Objective
Develop a real-time order notification system for a modern e-commerce platform. The system should alert various users (customers, sellers, logistics etc) about significant events in an order's lifecycle. The system must be designed in an extensible manner so that more users can be easily accommodated in the future if needed.
Requirements
1. Handling Event: Implement a system that handles three primary order events types
• Order Placed: Triggered when a customer successfully creates an order.
• Order Shipped: Triggered when the seller ships the package.
• Order Delivered: Triggered when the logistics partner confirms delivery.
2. Messaging Stakeholders : The system must notify different types of subscribers, each interested in specific events:
• Customer: Should be notified of all three events (Order Placed, Order Shipped, Order Delivered).
• Seller: Should only be notified when an order is placed.
• Logistics (Delivery Partner): Should only be notified when an order is shipped to manage the delivery.
• Only stakeholders linked to a particular order should be notified

3. Notification Channels: The system must support multiple channels for sending notifications. The choice of channel should be configurable. For example, it could be possible that the customer is receiving only mails for order getting placed. Types of notification channels could be
• Email Notification
• SMS Notification
• App Push Notification
Functional Requirements
Subscribe to Event:
Allow a stakeholder (e.g. a Customer) to subscribe to notifications for a specific event type with preferred channels of choice (e.g an Email).
Notifications should only be sent to Stakeholders linked with a particular order.
• Manage Subscriptions : Stakeholders should be able to
Unsubscribe from Event: Allow a stakeholder to unsubscribe from notifications for a particular event type.
Once any stakeholder unsubscribes from a particular event type, they will not receive any notification for any order for that particular event type.
Add / Remove Channels: A stakeholder should be able to manage their preferred channels for receiving notifications i.e add or remove channels as per choice. (It can be assumed that by default all the channels would be enabled by default for any user)

Bonus Requirements:
• Event replay : Implement a feature to replay the notification events for a particular order Id, event type and stakeholder.
When we replay a certain event, the stakeholder will receive the message on all the subscribed channels again for that particular order. For example if I can replay an Order delivery message

for the Order - 001 for the Customer who had subscription on Email and Phone, The tagged customer with that particular order will receive the message on Email and Phone.
The preferred channels for stakeholders during replay should be the latest preference on each subscription. For example, if customer A had Email and App push notification as preferred channels for Order O while getting notifications initially and later changed it to SMS only, the replay would only include SMS notifications.
• Concurrency: Handle cases related to user concurrency where multiple stakeholders could be trying to add / remove subscriptions
Asynchronous: The notifications should be sent in a non-blocking manner for the order system.
Implementation
Notes / FAQ
• Code should be Demoable : There should not be any compilation issues with the code, one can demonstrate the functionality either by using a
driver / main class, by writing extensive unit test cases or using an API, choice is up to the candidate.
Language: Java is the mandatory language to code.
• Requirements: Focus on the core functionality first before attempting bonus features.
Storage: Use in-memory data structures to store objects (e.g., a map of event types to a list of subscribers). Do not use any integration with databases like SQL or Mongo.
Framework / Library : Usage of web frameworks (e.g. Springboot / Dropwizard) and public Java libraries are allowed.
Example Scenario / Explanations
• Order Event: Exact design / detailing around how a particular order event which is received would look like is left up to the candidate to decide, however it should have the following characteristics
Event Type (Order Placed / Order Shipped /
Order Delivered)
Order (Id)
Customer (Id)
Seller (Id)
Delivery Partner Assigned (Id)
Notification Message

An example is given here. Candidate can use
other format if they prefer so
[Timestamp][NOTIFIACTION][Order ID][Event Ty
‹ Message ›
Timestamp can be in any format like Epoch (1758150296659) or human readable like 2025-09-18T00:00:00 and should be the timestamp of when the event was sent.
For example,
[18-09-202500:00:00] [NOTIFIACTION] [ORDER-001] [OR package with orderId ORDER-001 has been placed"
• Setup:

• A Customer (ID: CUSTOMER- 1) is subscribed to the notification system with SMS as the preferred channel.
• A Seller (ID: SELLER-1) is subscribed to the notification system with Email as preferred channel.
Delivery partner (ID : DELIVERY -AGENT - 1) is also subscribed.
with App push as preferred channel
• Event: Order Placed
The system triggers an Order Placed event for Order ID ORDER-001.
V
[18-09-202500:00:00][NOTIFIACTION] [ORDER -001] [OR
package with ID - ORDER-001 has been placed"
[18-09-2025T00:00:01] [NOTIFIACTION] [ORDER-001] [OF order with ID - ORDER-001 has been placed from

[18-09-202500:00:021 [NOTIFIACTION] [ORDER-0011 [OF
order with ID - ORDER-001 is ready for pickup"
6. Event: Order Delivered
[18-09-202500:00:01] [NOTIFIACTION] [ORDER-001] [OP
"Your package with ID - ORDER-001 has been delive
7. Subscription Change
?.
Unsubscribe from Event Type: In case of change in subscription for an event type, there should be a System message with the following attributes:
• Timestamp (Current)
• Subscription
• Subscription ID
• Message describing the change in Event type
For example,
V
[18-09-2025T00:00:01][SUBSCRIPTION][SUB-001] - "C from event ORDER_DELIVERED"

8. Replay Message: In case of message replay, there should be a System message describing for which order, event type and stakeholder is the message being replayed followed by the notifications. The following attributes should be present:
• System Message:
Timestamp (Current)
Message describing what was the replay executed for i.e Order Id, Event Type and Stakeholder (Customer/ Seller / Delivery Partner)
should be included here
• Replayed Notification :
Timestamp : (Current)
• Order Id
Event Type
Stakeholder Id (Customer ID / Seller

ID / Delivery Partner ID)
• Channel (Email / SMS / App)
Message (Replayed Notification
Message)
For example, if we replay the Order Delivered messenger on Order ORDER-001 for the customer, he gets a SMS with the same message. Please note that the timestamp here is the latest timestamp not when the message was initially sent.

```
[18-09-2025T06:30:00][SYSTEM] - "Replaying ORDER ORDER-001 for Customer"
[18-09-2025700:00:01] [NOTIFIACTION] [ORDER-001] [OP
"Your package with ID - ORDER-001 has been delive
```

Evaluation Criteria [IMPORTANT]

Evaluation Criteria [IMPORTANT]
• Correctness: The solution should correctly implement the required functionalities as given in the problem statement.
• Completeness: The solution should cover all aspects of the functional requirements(must haves)
, including sending notifications as well subscription management i.e subscribing / unsubscribing and adding / removing channel preferences (Bonus is a good to have)
Validation, Corner Cases & Exception Handling :
Code should contain appropriate check over passed input, throw exceptions wherever applicable and also handle different edge cases.
• Code Quality: The code should be clean, well-organized, and easy to understand. Follow 00 principles.
• Code Extensibility: Code should be modular & extensible
Efficiency: The solution should handle operations efficiently, considering edge cases and