📦 Inventory Tracking System using ArrayList

Objective

You are tasked with developing an **Inventory Tracking System** in Java that monitors and manages items stored in a warehouse.
The system must store **unique item names** along with their respective **quantities** using an `ArrayList`.

The application must support core operations such as:

- Registering new inventory items
- Searching for a specific item
- Removing an item from the inventory
- Calculating the average quantity of all remaining items
- Displaying the current list of inventory items

The system must prevent adding duplicate items.
After every critical operation, the system must display the updated list of active items and their respective quantities in the specified output format.

This project simulates a **real-world warehouse inventory backend** and must follow a **modular, testable structure** using clearly named classes and methods.

📁 Project Structure and Files

The project must contain the following Java files:

1. **InventorySystem.java**
   Implements the core logic using `ArrayList<String>` for item names and `ArrayList<Integer>` for their quantities.
2. **InventoryApp.java**
   Contains the `main()` method, handles input using `Scanner`, and invokes methods from `InventorySystem`.

🧩 Class Responsibilities

1. InventorySystem

Must declare two private fields:

- `ArrayList<String> itemList`
- `ArrayList<Integer> quantityList`

Must provide the following public methods with exact names and signatures:

java Copy code

```java
public void registerItem(String name, int quantity)
public void searchItem(String name)
public void removeItem(String name)
public void displayInventoryList(String context)
public void calculateAverageQuantity()
```

Each method should implement the corresponding functionality and print messages in the exact output format described below.

2. InventoryApp

- Reads input using `Scanner`.
- Registers `n` items and their respective quantities.
- Performs a search and remove operation as per the input.
- Calls methods from `InventorySystem` to carry out functionality and display results.
- Does not throw or handle any custom exceptions (none are required).

⚙️ Functional Requirements

Registering Inventory Items

- Accepts `n` as the number of items to record.
- Accepts `n` pairs of inputs: a string (item name) and an integer (quantity).
- Rejects duplicate item names and prints the appropriate error message.

Searching for an Item

- Accepts an item name and prints whether it is present in the system.

Removing an Item

- Accepts an item name and removes it if found.
- If the item is not found, prints the appropriate message.

Displaying Inventory List

- Displays the current list of active items and their respective quantities.

Calculating Average Quantity

- Computes the average of all remaining quantities **rounded to two decimal places**.
- Displays the result in the required format.

📥 Input Format

1. The first input is an integer `n`, representing the number of inventory items to be recorded.
2. The next `n` pairs of lines contain:
    - A string (**Item Name**)
    - An integer (**Quantity in stock**)
3. The next input is an item name to search for.
4. The next input is an item name to remove.

📤 Output Format

Registering Inventory Items

- If a duplicate item is entered, print:

```csharp
Error: Duplicate item '[name]' is not allowed.
```

- After adding valid items, print:

```yaml
Inventory list after adding items:
```

  followed by each item and its quantity on a new line.

Searching for an Item

- If found, print:

```csharp
Item '[name]' is in the inventory.
```

- If not found, print:

```csharp
Item '[name]' is not in the inventory.
```

Removing an Item

- If removed, print:

```nginx
Item '[name]' has been removed.
```

- If not found, print:

```pgsql
Item '[name]' is not found in the inventory.
```

Displaying the Updated Inventory List

- Print:

  ```yaml
  Updated inventory list after removal:
  ```

  followed by each remaining item and its quantity on a new line.

Calculating the Average Quantity

- Print:

  ```less
  Average quantity across items: [average] units
  ```

🧪 Sample Test Cases

Sample Input 1

```arduino
4
Laptop
50
Keyboard
120
Monitor
80
Mouse
200
Monitor
Laptop
```

Sample Output 1

```mathematica
Inventory list after adding items:
Laptop - 50 units
Keyboard - 120 units
Monitor - 80 units
Mouse - 200 units
Item 'Monitor' is in the inventory.
Item 'Laptop' has been removed.
Updated inventory list after removal:
Keyboard - 120 units
Monitor - 80 units
Mouse - 200 units
Average quantity across items: 133.33 units
```

Sample Input 2

```mathematica
3
Notebook
300
Pen
500
Notebook
250
Eraser
Notebook
```

Sample Output 2

```csharp
Error: Duplicate item 'Notebook' is not allowed.
Inventory list after adding items:
Notebook - 300 units
Pen - 500 units
Item 'Eraser' is not in the inventory.
Item 'Notebook' has been removed.
Updated inventory list after removal:
Pen - 500 units
Average quantity across items: 500.00 units
```