**Exercise 1: Inventory Management System**

**Scenario:**

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

1. why data structures and algorithms are essential in handling large inventories ?

Because :

* **Efficient Data Handling**: In a warehouse, inventory can involve thousands of products. Efficient data structures ensure quick access, modification, and storage of this data, which is crucial for maintaining accurate stock levels and facilitating smooth operations.
* **Scalability:** As the number of products grows, operations like searching, adding, updating, and deleting items need to remain efficient to avoid performance bottlenecks.
* **Resource Management:** Proper algorithms and data structures help in optimizing memory usage and processing power, ensuring the system runs smoothly even under high load.

2. Discuss the types of data structures suitable for this problem?

Suitable Data Structures:

* **ArrayList:** Good for storing products when frequent read operations are needed, and the list size is manageable. Provides fast random access but slow for add/delete operations (O(n)).
* **HashMap:** Ideal for quick lookups, updates, and deletions (average O(1) time complexity). Suitable when product IDs are unique keys. Provides efficient access but might require more memory.

3. Setup:

**Project Creation**: create a project name **InventoryManagementSystem.**

* **Classes** :
* Product : Represents individual products.
* InventoryManagementSystem : Manages the collection of products.
* Main : Entry point to demonstrate the functionality.

4. Implementation:

* **Class Definitions:**
* Product Class:

This is a simple data class that represents a product in the inventory.

It has attributes like productId, productName, quantity, and price.

It provides getters and setters for these attributes and a toString() method for easy printing.

* InventoryManagementSystem Class:

This class manages the inventory using a HashMap to store products.

Key methods include:

addProduct(): Adds a new product to the inventory.

updateProduct(): Updates an existing product.

deleteProduct(): Removes a product from the inventory.

getProduct(): Retrieves a product by its ID.

displayAllProducts(): Shows all products in the inventory.

saveToFile(): Saves the inventory data to a file.

loadFromFile(): Loads inventory data from a file.

* Main Class:

This is the entry point of the program and handles user interaction.

It creates an instance of InventoryManagementSystem.

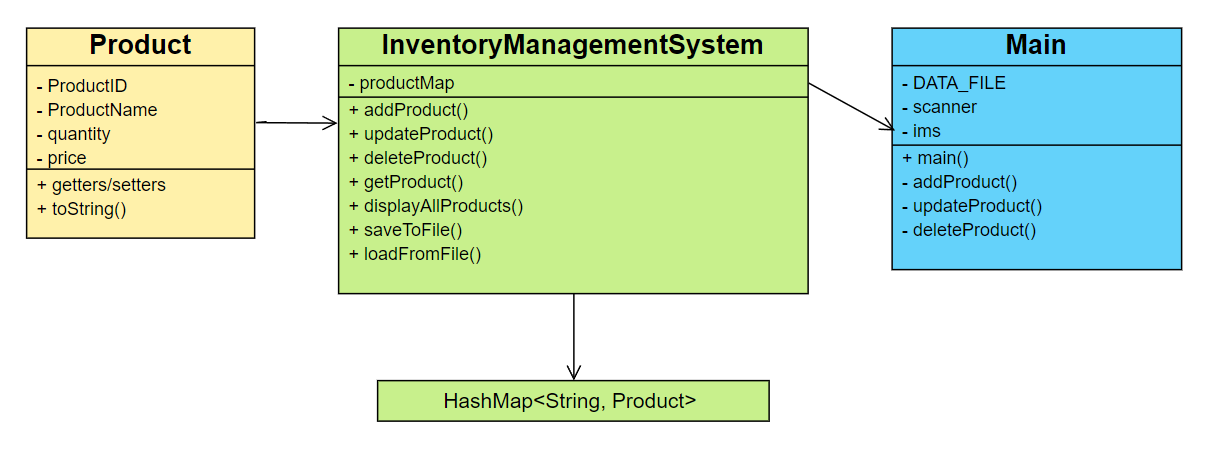
It provides a menu-driven interface for the user to interact with the system.

Key methods include:

main(): The main loop that displays the menu and processes user choices.

addProduct(), updateProduct(), deleteProduct(): Helper methods to handle user input for these operations.

LINK: [Click here for code.](https://github.com/Akashmondal55/Akash_5016855/tree/main/Week-1/DSA/Exercise-1)

* **Class diagram:** To describe the program through diagram
* **Data Structure:** To store the product I use Hash map where the key is **productid**

productMap = {

"A01" -> Product{id="A01", name="Laptop", quantity=10, price=40000},

"A02" -> Product{id="A02", name="Mouse", quantity=50, price=500},

"A03" -> Product{id="A03", name="Keyboard", quantity=30, price= 600}

……

}

The hash map is structured in this way.

* **Program flow:**

a. The program starts in the main() method of the Main class.

b. It first loads any existing data from the file specified by DATA\_FILE.

c. It then enters a loop, displaying a menu of options to the user.

d. Based on the user's choice, it calls the appropriate method:

For adding a product, it collects input and creates a new Product object, then calls ims.addProduct().

For updating, it first retrieves the product, then collects new information and updates.

For deleting, it simply calls ims.deleteProduct() with the provided ID.

For displaying all products, it calls ims.displayAllProducts().

e. When the user chooses to exit, it saves the current inventory to the file and terminates.

* **Data Persistence:**

The system uses simple file I/O to save and load inventory data. Data is stored in a CSV-like format in a .txt file called **Inventory.txt**, with each product on a new line. When the program starts, it attempts to load existing data. When the user exits, it saves the current state of the inventory.

Here is the github repo link –

5. Analysis:

* **Time Complexity:**
* Add operation: O(1) average case, as HashMap insertion is constant time.
* Update operation: O(1) average case, as HashMap lookup and modification are constant time.
* Delete operation: O(1) average case, as HashMap removal is constant time.
* **Optimization strategies:**
* Use a proper initial capacity for the HashMap to minimize rehashing. Implement a custom hash function if product IDs have a specific pattern. Consider using a concurrent HashMap for thread-safe operations in a multi-threaded environment.
* Implement periodic cleanup of deleted or outdated entries to maintain efficiency.
* Use lazy deletion (marking items as deleted without removing them immediately) if frequent deletions occur.