**INVENTORY MANAGEMENT SYSTEM**

**Understand The Problem**

* **Explain why data structures and algorithms are essential in handling large inventories?**
* **Data structures** are essential for efficiently organizing and accessing large inventory data, enabling quick searches, updates, and management tasks, which are crucial for maintaining system performance and reliability.
* **Algorithms** optimize inventory tasks like sorting and searching, ensuring fast and accurate processing of large volumes of items, thus maintaining real-time accuracy and operational efficiency.
* **Discuss the types of data structures available for this program**
* **HashMap**: This data structure, used in the provided code, maps product IDs to Product objects. HashMap offers average-case constant-time complexity (O(1)) for operations such as adding, updating, and deleting items, making it highly efficient for these tasks.
* **Array List**: While Array List can store products, it is less efficient for searching and deleting since these operations require linear time (O(n)) to locate a product by ID. However, it is suitable for scenarios requiring ordered storage or frequent access by index.

**Analysis**

* **Analyze the time complexity of each operation (add, update, delete) in your chosen data structure.**
* **Add Product**: In a HashMap, the average time complexity for adding a product is O(1). This efficiency arises because adding an entry involves computing a hash and inserting the product into the appropriate bucket.
* **Update Product**: The average time complexity for updating a product in a HashMap is O(1). This is due to the direct access capability of HashMap, where updating involves simply replacing the value associated with the specified key.
* **Delete Product**: Deleting a product from a HashMap typically has an average time complexity of O(1). This process involves locating the relevant bucket using the key and removing the entry from that bucket.
* **Discuss how you can optimize these operations.**

To maintain O(1) performance, it's important to use a hash function that evenly distributes keys to prevent collisions. Additionally, managing the load factor and resizing the hash table as necessary ensures efficient operation.