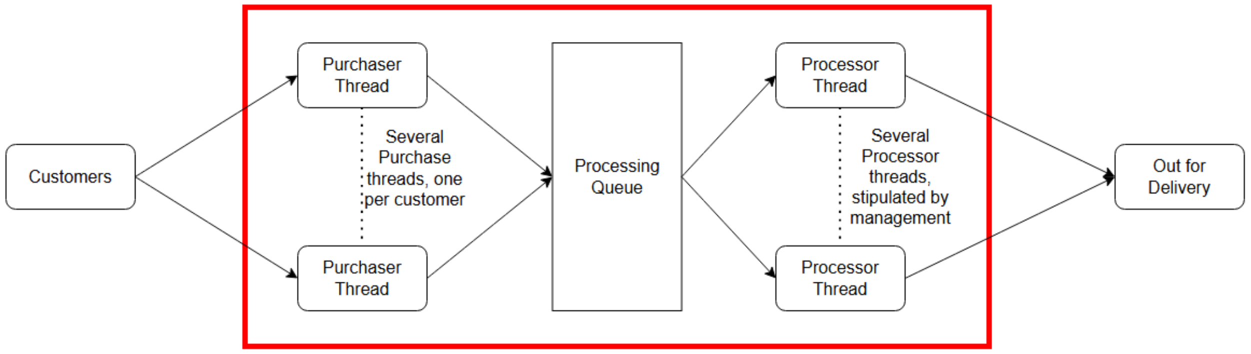
CSE1322L - Assignment 7 (Spring 2025)

Introduction:

In this assignment, we will simulate an online store system. Every time a customer connects to the system, they are assigned a Purchaser thread, which will take their order and store it in a queue for processing. At the end of the business day, the system will create several Processor threads which will traverse the queue and process all orders in parallel, removing said orders from the queue and tallying up the revenue generated. The orders would then be shipped to customers.

The diagram below shows a sketch of how the system is supposed to look like. Specifically, we will implement the red rectangle.



Requirements

The features described below must be in your program.

- An Item class. It has 2 fields: a String called "description", and a double called "cost". Both fields are public.
- An Inventory class, which will be used to keep track of the processing queue and the revenue generated.
  - It has the following fields:
    - A double called "balance", initialized at 0.
    - An integer called "itemsProcessed", initialized at 0.
    - A List of Items called "processingQueue". As the name implies, this field should be treated as a Queue.
  - It has the following methods:
    - **public void addItem(Item)**: Adds the item to the tail of processingQueue.
    - **public Item retrieveItem():** Increments itemsProcessed by 1 and then removes and returns the head of processingQueue.
    - **public void incrementBalance(double):** increments the balance field by the value passed as the argument (e.g.: if the balance is currently 10, a call to incrementBalance(15.5) should set balance to 25.5)
    - **public double retrieveBalance():** returns the value currently stored in the balance field.
    - **public int getQueueSize():** Returns processingQueue's current size.

        - **public int getItemsProcessed():** Returns the current value of itemsProcessed.
- A Purchaser class, which is a subclass of Thread. This class will be used to store items in the processing queue for processing later on.
    - It has the following fields:
        - A static integer called "nextId", initialized at 1.
        - An integer called id.
        - An integer called itemsProcessed.
        - An integer called itemsToProcess.
        - An Inventory called queue.
        - An Item called item.
    - It has the following constructor and methods:
        - **public Purchaser(Inventory, Item, int)**: Initializes the id field with the current content of nextId, after which nextId is incremented by 1. Sets itemsProcessed to 0, and initializes the rest of the fields with the arguments.
        - **public void run():** Override of the superclass method. This method calls queue's addItem(), using its own "item" field as the argument. This must be repeated as many times as itemsToProcess stipulates (e.g.: if itemsToProcess is 200, then addItem() must be called 200 times). For every item added to queue, increment ItemsProcessed by 1. Finally, once all calls to addItem() are done, print the Purchaser's id, followed by how many items it processed, and what the item's description was.
- A Processor class, which is a subclass of Thread. This class will be used to retrieve the items from the processing queue so their total can be tallied up.
    - It has the following fields:
        - A static integer called "nextId", initialized at 1.
        - An integer called id.
        - An Inventory called queue.
        - An integer called numberOfOrders.
        - A double called revenue.
    - It has the following constructor and methods:
        - **public Processor(Inventory)**: Initializes numberOfOrders and revenue to 0. Initializes the id field with the current content of nextId, after which nextId is incremented by 1. Finally, it uses the argument to initialize queue.
        - **public void run():** Override of the superclass method. In a loop, this method must retrieve items from queue and process them:
            - If the queue is empty, terminate the loop.
            - If the queue is not empty, retrieve an Item from the queue.
            - Increment queue's balance by calling incrementBalance() using the Item's cost field.
            - Increment the Processor's numberOfOrders by 1.
            - Increment the Processor's revenue using the Item's cost field.

        - Once the loop terminates, print out the Processor’s id, followed by the number of orders and amount of revenue it processed.
- Driver:
    - Creates an Inventory called “inventory”.
    - The table below shows what merchandise is available at the store, as well as its costs.

Description	**Price**
T-shirt	$6.50
Sweater	$8.50
Sweatpants	$10.00
Skirt	$25.50
Dress	$15.50

    - Create one Item object for each of the items in the table above.
    - Create 5 Purchaser objects. Each should take one of the Item objects created above as their Item argument, as well as “inventory” as their Inventory argument. As for the quantity, prompt the user for it.
    - **After all Purchasers have been created**, start their threads.
    - **After all Purchasers have finished executing**, prompt the user for the number of Processors they want. Create that many Processor objects using “inventory” as the argument for their constructors.
    - **After all Processors have been created**, start their threads.
    - **After all Processors have finished executing**, print out inventory’s getItemsProcessed() and retrieveBalance().

