**Exercise 2: Implementing the Factory Method Pattern**

**Scenario:**

I were developed a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Used the Factory Method Pattern to achieve this.

**Solution:**

**Notification.java**

**public interface Notification {**

**void notifyUser();**

**}**

**Email.java**

**public class Email implements Notification {**

**@Override**

**public void notifyUser() {**

**System.out.println("Sending an Email Notification");**

**}**

**}**

**SMS.java**

**public class SMS implements Notification {**

**@Override**

**public void notifyUser() {**

**System.out.println("Sending an SMS Notification");**

**}**

**}**

**Push.java**

**public class Push implements Notification {**

**@Override**

**public void notifyUser() {**

**System.out.println("Sending a Push Notification");**

**}**

**}**

**NotificationFactory.java**

**public class NotificationFactory {**

**public Notification createNotification(String type) {**

**if (type == null || type.isEmpty()) {**

**return null;**

**}**

**if (type.equalsIgnoreCase("EMAIL")) {**

**return new Email();**

**} else if (type.equalsIgnoreCase("SMS")) {**

**return new SMS();**

**} else if (type.equalsIgnoreCase("PUSH")) {**

**return new Push();**

**}**

**return null;**

**}**

**}**

**Main.java**

**public class Main {**

**public static void main(String[] args) {**

**NotificationFactory factory = new NotificationFactory();**

**Notification n1 = factory.createNotification("EMAIL");**

**n1.notifyUser();**

**Notification n2 = factory.createNotification("SMS");**

**n2.notifyUser();**

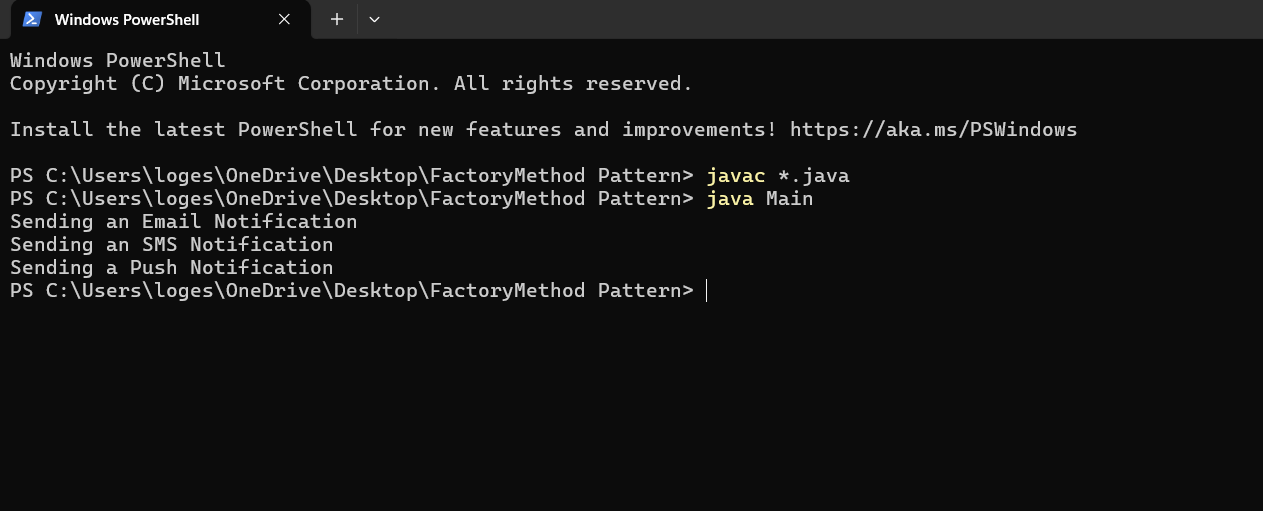
**Notification n3 = factory.createNotification("PUSH");**

**n3.notifyUser();**

**}**

**}**

**Output ScreenShot:**

****