**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.

Solution:

Data structures and algorithms are essential in handling large inventories because they enable efficient storage, retrieval, and manipulation of data. A well-designed data structure can reduce the time complexity of operations, making it possible to manage large datasets quickly.

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

Solution :

Types of data structures suitable for inventory management include:

* **ArrayList**: Useful for maintaining a list of products with the ability to access elements by index. However, operations like insertion and deletion can be slow for large lists due to the need for shifting elements.
* **HashMap**: Provides fast access, insertion, and deletion of products using unique keys (e.g., product IDs). It is particularly suitable for large inventories where quick lookups are necessary.

1. **Setup:**
   * Create a new project for the inventory management system.
2. **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.

Solution:

Implementation part is in java file attached in folder Ecercise\_1.

1. **Analysis:**
   * Analyze the time complexity of each operation (add, update, delete) in your chosen data structure.

Solution :

Time Complexity

1. Add Product:
   * Using HashMap.put(), the average time complexity is O(1)O(1)O(1).
2. Update Product:
   * Since updating is done using HashMap.put(), the average time complexity is O(1)O(1)O(1).
3. Delete Product:
   * Using HashMap.remove(), the average time complexity is O(1)O(1)O(1).
   * Discuss how you can optimize these operations.

Solution:

Optimization

* Load Factor: Ensure the load factor of the HashMap is maintained properly to avoid excessive rehashing, which can degrade performance.
* Concurrent Operations: For a multi-threaded environment, consider using ConcurrentHashMap to handle concurrent access efficiently.
* Data Validation: Implement data validation checks before adding or updating products to maintain data integrity.