >>  ====================================================
              main  >>  Summary
              main  >>  FactoryThread_2    total products =   320     shipped =   240  ( 75.00%)
              main  >>  FactoryThread_0    total products =   295     shipped =   255  ( 86.44%)
              main  >>  FactoryThread_1    total products =   222     shipped =   142  ( 63.96%)
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```