INVENTORY ALERT MODULE
STACK-BASED ALERT HANDLING SYSTEM

Java Implementation for Intelligent Inventory Management

Uses Stack (LIFO) | No Arrays Used

PRESENTATION

Presented by

Rishwana Sirajutheen

PROBLEMS

Problem Statement

- Inventory systems generate alerts when stock levels change
- Repeated alerts create confusion for managers
- Only the latest alert reflects the current stock condition

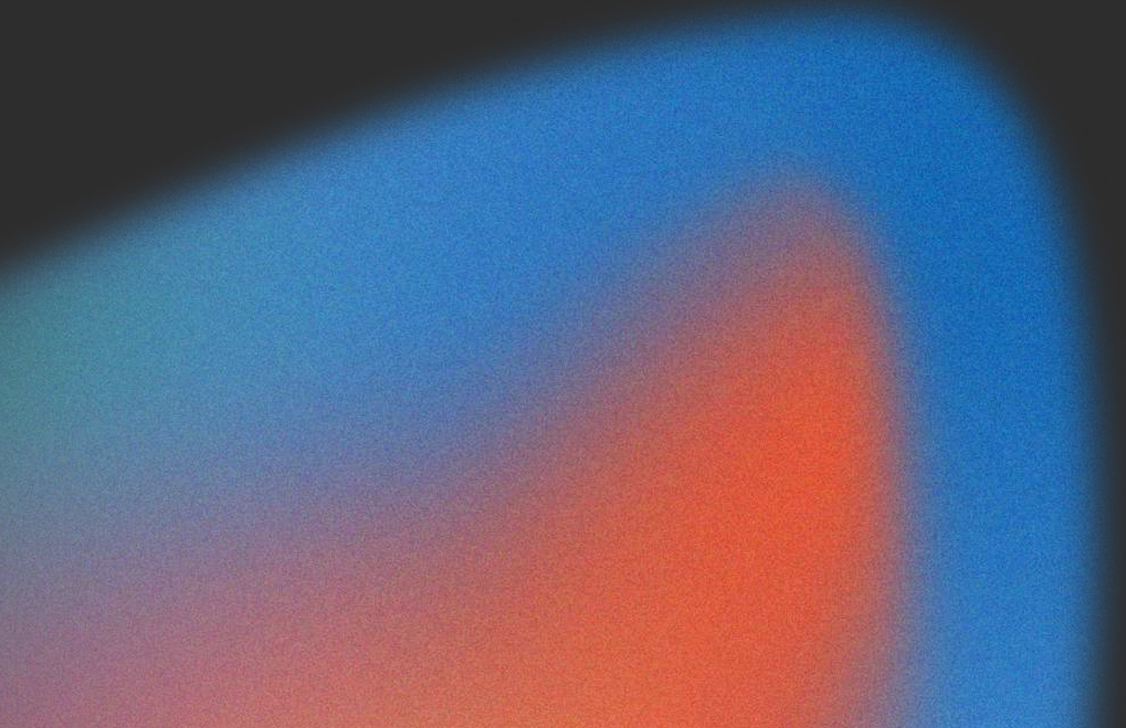
Requirement

- Store alerts using Stack
- Display the most recent alert
- Do not remove alert after displaying

WHY STACK?

Why Stack Data Structure?

