Inventory Management System

Outline

1- **Description & Features**

2- **Testing**

3- **Analysis of code**

4- **GitHub Repository**

Project description

The Inventory Management System is a simple, console-based application designed to simulate the management of a store's inventory. This system enables users to perform various inventory management tasks such as adding or updating stock, placing orders, processing pending orders, generating reports, deleting items, and retrieving the history of completed orders. The main goal of this project is to provide a straightforward simulation of inventory operations in a store using basic data structures

Features:

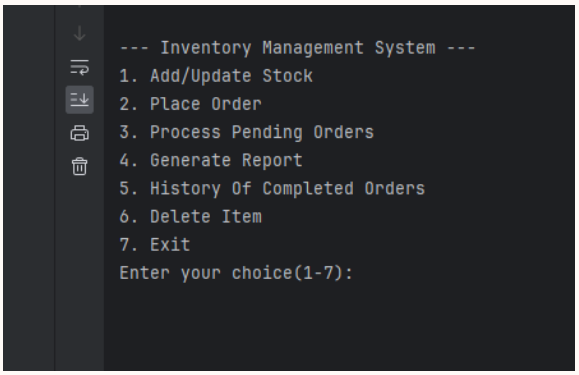
- **Add/Update Stock:** Users can add new items to the inventory or update the quantity of existing items. The system provides feedback if an item is being updated (e.g., from an old quantity to a new quantity).
- **Place Order:** Users can place orders for items. If the item is in stock and the quantity is available, the order is processed immediately. If the stock is insufficient, the order is added to a queue and remains pending until the item is restocked.
- **Process Pending Orders:** The system checks for pending orders and processes them based on stock availability. If an item is out of stock, the order is requeued and remains pending until enough stock is available. Once an order is processed, it is considered completed, and its details are stored in the order history.

- **Generate Report:** A report is generated displaying the current inventory, including item names and quantities. It also lists pending orders that are yet to be processed.
- **Delete Item:** Users can delete items from the inventory. If an item is not found in the inventory, the system notifies the user.
- **Order History:** The system maintains a history of completed orders. Users can view the history of all successfully processed orders, displaying the item name and quantity for each completed transaction.

Data Structures Used:

- **HashMap (for Inventory):** The inventory is stored using a HashMap where the key is the item name (String), and the value is a Product object containing the item's quantity.
- **Queue (for Pending Orders):** Orders that cannot be fulfilled immediately are stored in a Queue (using a LinkedList), following the FIFO (First-In-First-Out) principle. Once the stock is available, these orders are processed.
- **Stack (for Completed Orders):** Completed orders are stored in a Stack, allowing for easy retrieval and maintaining the order in which they were completed.

Testing: -



First, we need to add items to stock:

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 1
Enter item name: apple
Enter quantity: 15
15 units of apple added to inventory.
```

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 1
Enter item name: bannana
Enter quantity: 30
30 units of bannana added to inventory.
```

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 1
Enter item name: orange
Enter quantity: 10
10 units of orange added to inventory.
```

Now generate a report of items and its quantity in stock and pending orders.

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 4

--- Inventory Report ---
orange: 10 units
apple: 15 units
bannana: 30 units

--- Pending Orders ---
No pending orders.
```

Now adding orders completed

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 2
Enter item name: bannana
Enter quantity: 20
Order processed: 20 units of bannana
```

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 2
Enter item name: apple
Enter quantity: 5
Order processed: 5 units of apple
```

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 2
Enter item name: orange
Enter quantity: 3
Order processed: 3 units of orange
```

Now adding orders not completed (pending orders)

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 2
Enter item name: orange
Enter quantity: 25
Order queued: 25 units of orange
```

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 2
Enter item name: apple
Enter quantity: 20
Order queued: 20 units of apple
```

Now generate a report

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 4

--- Inventory Report ---
orange: 7 units
apple: 10 units
bannana: 10 units

--- Pending Orders ---
apple:20
orange:25
```

Now update stock of items that have pending orders

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 1
Enter item name: orange
Enter quantity: 40
Updated stock for orange: 7 -> 47
```

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 1
Enter item name: apple
Enter quantity: 30
Updated stock for apple: 10 -> 40
```

Now process pending orders

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 3

Processing Pending Orders...
Order fulfilled: 20 units of apple
Order fulfilled: 25 units of orange
```

Now view history of completed orders (oldest first)

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 5

--- Completed Orders History ---
apple:5
bannana:20
orange:3
apple:20
orange:25
```

