EGCI 213
Group Project 2 – Stock Simulation

Implement a simulation program that manages the stocks of 4 products. The stock management is done by 3 VendorThreads as follows.

1. Create file **products.txt** containing names of 4 products.

```
bluetooth speaker
clip lens
mini projector
power bank
```

Implement **class Product**, with at least the following

- Variables name and balance : product names are read from products.txt
- Methods **addToStock** and **removeFromStock** : add and remove product from its stock
- Add more variables and methods as needed
- Your program will work with an array or ArrayList of 4 Product objects

** Don't hard code product names. I'll check your program by using your input file and mine

2. Create 3 input files for 3 VendorThreads. Each file contains 10 lines of transactions. Each transaction consists of transaction ID, product names, and amount of product. Positive amount indicates buying (adding product to stock), while negative amount indicates selling (removing product from stock)

```
 1, bluetooth speaker,  100
 2, clip lens,         -100
 3, mini projector,     100
 4, power bank,         100
 5, bluetooth speaker, -500
 6, clip lens,         -500
 7, mini projector,    -500
 8, power bank,         500
 9, bluetooth speaker,  600
10, clip lens,          600
```

Implement **class VendorThread** that extends class Thread or implements interface Runnable. It must have at least

- Variable allStocks : an array or ArrayList of 4 Products from (1). All threads must handle the same set of Products
- Add more variables and methods as needed
- The VendorThread will process all buying transactions (#1, #3, #4, #8, #9, #10) followed by all selling transactions (#2, #5, #6, #7)

** Don't hard code transactions. I'll check your program by using your input files and mine

3. Implement **class StockSimulation** as the main class. When the program starts
   3.1 Create an array or ArrayList of 4 Products. The initial balance of each product is 0.
   3.2 Create 3 VendorThreads to read input files and process transactions as follows:
   - Each thread processes buying transactions (adding products to stocks) first. From the example in (2), it will process transactions #1, #3, #4, #8, #9, #10. When processing each transaction, print at least the following to screen : transaction ID, product name, amount of buying, current balance of that Product
   - After all threads complete their buying transactions, let one thread (either VendorThread or main thread) report the summary of all Products. Products must be printed in increasing or decreasing order of their balances
   - Let all thread process selling transactions (removing products from stocks). From the example in (2), the thread will process transactions #2, #5, #6, #7. The amount of product to be sold must not exceed the balance. Again, print at least the following to screen : transaction ID, product name, amount of selling (may be smaller than input value, depending on its balance), current balance of that Product

- After all threads complete their selling transactions, let one thread (either VendorThread or main thread) report the summary of all Products. Products must be printed in increasing or decreasing order of their balances

3.3 Ask whether the user wants to run another stock simulation. Repeat (3.1)-(3.2) using the same input files. All balances must be reset to 0. Due to the non-deterministic nature of multithreaded program, each stock simulation may yield different thread orders. But the stock summaries remain the same.

3.4 Every line of output must be labeled with the name of the thread who prints it.
Do not hard code `System.out.printf("main > ...")`
But rather use `System.out.printf("%s > ...", Thread.currentThread().getName())`

4. In summary, your program must have
   - Files : products.txt (4 products) & 3 input files for 3 threads (10 transactions each), assuming no input error and all product names in transaction files being matched those in products.txt. But your program must still handle missing file(s)
   - Classes : Product, VendorThread, StockSimulation (main class)
   - You may add more classes as needed

The project can be done in a group of 2-4 students (a group of 1 student is not allowed). Each group must do the project by themselves. **Everyone involved in cheating will get ZERO point**

Marking

2 points	correct transaction processing (buying, selling)
2 points	correct stock summaries after buying and selling
1 point	looping for another stock simulation
1 point	other requirements (output, thread labels, missing file handling, etc.)
4 points	proper design & programming in OOP and multithreaded style

Submission : Saturday 29 June, 18.00

1. Your source file (.java) and input files
2. File Readme.txt containing names & IDs of everyone in your group
3. All files must be put in one folder. Name the folder after a member's name or ID
4. Zip the folder and submit it to rangsipan@gmail.com
   - In case of multiple submissions, the earliest version will be marked
   - Put EGCI 213 – project 2 in the email's subject

