**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'll discuss how software engineers handle large inventories through effective data structure and algorithm management. With numerous products to manage, the focus is on ensuring fast operations for real-time actions such as adding, updating, or deleting items. Poor data management can result in slow performance, increased computational costs, and a negative user experience. Using the right data structures helps in performing these operations optimally, which reduces time complexity and resource consumption.

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

1. **ArrayList:** This data structure is ideal for scenarios where data access is sequential and the inventory size can vary. It provides quick random access but can be slow when inserting or deleting elements in the middle of the list.
2. **HashMap:** This structure is designed for situations requiring frequent look-ups, insertions, or deletions. It employs a hashing algorithm to locate objects, making these operations average constant time complexity [O(1)].
3. **Setup:**
   * Create a new project for the inventory management system.
4. **Implementation:**

* **Define a Product class** with attributes such as productId, productName, quantity, and price.
* **Select a suitable data structure** for storing products (e.g., ArrayList or HashMap).
* **Implement methods** to add, update, and delete products within the inventory.

1. **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 initializing the HashMap, setting an initial capacity and load factor can help minimize rehashing operations, especially if you have a reliable estimate of the number of products.