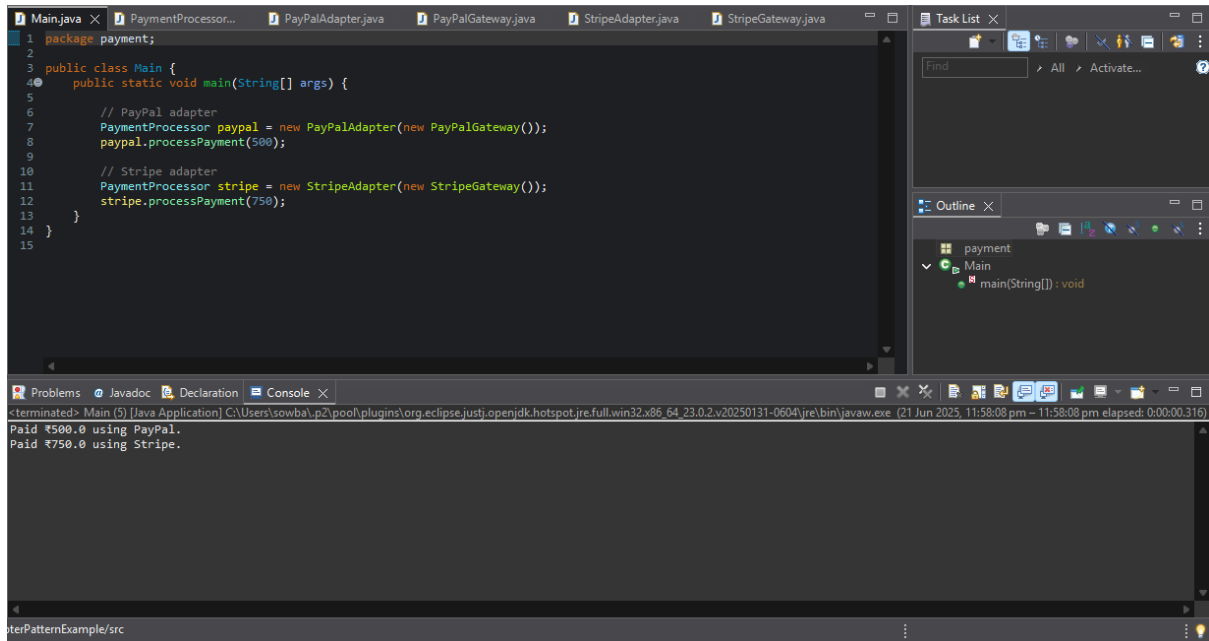


Exercise 4: Implementing the Adapter Pattern

OUTPUT:



The screenshot displays an IDE with several Java files open: Main.java, PaymentProcessor.java, PayPalAdapter.java, PayPalGateway.java, StripeAdapter.java, and StripeGateway.java. The Main.java file is the active editor, showing the following code:

```
1 package payment;
2
3 public class Main {
4     public static void main(String[] args) {
5
6         // PayPal adapter
7         PaymentProcessor paypal = new PayPalAdapter(new PayPalGateway());
8         paypal.processPayment(500);
9
10        // Stripe adapter
11        PaymentProcessor stripe = new StripeAdapter(new StripeGateway());
12        stripe.processPayment(750);
13    }
14 }
15
```

The Outline view on the right shows the package structure: payment > Main > main(String[]): void.

The Console view at the bottom shows the execution output:

```
<terminated> Main (5) [Java Application] C:\Users\sowba\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_23.0.2.v20250131-0604\jre\bin\javaw.exe (21 Jun 2025, 11:58:08 pm - 11:58:08 pm elapsed: 0:00:00.316)
Paid ₹500.0 using PayPal.
Paid ₹750.0 using Stripe.
```

The Adapter Pattern allows integration of multiple third-party payment gateways with different interfaces by using adapter classes. It promotes flexibility, code reusability, and seamless compatibility without changing existing code.