**Inventory System Documentation**

**Overview**

The Inventory system manages products, sales, and an inventory check process in a concurrent environment. The system uses multiple threads for sales operations and periodically checks the inventory to ensure data integrity. This documentation details the synchronization mechanisms in place to protect shared resources.

**Mutexes and Invariants**

**1. Product Locks**

* **Mutex**: Lock in the Product class (e.g., product.getLock()).
* **Invariants Protected**:
  + **Product Quantity**: Each product maintains its available quantity. The lock ensures that concurrent sales operations on the same product do not lead to race conditions, ensuring the quantity is accurately decremented.
  + **Sales Operations**: The lock protects the operations of selling a product, ensuring that only one thread can modify the quantity of a product at a time.

**2. Money and Sales Records Lock**

* **Mutex**: moneyAndSalesLock in the Inventory class.
* **Invariants Protected**:
  + **Total Money**: This lock protects the totalMoney variable, ensuring that concurrent updates from multiple sales threads do not lead to inconsistencies in the total amount of money collected from sales.
  + **Sales Records**: It protects the salesRecords list, ensuring that the addition of new bills does not result in lost updates or incorrect state during concurrent access.

**3. Inventory Check Process**

* **Mutex**: moneyAndSalesLock is also used during the inventory check.
* **Invariants Protected**:
  + **Integrity of Sales Data**: During the inventory check, the lock ensures that no new sales are being processed, allowing for a consistent snapshot of sales records and total money.
  + **Consistency Between Bills and Inventory**: The inventory check verifies that all products sold match the recorded sales. The lock ensures that the data used for this verification is consistent and not being modified by ongoing sales operations.