Deliverables

- Assignment7.java (driver)
- Inventory.java
- Item.java
- Processor.java
- Purchaser.java

Considerations

- Remember that you will get partial credit for partial work. Try to deliver as much of the assignment as you can.
- You can add any helper methods you believe are necessary, but they will not count towards your grade.
- Despite what the deliverables above say, you can submit all classes listed under Deliverables in the same file as your driver class.

- You don’t have to worry about rounding your floating-point numbers to a specific decimal place or about rounding errors. As long as your calculations are correct, you will get full credit.
- This assignment requires you to use a List as a queue. Note that any class that implements the List interface can be used, though you should prefer using a concrete class that you are already familiar with.
    - Review the lecture slides for details as to how a queue is supposed to work.
- There will not be any rubric items checking if any race conditions have been prevented.
- Because you are only asked to create 5 Item objects, your queue will contain only copies of those 5 copies.
    - This is being done to preserve memory and to save processing time. Otherwise, your computer would be required to create 80000 Item objects just so you could test one of the sample inputs.
- Your output will differ from the samples below due to the nature of concurrency. Specifically, the order in which your Purchasers and Processors terminate, and how many Items they process, will vary between runs.
    - Still, some of the main()’s print statements must appear at specific locations in your output.
    - Remember that you can make a thread wait for another thread to terminate before continuing its own execution by using join().

Sample Output (user input in red)

```
[Order Queue Simulator]
Purchase how many 't-shirt' at $6.50? 15000
Purchase how many 'sweater' at $8.50? 20000
Purchase how many 'sweatpants' at $10.00? 20000
Purchase how many 'skirt' at $25.50? 15000
Purchase how many 'dress' at $15.50? 10000
Purchasers created. Press 'enter' to start purchases...

Purchasers have started working...
Purchaser 3 starting purchases...
Purchaser 5 starting purchases...
Purchaser 2 starting purchases...
Purchaser 4 starting purchases...
Purchaser 1 starting purchases...
Purchaser 3 purchased 20000 'sweatpants'.
Purchaser 5 purchased 10000 'dress'.
Purchaser 2 purchased 20000 'sweater'.
Purchaser 1 purchased 15000 't-shirt'.
Purchaser 4 purchased 15000 'skirt'.
Purchasers are done working. A total of 80000 items are awaiting processing.

Create how many processors? 10
OrderProcessors created. Press 'enter' to start processing orders...

Processors are now working...
OrderProcessor 2 starting order processing...
OrderProcessor 3 starting order processing...
OrderProcessor 1 starting order processing...
OrderProcessor 4 starting order processing...
OrderProcessor 5 starting order processing...
OrderProcessor 6 starting order processing...
OrderProcessor 8 starting order processing...
OrderProcessor 7 starting order processing...
OrderProcessor 9 starting order processing...
OrderProcessor 10 starting order processing...
OrderProcessor 2 processed a total of 3521 orders for a total of $42920.00
OrderProcessor 9 processed a total of 9631 orders for a total of $118247.00
OrderProcessor 8 processed a total of 9814 orders for a total of $112563.50
OrderProcessor 7 processed a total of 13018 orders for a total of $151670.50
OrderProcessor 6 processed a total of 8459 orders for a total of $113882.50
OrderProcessor 5 processed a total of 8646 orders for a total of $103870.00
OrderProcessor 1 processed a total of 5940 orders for a total of $74004.50
OrderProcessor 10 processed a total of 8809 orders for a total of $114521.50
OrderProcessor 4 processed a total of 5489 orders for a total of $70295.00
OrderProcessor 3 processed a total of 6673 orders for a total of $103025.50
All OrderProcessors are done processing orders.

80000 items were processed for a total of $1005000.00.
Simulation complete.
```

Sample Output (user input in red)

```
[Order Queue Simulator]
Purchase how many 't-shirt' at $6.50? 2000
Purchase how many 'sweater' at $8.50? 2000
Purchase how many 'sweatpants' at $10.00? 2000
Purchase how many 'skirt' at $25.50? 2000
Purchase how many 'dress' at $15.50? 2000
Purchasers created. Press 'enter' to start purchases...

Purchasers have started working...
Purchaser 1 starting purchases...
Purchaser 4 starting purchases...
Purchaser 3 starting purchases...
Purchaser 1 purchased 2000 't-shirt'.
Purchaser 3 purchased 2000 'sweatpants'.
Purchaser 2 starting purchases...
Purchaser 4 purchased 2000 'skirt'.
Purchaser 5 starting purchases...
Purchaser 2 purchased 2000 'sweater'.
Purchaser 5 purchased 2000 'dress'.
Purchasers are done working. A total of 10000 items are awaiting processing.

Create how many processors? 5
OrderProcessors created. Press 'enter' to start processing orders...

Processors are now working...
OrderProcessor 2 starting order processing...
OrderProcessor 4 starting order processing...
OrderProcessor 3 starting order processing...
OrderProcessor 1 starting order processing...
OrderProcessor 5 starting order processing...
OrderProcessor 2 processed a total of 1671 orders for a total of $22509.00
OrderProcessor 3 processed a total of 1765 orders for a total of $24971.00
OrderProcessor 5 processed a total of 2425 orders for a total of $30180.50
OrderProcessor 4 processed a total of 2537 orders for a total of $29711.50
OrderProcessor 1 processed a total of 1602 orders for a total of $24628.00
All OrderProcessors are done processing orders.

10000 items were processed for a total of $132000.00.
Simulation complete.
```