**Time Complexity Analysis (Using ArrayList):**

* **Add Operation (addItem)**
  + **Time Complexity: O(1) amortized**
  + **Reason: Adding at the end of an ArrayList is typically constant time, unless internal resizing occurs.**
* **Update Operation (modifyItem)**
  + **Time Complexity: O(n)**
  + **Reason: A linear search is required to find the item by ID.**
* **Delete Operation (removeItem)**
  + **Time Complexity: O(n)**
  + **Reason: Linear search for ID and then shifting elements to fill the gap.**
* **Display Operation (showItems)**
  + **Time Complexity: O(n)**
  + **Reason: All items are visited once.**

**Optimizations:**

* **Use a HashMap<Integer, StockItem> instead of ArrayList<StockItem>:**
  + **✅ Add: O(1)**
  + **✅ Update: O(1)**
  + **✅ Delete: O(1)**
  + **✅ Search: O(1)**
  + **Explanation: HashMap provides constant-time access using keys (e.g., item ID) with better scalability for larger inventories.**
* **If order of insertion matters, consider LinkedHashMap.**
* **For sorted inventories, consider TreeMap (provides O(log n) operations and natural ordering).**