Exercise 7 (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 Ex7_xxx where xxx = your full ID. That is, your source files must be in package Ex7_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 Ex7_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.

==

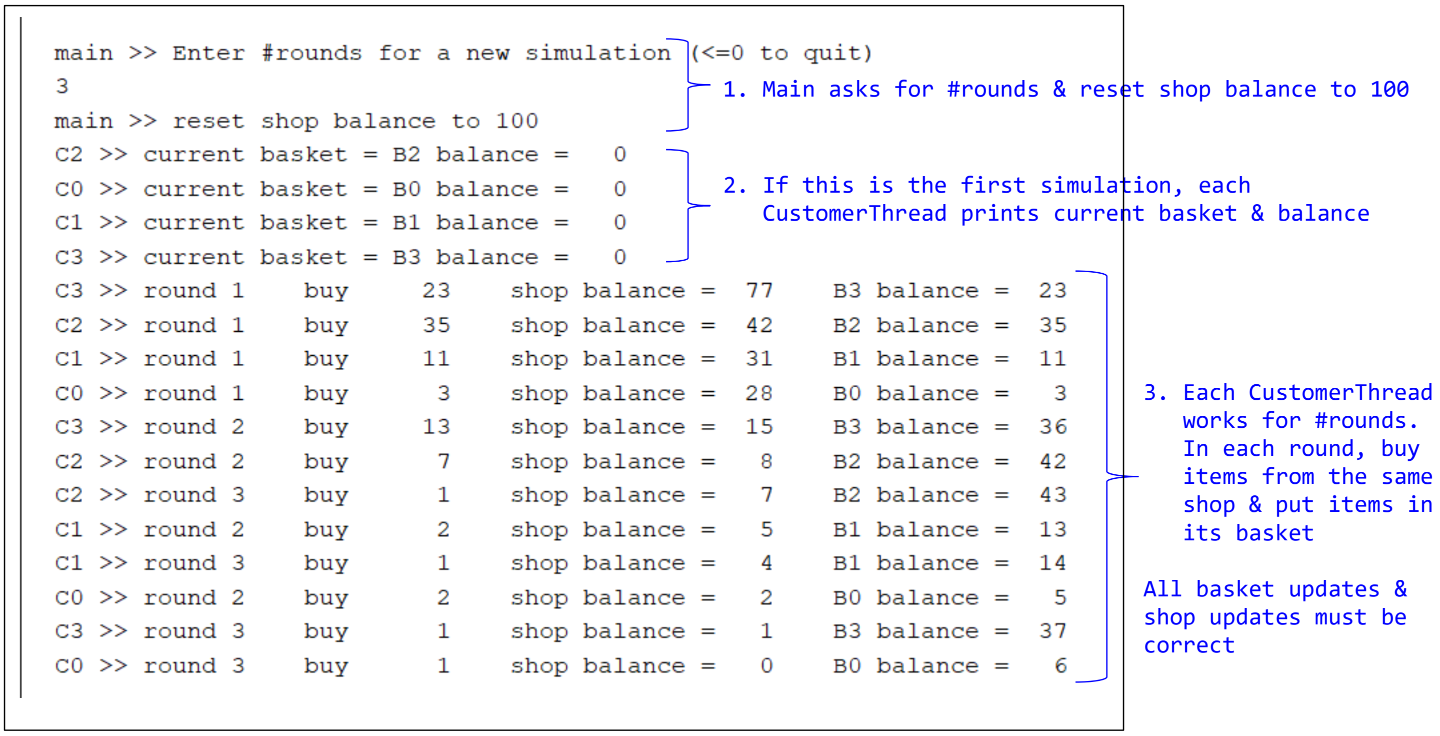
Complete the given source file to make the program work as follows:

1. Complete class CustomerThread. You can add more variables & methods, change method headers, but don't change the visibility of existing ones.
   - Use Exchanger to exchange Basket between 2 CustomerThreads.
   - Use CyclicBarrier to make threads start some tasks at the same time.

