PDP Lab 1 Documentation

# Chitimus Mara - Group 932

The chosen problem is “Supermarket Inventory”. During the execution of the operations on multiple threads, we have to take into account which data to protect in order to obtain consistent results.

* When the quantity of a product is decreased, we have to ensure that no two threads are modifying the same product as the same time
* When the profit is increased after a sale, we have to make sure that no two threads are simultaneously modifying the value of the income

I have chosen 2 methods of synchronizing the threads in order to protect the integrity of the data, both being implemented in the repository classes.

* MainLockInventory: in this situation, I used the ***synchronized*** keyword over the currentStock object, which contains all the products and their respective available quantity; in this case, each thread has to wait for the other threads in order to release the lock on the currentStock object
* ProductLockInventory: in this case, I used the ***synchronized*** keyword only over the quantity of the product that was currently being modified, in order to allow other threads to modify different products from the currentStock

In both cases, though, the income field and the list of processed bills are “locked”.

The *checkInventory()* method checks that the profit is justified by the bills – this is the invariant

I vary the numbers of threads running and the type of repository. Tests have been done with 10000 products in the stock, having an initial quantity of 100000. Each thread runs 100 operations with randomly chosen parameters. In order to have a better stats, I decided to run each test configuration 10 times and make an average of the obtained runtimes.

|  |  |  |
| --- | --- | --- |
| **Threads** | **Repository Type** | **Average RunTime (ms)** |
| 12 | MainLock | 5.90 |
| 12 | ProductLock | 3.50 |
| 8 | MainLock | 3.20 |
| 8 | ProductLock | 3.01 |
| 5 | MainLock | 6.60 |
| 5 | ProductLock | 4.10 |