**Exercise 1: Inventory Management System**

**Scenario:**

You are developing an inventory management system for a warehouse. Efficient data storage and retrieval are crucial.

**Why are Data Structures & Algorithms Important?**

* **Fast Lookup:** In large inventories, finding a product quickly is vital — especially by unique keys like productId.
* **Efficient Modifications:** Adding, updating, or deleting product information must be done efficiently without scanning the entire list.
* **Scalability:** Efficient data structures help keep the system performant as inventory grows to thousands or millions of items.

**Suitable Data Structures – Comparison in Points**

* **ArrayList**
  + Maintains insertion order.
  + Easy to iterate through.
  + Searching by product ID takes **linear time** – O(n).
* **LinkedList**
  + Efficient for **frequent insertions/deletions** at the beginning or end.
  + **Slow access time** – O(n) for searching or accessing by index.
  + Not ideal for search-heavy use cases.
* **HashMap<String, Product>**
  + **Fast access, insert, and delete** operations – average **O(1)** time.
  + Ideal for **key-based lookups** using productId.
  + Does **not maintain order** of insertion or elements.

**Best Choice:**

* **HashMap** is the most suitable data structure for scenarios requiring fast lookups using unique keys like productId → Product.

**Analysis**

**Time Complexity (Using HashMap)**

|  |  |  |
| --- | --- | --- |
| **Operation** | **Time Complexity** | **Reason** |
| **addProduct** | **O(1) average** | **put() in HashMap is constant time** |
| **updateProduct** | **O(1) average** | **get() + field updates** |
| **removeProduct** | **O(1) average** | **remove() from HashMap** |
| **displayInventory** | **O(n)** | **Iterating over all values in HashMap** |

**In the worst case (hash collisions), operations could degrade to O(n), but this is rare with good hash functions.**

**How to Optimize Further?**

* **Use ConcurrentHashMap for thread-safe operations in multi-threaded environments.**
* **Index products by multiple attributes (e.g., productName) using secondary maps.**
* **Add caching or lazy loading if using persistent storage (e.g., database).**
* **Use binary search trees if you want sorted access (TreeMap).**