2. Complete classes Basket and Shop. You can add more variables & methods, change method headers, but don't change the visibility of existing ones.
   - 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 activities
   - Use CyclicBarrier to make threads start some tasks at the same time
   - Use Join to make main thread wait until all CustomerThreads complete their works before printing final basket balances

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



```
main >> Enter #rounds for a new simulation (<=0 to quit)
3
main >> reset shop balance to 100
C2 >> current basket = B2 balance =   0
C0 >> current basket = B0 balance =   0
C1 >> current basket = B1 balance =   0
C3 >> current basket = B3 balance =   0
C3 >> round 1   buy   23   shop balance =  77   B3 balance =  23
C2 >> round 1   buy   35   shop balance =  42   B2 balance =  35
C1 >> round 1   buy   11   shop balance =  31   B1 balance =  11
C0 >> round 1   buy    3   shop balance =  28   B0 balance =   3
C3 >> round 2   buy   13   shop balance =  15   B3 balance =  36
C2 >> round 2   buy    7   shop balance =   8   B2 balance =  42
C2 >> round 3   buy    1   shop balance =   7   B2 balance =  43
C1 >> round 2   buy    2   shop balance =   5   B1 balance =  13
C1 >> round 3   buy    1   shop balance =   4   B1 balance =  14
C0 >> round 2   buy    2   shop balance =   2   B0 balance =   5
C3 >> round 3   buy    1   shop balance =   1   B3 balance =  37
C0 >> round 3   buy    1   shop balance =   0   B0 balance =   6
```

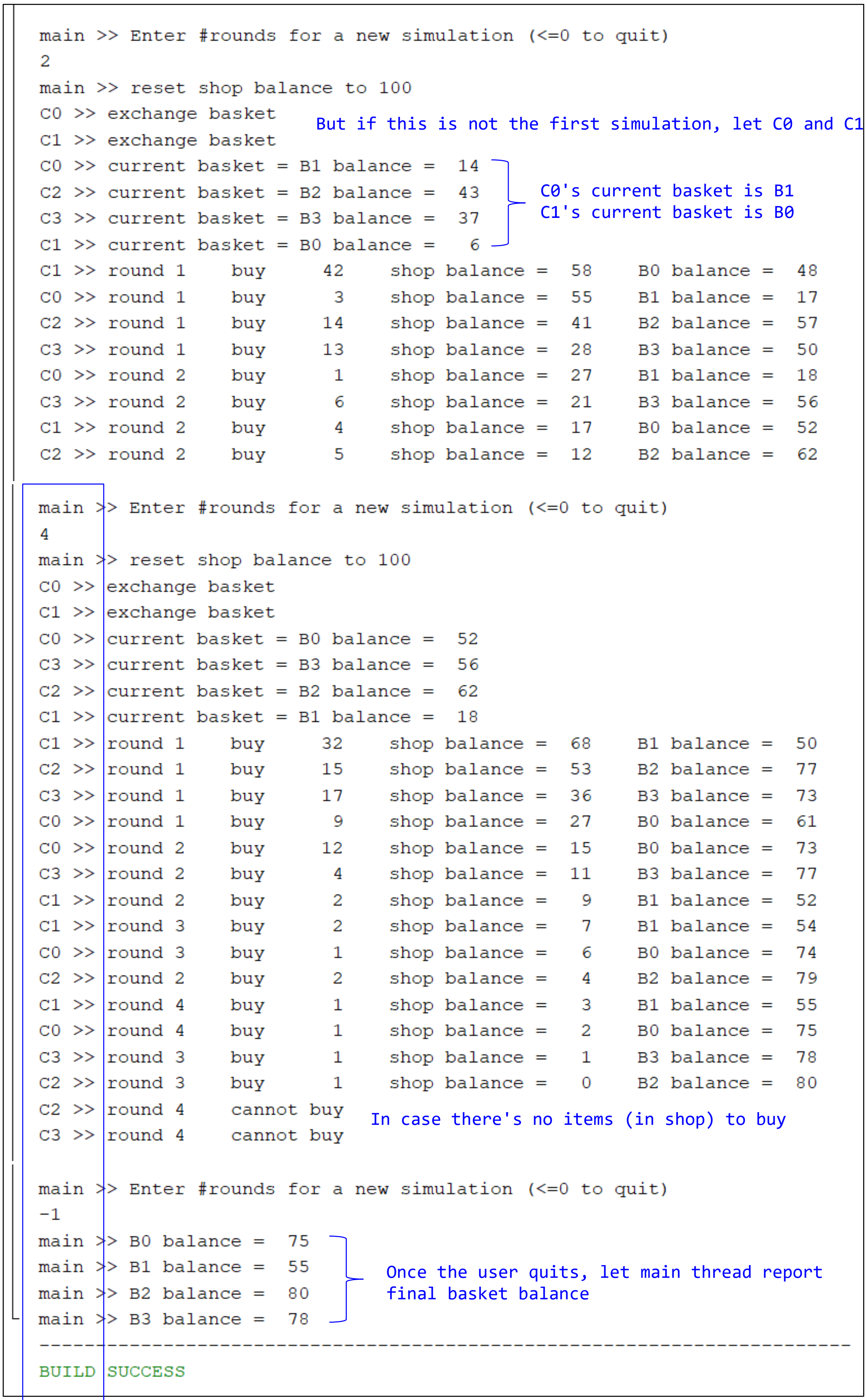
1. Main asks for #rounds & reset shop balance to 100