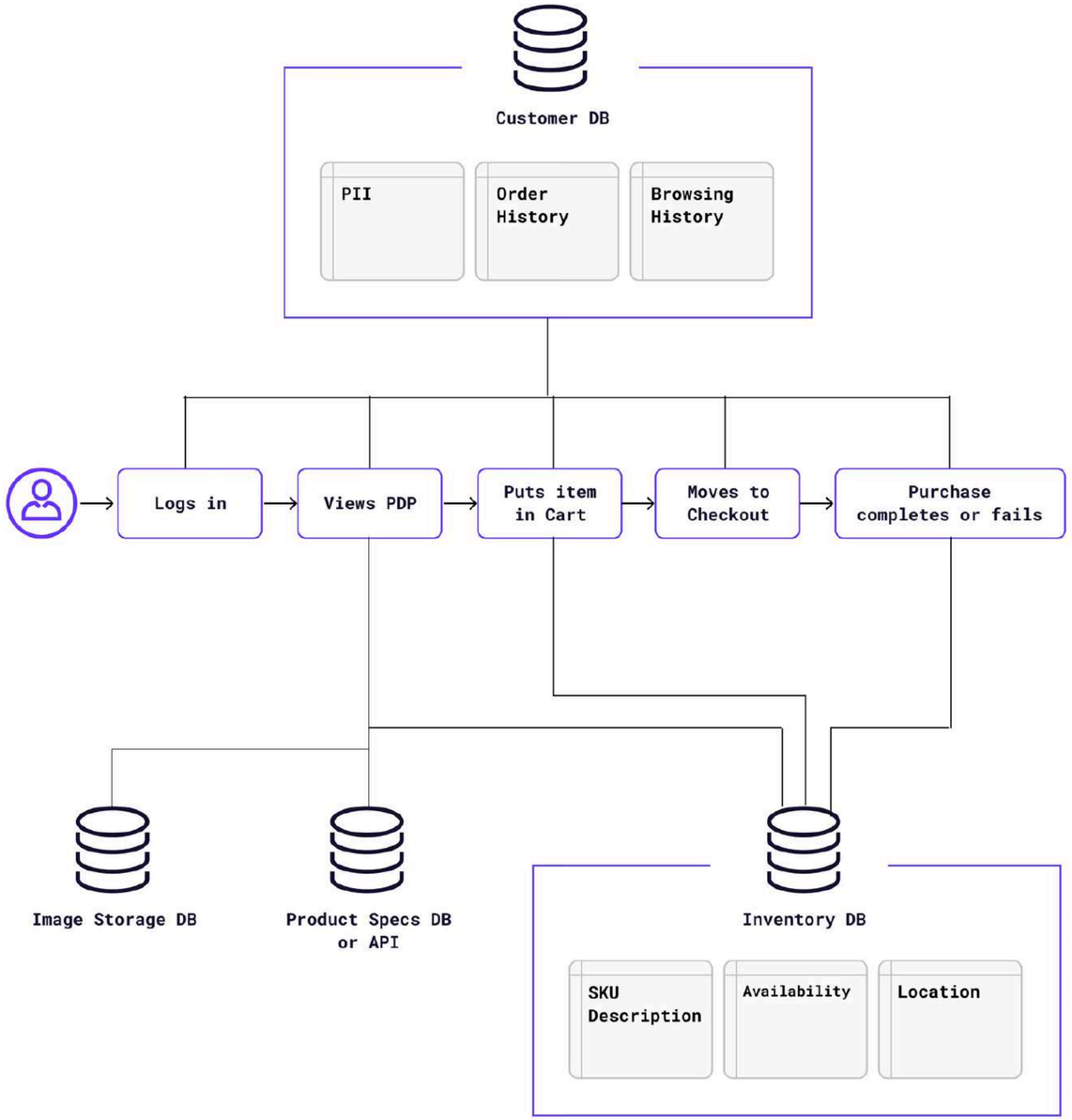
- Stack follows LIFO (Last In First Out)
- Latest alert must be shown first
- Efficient alert priority handling



