# Exercise 1 - Inventory Management System

## Understand the Problem:

## Explain why data structures and algorithms are essential in handling large inventories.

**Answer:**

Because they dictate how effectively data is stored, accessed, updated, and deleted, data structures and algorithms are essential to the management of large inventories. Constant-time operations are made possible by a carefully selected data structure (such as a hash map), which is essential when working with thousands of products. Effective algorithms guarantee seamless operation, cut down on processing time, and preserve scalability as the inventory expands.

## Discuss the types of data structures suitable for this problem.

**Answer:**

HashMap (Dictionary): Perfect for quick updates, deletions, and lookups with a distinct key, such as productId, provides key operations with average-case O(1) time complexity.  
  
ArrayList: Although searching and deleting by ID takes O(n) time, array lists are useful when products need to be kept in a sequential order and frequently traversed.  
  
TreeMap: Helpful when productId or another attribute sorting is required.  
  
LikedList: Less frequently used here, unless frequent additions or deletions at particular locations are required, is the linked list.  
  
HashMap is the best option for quick, ID-based product management.

## Implementation:

## Analysis

## 3.1 Analyze the time complexity of each operation (add, update, delete) in your chosen data structure.

**Answer:**

HashMap is utilized  
  
Add Product: O(1) average case; key is inserted directly.  
  
Product Update: O(1) average case: overwrite values and access by key.  
  
Delete Product: remove by key in O(1) average case.  
  
Display inventory: Iterate over every entry using O(n).  
  
Hash collisions can cause worst-case time to drop to O(n), although this is uncommon in practice when hashing is done correctly.

## 3.2 Discuss how you can optimize these operations.

**Answer:**

* To minimize collisions in the HashMap, we ensure productId is a widely distributed key.
* We reduce the number of pointless traversals by using direct key access rather than iterating through the collection.