Assignment 2

The assignment was based on multithreading in java. The src folder contains the files which were already provided along with the files Queue.java, PriorityQueue.java (implemented from QueueInterface<V>), Node.java (which extends NodeBase<V>),
Buyer.java(which extends BuyerBase<V>), and Seller.java(which extends SellerBase<V>).

1.Node.java:

- The variables priority and value are inherited from the abstract class signature.
- The method getPriority() returns the priority(int) of the node.
- The method getValue() returns the value(V) stored by the node.

2.Queue.java:

- The variable queue is an array(NodeBase<V>) of size capacity. The variables front(int) and rear(int) are used to store indexes, the capacity(int) denotes the size of the array and the currentSize(int) is used to maintain the current size of the array. All the variables are private.
- The constructor takes capacity as the parameter. Front and rear are initialized to 0 and a new array is declared. Also the variable currentSize is initialized to 0 as well.
- The method int() returns the currentSize, the current size of the array.
- The method isEmpty() returns true if currentSize is equal to 0 and false otherwise.
- The method isFull() returns true if currentSize equals the capacity and false otherwise.
- The method enqueue(NodeBase<V> node) adds a node to the array queue at the position rear if the queue isn't full else it doesn't do anything. Addition of the node is followed by incrementing currentSize and rear.
- The method dequeue() returns node from the front of the array queue if the queue isn't empty and this is followed by shifting the other elements of the array one place left. The variables rear and currentSize are decremented as well. If the queue is empty, null is returned.

3.PriorityQueue.java:

- The variable queue is an array which stores the NodeBase<V> objects. The variables capacity and currentSize are integers, which store the maximum capacity and the current size of the array queue.

- The variable capacity(int) is passed to the constructor. The array queue is initialized along with initialization of the currentSize to 0.
- The method size() returns the currentSize(int) of the array queue.
- The method isEmpty() returns rue if the current size of the queue is 0 and false otherwise.
- The method isFull() returns true if the currentSize of the queue is equal to the capacity and false otherwise.
- The method enqueue() adds a node to the array if the array is empty(i.e. currentSize isn't equal to capacity) according to the priority in the sorted order. This is followed by incrementing the variable currentSize.
- The method dequeue() returns the first element(front) of the queue if the queue is not empty(i.e. The Node with lowest priority) as the queue is already maintained in a sorted order. This is accompanied by left shifting of the elements of the object. If the queue is empty, null is returned.

4.Buyer.java

- The variable catalog is a shared PriorityQueue(NodeBase<Item>), full and empty are condition objects, and lock is Lock object. The variables sleepTime(int) and iteration(int) is also passed to the constructor and the corresponding variables are set by calling methods.
- The method buy() is commonly called by all the Buyer threads, and hence it is synchronized by using reentrant locks. As soon as a particular thread calls the method buy(), the method is eventually locked. If the catalog is empty, the current thread awaits for signal for the full condition from any of the seller thread and temporarily unlocks the lock. If the queue is empty, then an item id removed from the catalog and signal for the empty condition is made to all the buyer threads and the lock is unlocked.

5.Seller.java

- The variables sleepTime and catalogSize are integers, lock is a Lock, full and empty are condition objects, inventory(Queue<NodeBase>) which is shared and the catalog(PriorityQueue<Item>) is also shared among all buyer and seller threads. The sleepTime(int) passed to the constructor and setSleeptime is called which sets the sleepTime to the required value.
- The sell() method is commonly called by many seller threads and therefore it is synchronized by using reentrant locks. The lock object is locked() initially and if the queue is full, then the current thread is temporarily suspended until any buyer threads calls signal on empty. If

the catalog isn't full, an item is dequeued from the inventory and added to the catalog. Then the lock is released and signal is called for full, enabling buyer threads to run buy().

6.Assignment2Driver.java

- A for loop executes which instantiates new Seller objects as well as new Thread objects and starts them.
- Another for loop executes which instantiates new Buyer objects and new Threads for each of them and starts them.