Operations Used

- push() → Add new alert
- peek() → View latest alert
- isEmpty() → Safety check

SYSTEM ARCHITECTURE OVERVIEW



Architecture Flow

Inventory Update

→ Alert Generated

→ Stored in Stack

→ Latest Alert Displayed on Dashboard

CLASS DESIGN OVERVIEW

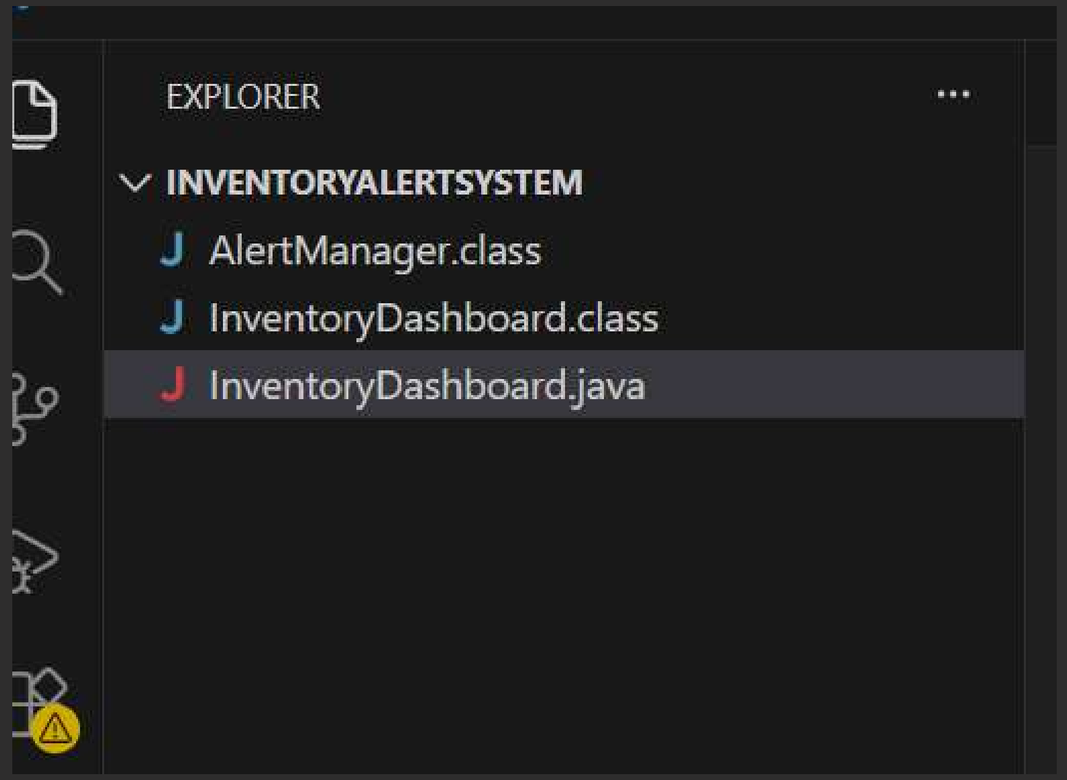
Classes Used

AlertManager

- Manages alert stack
- Handles push and peek operations

InventoryDashboard

- Acts as dashboard
- Displays latest aler



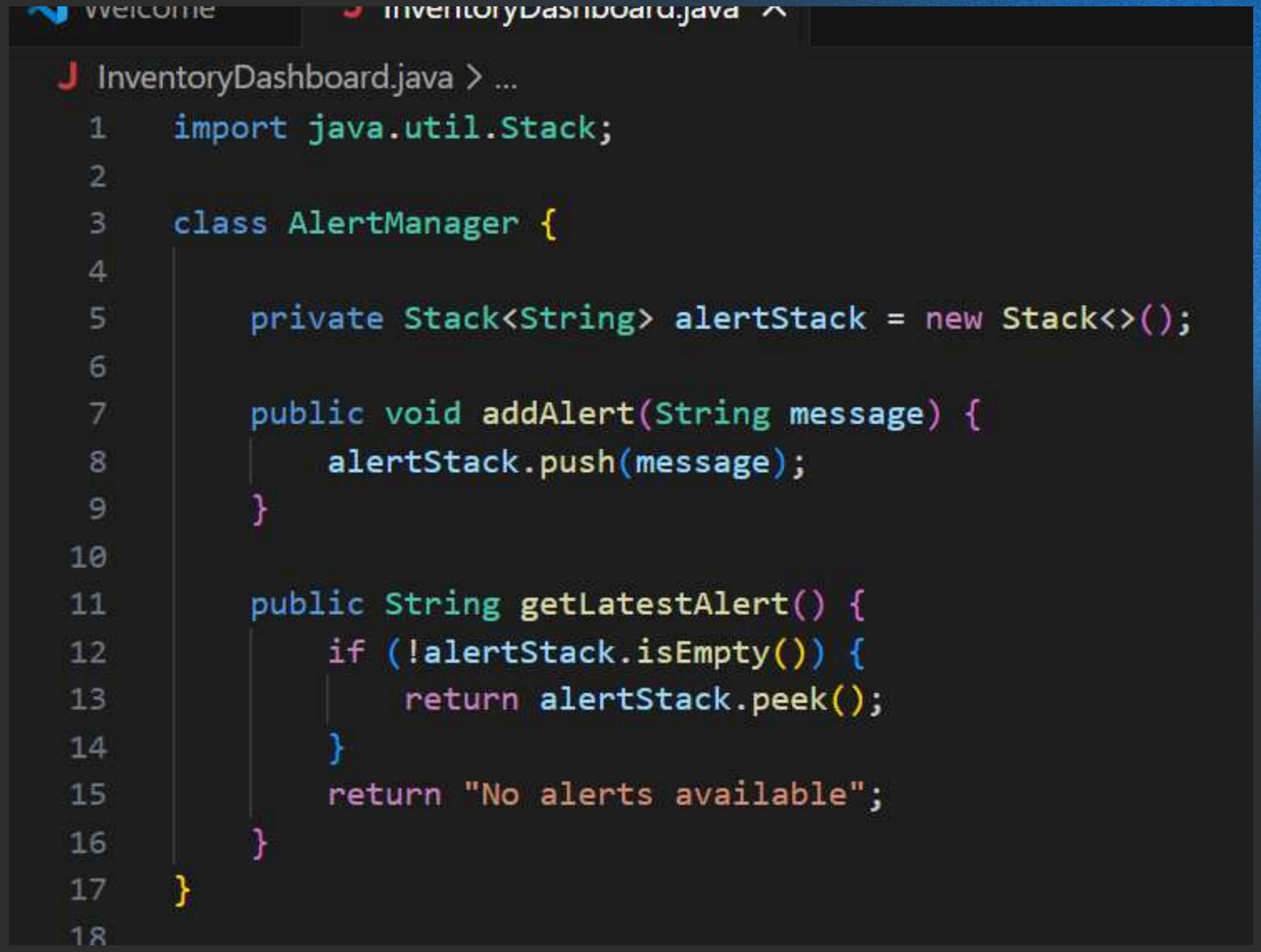
ALERTMANAGER CLASS (EXPLANATION)

