**Exercise 1: Inventory Management System**

1. **Understand the Problem**

**a. Importance of Data Structures and Algorithms:**  
Efficient data structures and algorithms are essential for managing large inventories because they help in:

* **Fast lookup and updates** of product details.
* **Optimized memory usage** to store a large number of items.
* **Scalability** when handling growth in product count or queries.

**b. Suitable Data Structures:**

* **ArrayList**: Useful for small lists, ordered storage, but searching is O(n).
* **HashMap**: Ideal for key-based retrieval (like productId), offering O(1) average time complexity for operations.
* **TreeMap**: Keeps elements sorted by key but with O(log n) operations.

**Best choice here**: HashMap<Integer, Product> — efficient for warehouse inventory where quick access by productId is critical.

1. **Analysis**

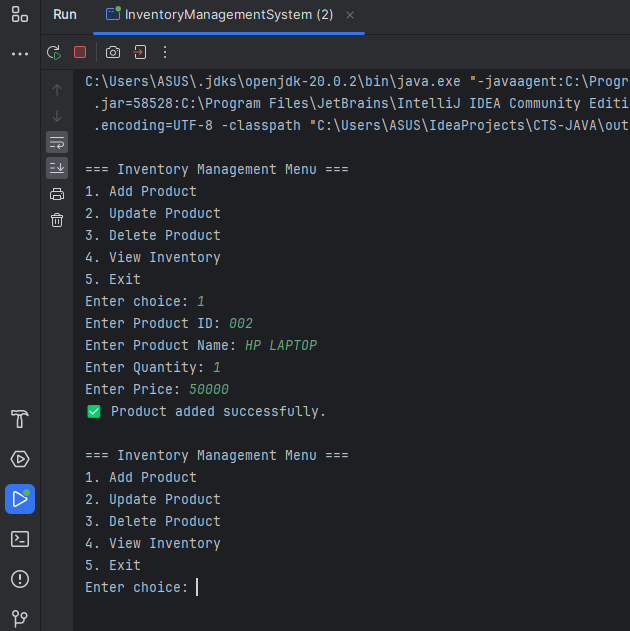
* **Add Product:** The addition of a product to the inventory using a HashMap is efficient, with an average time complexity of **O(1)**. This is because put() in HashMap allows constant-time insertion.
* **Update Product:** Updating an existing product also uses the put() method. If the product ID already exists, it replaces the value. This operation also has a time complexity of **O(1)** on average.
* **Delete Product:** Deleting a product by ID is done using the remove() method, which operates in **O(1)** time on average.
* **Search Product:** Retrieving a product by ID with get() takes **O(1)** time on average as well, making it very fast for lookups.

**Optimization Suggestions**

To further improve performance:

* Use **batch processing** when dealing with multiple product updates or inserts.
* Integrate a **database** for persistence and handling large-scale data across sessions.
* Add **filters and indexing** if more advanced queries are required (like searching by product name or range of prices).

1. **Output**

****