Example

products.txt

```
bluetooth speaker
clip lens
mini projector
power bank
```

v1.txt

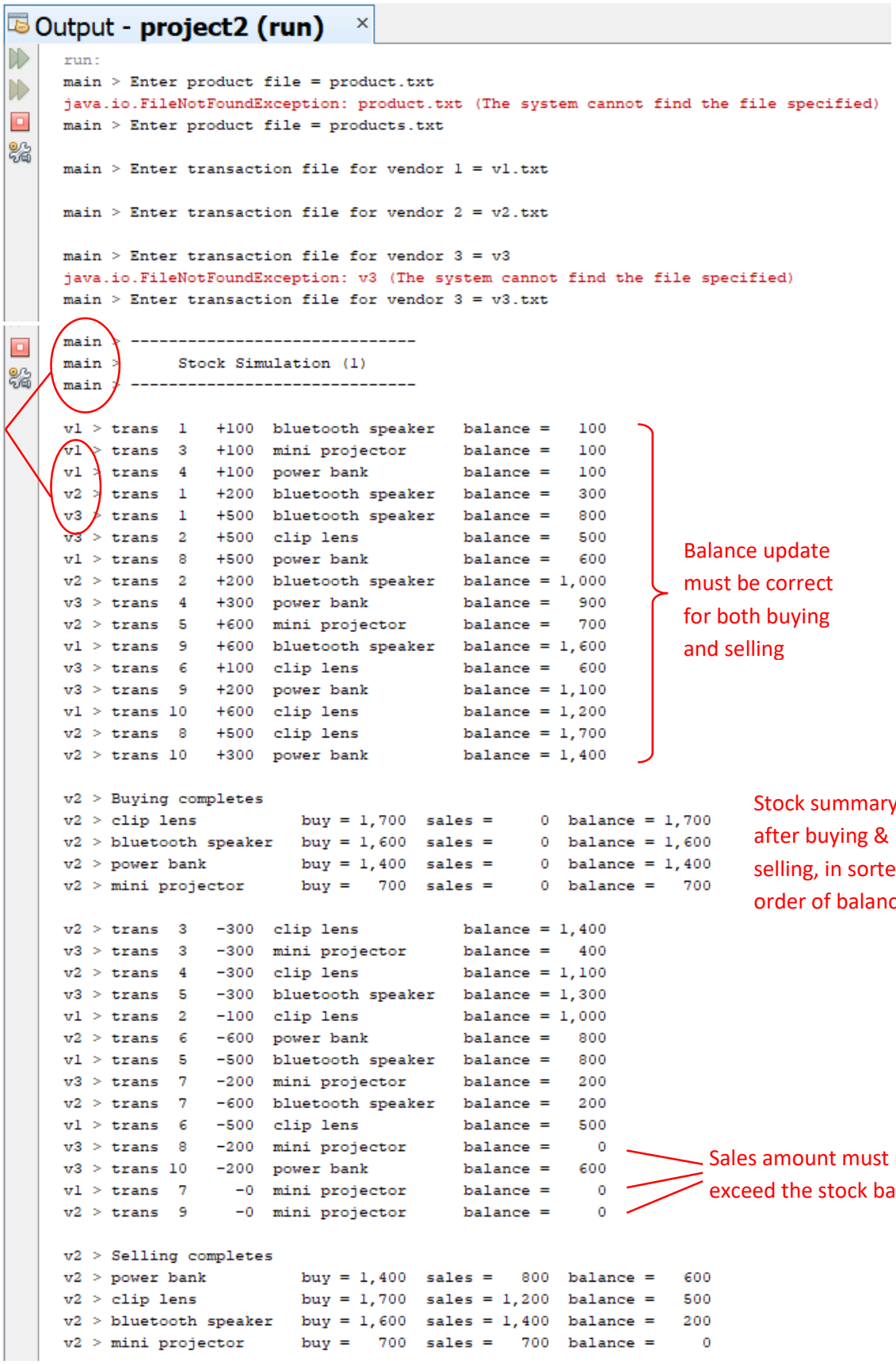
```
 1, bluetooth speaker,  100
 2, clip lens,         -100
 3, mini projector,     100
 4, power bank,         100
 5, bluetooth speaker, -500
 6, clip lens,         -500
 7, mini projector,    -500
 8, power bank,         500
 9, bluetooth speaker,  600
10, clip lens,          600
```

v2.txt

```
 1, bluetooth speaker,  200
 2, bluetooth speaker,  200
 3, clip lens,         -300
 4, clip lens,         -300
 5, mini projector,     600
 6, power bank,        -600
 7, bluetooth speaker, -600
 8, clip lens,          500
 9, mini projector,    -300
10, power bank,         300
```

v3.txt

```
 1, bluetooth speaker,  500
 2, clip lens,          500
 3, mini projector,    -300
 4, power bank,         300
 5, bluetooth speaker, -300
 6, clip lens,          100
 7, mini projector,    -200
 8, mini projector,    -400
 9, power bank,         200
10, power bank,        -200
```