AlertManager Responsibilities

- Stores alert messages in Stack
- Adds new alerts
- Returns latest alert without removing it

Key Code Logic

- Stack of String
- Uses peek() to fetch recent alert
- Prevents empty stack access



import java.util.Stack;

- Imports the Stack class from Java's utility package.
- Stack follows LIFO (Last In, First Out).
- Used to store inventory alert messages.

◆ class AlertManager {

- Defines a class named AlertManager.
- This class is responsible for handling alert operations.
- Keeps alert logic separate from the dashboard.

private Stack<String> alertStack = new Stack<>();

- Declares a Stack of String type.
- Each element in the stack is an alert message.
- private ensures data hiding (cannot be accessed directly outside this class).

public void addAlert(String message) {

- Method to add a new alert to the stack.
- Accepts an alert message as input.

alertStack.push(message);

- Pushes the alert message onto the top of the stack.
- Latest alert becomes the highest priority alert.

}

- Ends the addAlert() method.

public String getLatestAlert() {

- Method to retrieve the most recent alert.
- Returns a String value.

if (!alertStack.isEmpty()) {

- Checks whether the stack has any alerts.
- Prevents runtime error when accessing an empty stack.

return alertStack.peek();

- peek() returns the top element of the stack.
- The alert is not removed from the stack.
- Ensures alert history is preserved.

}

- Ends the if block.

return "No alerts available";

- Executed when the stack is empty.
- Prevents returning null.

}

- Ends the getLatestAlert() method.

}

- Ends the AlertManager class.

INVENTORYDASHBOARD CLASS (EXPLANATION)

Dashboard Responsibilities

- **Creates AlertManager object**
- **Adds inventory alerts**
- **Displays latest alert**

Execution Flow

1. **Program starts in main()**
2. **Alerts are added**
3. **Latest alert is retrieved**
4. **Output shown to user**

```
public class InventoryDashboard {

    Run | Debug
    public static void main(String[] args) {

        AlertManager alertManager = new AlertManager();

        alertManager.addAlert(message: "LOW STOCK: Item A");
        alertManager.addAlert(message: "OUT OF STOCK: Item B");

        System.out.println(x: "INVENTORY DASHBOARD");
        System.out.println(x: "Latest Alert:");
        System.out.println(alertManager.getLatestAlert());
    }
}
```

◆ public class InventoryDashboard {

- Defines the main dashboard class.
- Contains the main method.
- Acts as the user interface layer.

public static void main(String[] args) {

- Entry point of the Java program.
- Execution starts from this method.

AlertManager alertManager = new AlertManager();

- Creates an object of AlertManager.
- Allows the dashboard to add and retrieve alerts.

alertManager.addAlert("LOW STOCK: Item A");

- Adds a low-stock alert for Item A.
- Stored at the top of the stack.

alertManager.addAlert("OUT OF STOCK: Item B");

- Adds another alert.
- This becomes the latest alert due to LIFO behavior.

System.out.println("INVENTORY DASHBOARD");

- Prints the dashboard title.

System.out.println("Latest Alert:");

- Prints a label indicating the latest alert.

System.out.println(alertManager.getLatestAlert());

- Fetches the most recent alert using peek().
- Displays it on the dashboard.

}

- Ends the main() method.

}

- Ends the InventoryDashboard class.

SAMPLE OUTPUT

```
[Running] cd "d:\InventoryAlertSystem\" && javac InventoryDashboard.java && java InventoryDashboard
INVENTORY DASHBOARD
Latest Alert:
OUT OF STOCK: Item B

[Done] exited with code=0 in 1.198 seconds
```

Observation

- Last alert added is displayed
- Confirms LIFO behavior

ADVANTAGES OF THIS DESIGN

Advantages

- No arrays used
- Simple and efficient
- Latest alert always prioritized
- Easily extendable to web or cloud dashboard

CONCLUSION

- Stack is ideal for alert management
- peek() helps display latest alert safely
- System improves decision-making efficiency
- Suitable for real-time inventory dashboards

THANKS