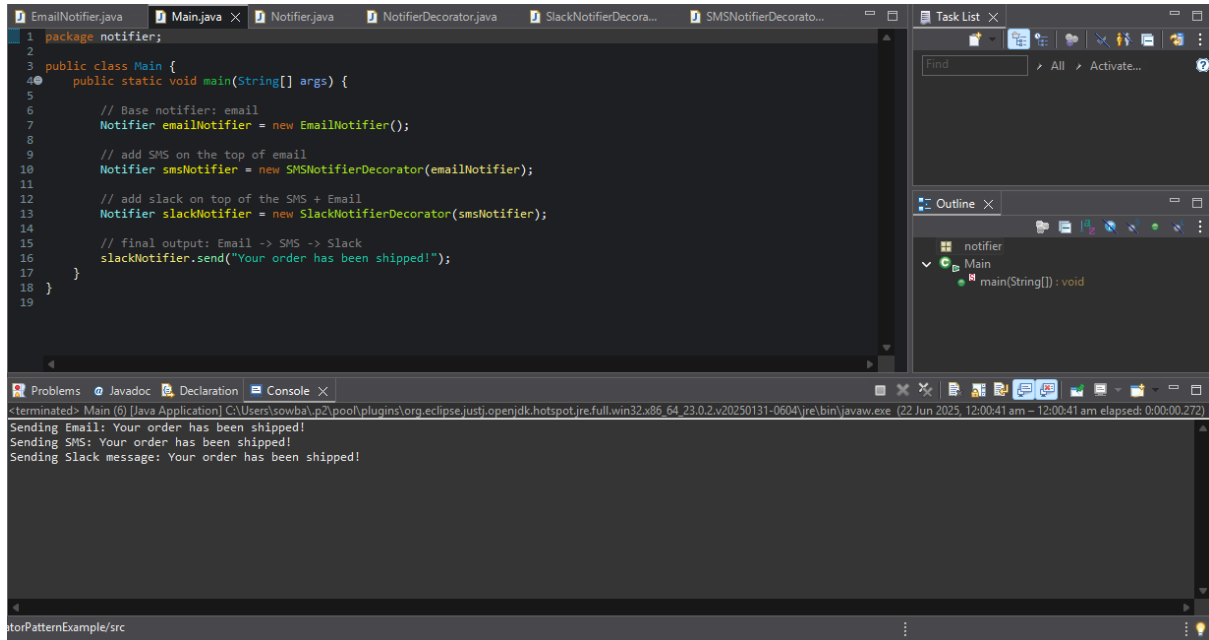


Exercise 5: Implementing the Decorator Pattern

OUTPUT:



```
1 package notifier;
2
3 public class Main {
4     public static void main(String[] args) {
5
6         // Base notifier: email
7         Notifier emailNotifier = new EmailNotifier();
8
9         // add SMS on the top of email
10        Notifier smsNotifier = new SMSNotifierDecorator(emailNotifier);
11
12        // add slack on top of the SMS + Email
13        Notifier slackNotifier = new SlackNotifierDecorator(smsNotifier);
14
15        // final output: Email -> SMS -> Slack
16        slackNotifier.send("Your order has been shipped!");
17    }
18 }
19
```

Sending Email: Your order has been shipped!
Sending SMS: Your order has been shipped!
Sending Slack message: Your order has been shipped!

The Decorator Pattern enables dynamic addition of notification channels (like SMS, Slack) without modifying existing code. It enhances flexibility and scalability by allowing features to be added at runtime.