Output - project2 (run)
run:
main > Enter product file = product.txt
java.io.FileNotFoundException: product.txt (The system cannot find the file specified)
main > Enter product file = products.txt
main > Enter transaction file for vendor 1 = v1.txt
main > Enter transaction file for vendor 2 = v2.txt
main > Enter transaction file for vendor 3 = v3
java.io.FileNotFoundException: v3 (The system cannot find the file specified)
main > Enter transaction file for vendor 3 = v3.txt
main > ------------------------------
main > Stock Simulation (1)
main > ------------------------------
v1 > trans 1 +100 bluetooth speaker balance = 100
v1 > trans 3 +100 mini projector balance = 100
v1 > trans 4 +100 power bank balance = 100
v2 > trans 1 +200 bluetooth speaker balance = 300
v3 > trans 1 +500 bluetooth speaker balance = 800
v3 > trans 2 +500 clip lens balance = 500
v1 > trans 8 +500 power bank balance = 600
v2 > trans 2 +200 bluetooth speaker balance = 1,000
v3 > trans 4 +300 power bank balance = 900
v2 > trans 5 +600 mini projector balance = 700
v1 > trans 9 +600 bluetooth speaker balance = 1,600
v3 > trans 6 +100 clip lens balance = 600
v3 > trans 9 +200 power bank balance = 1,100
v1 > trans 10 +600 clip lens balance = 1,200
v2 > trans 8 +500 clip lens balance = 1,700
v2 > trans 10 +300 power bank balance = 1,400
v2 > Buying completes
v2 > clip lens buy = 1,700 sales = 0 balance = 1,700
v2 > bluetooth speaker buy = 1,600 sales = 0 balance = 1,600
v2 > power bank buy = 1,400 sales = 0 balance = 1,400
v2 > mini projector buy = 700 sales = 0 balance = 700
v2 > trans 3 -300 clip lens balance = 1,400
v3 > trans 3 -300 mini projector balance = 400
v2 > trans 4 -300 clip lens balance = 1,100
v3 > trans 5 -300 bluetooth speaker balance = 1,300
v1 > trans 2 -100 clip lens balance = 1,000
v2 > trans 6 -600 power bank balance = 800
v1 > trans 5 -500 bluetooth speaker balance = 800
v3 > trans 7 -200 mini projector balance = 200
v2 > trans 7 -600 bluetooth speaker balance = 200
v1 > trans 6 -500 clip lens balance = 500
v3 > trans 8 -200 mini projector balance = 0
v3 > trans 10 -200 power bank balance = 600
v1 > trans 7 -0 mini projector balance = 0
v2 > trans 9 -0 mini projector balance = 0
v2 > Selling completes
v2 > power bank buy = 1,400 sales = 800 balance = 600
v2 > clip lens buy = 1,700 sales = 1,200 balance = 500
v2 > bluetooth speaker buy = 1,600 sales = 1,400 balance = 200
v2 > mini projector buy = 700 sales = 700 balance = 0
Each output is labeled with thread's name
Balance update must be correct for both buying and selling
Stock summary after buying & selling, in sorted order of balance
Sales amount must not exceed the stock balance

```
main > Run another simulation (y/n) ? y
main > ------------------------------
main >      Stock Simulation (2)
main > ------------------------------

v1 > trans  1   +100  bluetooth speaker   balance =   100
v1 > trans  3   +100  mini projector      balance =   100
v2 > trans  1   +200  bluetooth speaker   balance =   300
v1 > trans  4   +100  power bank          balance =   100
v2 > trans  2   +200  bluetooth speaker   balance =   500
v1 > trans  8   +500  power bank          balance =   600
v2 > trans  5   +600  mini projector      balance =   700
v3 > trans  1   +500  bluetooth speaker   balance = 1,000
v2 > trans  8   +500  clip lens           balance =   500
v1 > trans  9   +600  bluetooth speaker   balance = 1,600
v2 > trans 10   +300  power bank          balance =   900
v3 > trans  2   +500  clip lens           balance = 1,000
v1 > trans 10   +600  clip lens           balance = 1,600
v3 > trans  4   +300  power bank          balance = 1,200
v3 > trans  6   +100  clip lens           balance = 1,700
v3 > trans  9   +200  power bank          balance = 1,400

v3 > Buying completes
v3 > clip lens           buy = 1,700  sales =     0  balance = 1,700
v3 > bluetooth speaker   buy = 1,600  sales =     0  balance = 1,600
v3 > power bank          buy = 1,400  sales =     0  balance = 1,400
v3 > mini projector      buy =   700  sales =     0  balance =   700

v3 > trans  3   -300  mini projector      balance =   400
v2 > trans  3   -300  clip lens           balance = 1,400
v3 > trans  5   -300  bluetooth speaker   balance = 1,300
v2 > trans  4   -300  clip lens           balance = 1,100
v3 > trans  7   -200  mini projector      balance =   200
v2 > trans  6   -600  power bank          balance =   800
v1 > trans  2   -100  clip lens           balance = 1,000
v2 > trans  7   -600  bluetooth speaker   balance =   700
v3 > trans  8   -200  mini projector      balance =     0
v1 > trans  5   -500  bluetooth speaker   balance =   200
v3 > trans 10   -200  power bank          balance =   600
v2 > trans  9     -0  mini projector      balance =     0
v1 > trans  6   -500  clip lens           balance =   500
v1 > trans  7     -0  mini projector      balance =     0

v1 > Selling completes
v1 > power bank          buy = 1,400  sales =   800  balance =   600
v1 > clip lens           buy = 1,700  sales = 1,200  balance =   500
v1 > bluetooth speaker   buy = 1,600  sales = 1,400  balance =   200
v1 > mini projector      buy =   700  sales =   700  balance =     0
```

Another run may give a different thread order, but the stock summaries must be the same as before