**Exercise 1: Inventory Management System**

**Scenario:**

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

**Steps:**

1. **Understand the Problem:**

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

I will discuss this on the context of data structures and algorithms, as I walk through how software engineers manage large inventories using efficient organization and where to store this information in memory. Since there are a ton of products, the focus is on how fast the operations would be regarding the actions that will take place in real-time and should scale as well such as adding or updating product / deleting it. Slow performance, high computational costs and poor user experience can be the result of poor data handling. If data structure is correct operations can be performed in best possible way which will automatically reduce time complexity and use of resources.

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

1. ArrayList: It is suitable for situations where the data needs to be accessed sequentially and the size of inventory is not fixed. Allows fast random access but slow insertions and deletions in the middle of the list.
2. HashMap: Used when frequent look-ups, insertion or deletion are needed, i.e it uses a hashing algorithm to find object. It enables these operations to be done in constant time complexity[O(1)] on average.
3. **Setup:**
   * Create a new project for the inventory management system.
4. **Implementation:**
   * Define a class Product with attributes like **productId**, **productName**, **quantity**, and **price**.
   * Choose an appropriate data structure to store the products (e.g., ArrayList, HashMap).
   * Implement methods to add, update, and delete products from the inventory.
5. **Analysis:**

* Analyze the time complexity of each operation (add, update, delete) in your chosen data structure.
* Add product - Average time complexity is O(1).
* Update product - Average time complexity is O(1).
* Delete product - Average time complexity is O(1).
* Discuss how you can optimize these operations.

When creating the HashMap, an initial capacity and load factor can be set to minimize rehashing operations if there is a good estimate of the number of products.