Problem 1: Merchandise sale for Alcheringa 2020

The synchronisation method used in this code is Monitors. As only one thread can have the monitor at a time, so when a thread acquires a lock we say the thread has entered the monitor. All the other threads which are attempting to enter the monitor which is locked will not be able to get the monitor until the first thread exits the monitor.

We write three classes

1) Inventory: Here we store the values of the existing quantities of the Sizes.

```
public class Inventory{
    public static int s;
    public static int m;
    public static int l;
    public static int c;
}
```

2) Merchandise: This class has the main and we take input. We pass on the input through the order function in Order class in order to activate the monitor.
3) Order: This class holds the monitor lock.

```
public class Order extends Thread{
    char size;
    int quantity;
    int orderNum;
    Inventory inv;
    int s,m,l,c;
```

```
public void run()
{
    synchronized(inv)
    {
```

First, we extend the class named Order with Thread class from java.utils.concurrent, Then we'll use start() in Order Function. We'll be using run() function next and use synchronised() function on inventory.
When we call the *start()*, a new thread by default name Thread-0 is created subsequently *run()* method is executed and everything inside it is executed on the newly created thread. But if we directly call the *run()* then no new thread will be created and *run()* will be executed as a normal way and multitasking will not take place.
Then we use the lock on inventory by using synchronised(inv) and synchronized block construct takes on the object in parentheses here inventory. Only one thread can execute inside a Java code block synchronized on the same monitor object.
So synchronization is achieved.