2. If this is the first simulation, each CustomerThread prints current basket & balance

3. Each CustomerThread works for #rounds. In each round, buy items from the same shop & put items in its basket

All basket updates & shop updates must be correct

Repeat steps 1-3



```
main >> Enter #rounds for a new simulation (<=0 to quit)
2
main >> reset shop balance to 100
C0 >> exchange basket
C1 >> exchange basket
C0 >> current basket = B1 balance =  14
C2 >> current basket = B2 balance =  43
C3 >> current basket = B3 balance =  37
C1 >> current basket = B0 balance =   6
C1 >> round 1    buy     42    shop balance =  58    B0 balance =  48
C0 >> round 1    buy      3    shop balance =  55    B1 balance =  17
C2 >> round 1    buy     14    shop balance =  41    B2 balance =  57
C3 >> round 1    buy     13    shop balance =  28    B3 balance =  50
C0 >> round 2    buy      1    shop balance =  27    B1 balance =  18
C3 >> round 2    buy      6    shop balance =  21    B3 balance =  56
C1 >> round 2    buy      4    shop balance =  17    B0 balance =  52
C2 >> round 2    buy      5    shop balance =  12    B2 balance =  62

main >> Enter #rounds for a new simulation (<=0 to quit)
4
main >> reset shop balance to 100
C0 >> exchange basket
C1 >> exchange basket
C0 >> current basket = B0 balance =  52
C3 >> current basket = B3 balance =  56
C2 >> current basket = B2 balance =  62
C1 >> current basket = B1 balance =  18
C1 >> round 1    buy     32    shop balance =  68    B1 balance =  50
C2 >> round 1    buy     15    shop balance =  53    B2 balance =  77
C3 >> round 1    buy     17    shop balance =  36    B3 balance =  73
C0 >> round 1    buy      9    shop balance =  27    B0 balance =  61
C0 >> round 2    buy     12    shop balance =  15    B0 balance =  73
C3 >> round 2    buy      4    shop balance =  11    B3 balance =  77
C1 >> round 2    buy      2    shop balance =   9    B1 balance =  52
C1 >> round 3    buy      2    shop balance =   7    B1 balance =  54
C0 >> round 3    buy      1    shop balance =   6    B0 balance =  74
C2 >> round 2    buy      2    shop balance =   4    B2 balance =  79
C1 >> round 4    buy      1    shop balance =   3    B1 balance =  55
C0 >> round 4    buy      1    shop balance =   2    B0 balance =  75
C3 >> round 3    buy      1    shop balance =   1    B3 balance =  78
C2 >> round 3    buy      1    shop balance =   0    B2 balance =  80
C2 >> round 4    cannot buy
C3 >> round 4    cannot buy

main >> Enter #rounds for a new simulation (<=0 to quit)
-1
main >> B0 balance =  75
main >> B1 balance =  55
main >> B2 balance =  80
main >> B3 balance =  78
------------------------------------------------------------------------
BUILD SUCCESS
```

But if this is not the first simulation, let C0 and C1 exchange baskets

C0's current basket is B1
C1's current basket is B0

Shop balance is reset at the beginning of each simulation.

But basket balance is accumulated from previous simulations

In case there's no items (in shop) to buy

Once the user quits, let main thread report final basket balance

Every output line must be labelled by thread's name (from Thread.currentThread())