Now remove an item

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 6
Enter item name to delete: bannana
Item 'bannana' removed from inventory.
```

Now generate a report

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 4

--- Inventory Report ---
orange: 22 units
apple: 20 units

--- Pending Orders ---
No pending orders.
```

Now exit console

```
--- Inventory Management System ---
1. Add/Update Stock
2. Place Order
3. Process Pending Orders
4. Generate Report
5. History Of Completed Orders
6. Delete Item
7. Exit
Enter your choice(1-7): 7
Exiting system...
```

Analysis of code

-Class: Product

The Product class is designed to represent an individual product in the inventory.

It holds the *product's name* and *quantity*, and includes methods to manage the stock, such as

- adding to the quantity (**addStock()**)
- reducing the quantity (**reduceStock()**).

Here is a breakdown of the class:

Fields:

- **name**: A String representing the product's name.
- **quantity**: An int representing the product's stock count.

Constructor: The constructor takes name and quantity as parameters to initialize the product.

Methods:

- getQuantity(): Returns the current stock quantity of the product.
- addStock(int quantity): Increases the stock by the specified quantity and returns the updated Product object.
- reduceStock(int quantity): Reduces the stock by the specified quantity.

-Class: InventorySystem

The InventorySystem class is designed to manage the inventory of items in a store. It allows the user to add or update stock levels, place orders, process pending orders, generate inventory reports, delete items, and maintain a history of completed orders. The class uses a combination of a Map, Queue, and Stack to track inventory, pending orders, and completed orders, respectively.

Key Methods:

- **addOrUpdateStock(String item, int quantity)**:
- **Purpose**: Adds new stock to an item or updates the existing stock if the item already exists in the inventory.
- **Functionality**: If the item is already in the inventory, it updates the quantity. Otherwise, it adds the item to the inventory with the specified quantity.

- **placeOrder(String item, String quantityStr)**:
- **Purpose**: Places an order for a specified item and quantity.
- **Functionality**: The method checks if the item is available in sufficient quantity. If available, it processes the order and reduces the stock. If not, the order is added to a queue for future processing when the stock becomes available.

- **processPendingOrders()**:
- **Purpose**: Processes any orders that were placed when the item was out of stock.
- **Functionality**: It iterates through the queue of pending orders and attempts to fulfill them. If the stock is available, the order is processed, and the item's stock is reduced. If the stock is still insufficient, the order remains in the queue for future processing.

- **generateReports()**:
- **Purpose**: Generates a report showing the current inventory and the list of pending orders.
- **Functionality**: This method prints out the current stock of each item in the inventory and displays any pending orders.

- **deleteItem(String item)**:
- **Purpose**: Removes an item from the inventory.
- **Functionality**: The method checks if the item exists in the inventory. If found, it deletes the item; otherwise, it prints an error message indicating the item is not found.

- **showCompletedOrders()**:
- **Purpose**: Displays the history of completed orders.
- **Functionality**: It retrieves and displays the orders that have been successfully processed, which are stored in a stack for easy access in reverse order (most recent orders first).

-Class: Main

The Main class serves as the entry point for the Inventory Management System, offering a simple menu-driven console interface that enables users to manage inventory and perform various related tasks.

Key Features of the Code:

1. **Menu:**
    - The user is presented with several options to choose from:
        1. Add/Update Stock: Adds or updates the stock of a specified item.
        2. Place Order: Places an order for a specific item with a defined quantity.
        3. Process Pending Orders: Processes orders that are pending due to insufficient stock.
        4. Generate Report: Displays a report detailing the current inventory and pending orders.

5. History of Completed Orders: Shows a history of orders that have been fulfilled.
6. Delete Item: Removes an item from the inventory.
7. Exit: Exits the application.

2. Input Validation:

- Ensures that item names are neither empty nor contain numbers, using the containsNumber() helper method to validate item names.
- Exception handling is employed to catch invalid inputs, such as non-numeric values for quantities.

3. Case Insensitivity:

- Item names are converted to lowercase before any operation (add or update stock, place order, or delete item), ensuring the program handles case insensitivity.

4. Flow:

- A do-while loop keeps the application running, allowing users to perform various operations until they choose to exit (option 7).
- After each operation, the menu reappears, giving the user the ability to select another action.

##Github Repository

[mohamad0ahmad/Inventoy-Managment-Sys: console project on data structures]