Exercice 1 : Adaptateur de Système de Paiement

Contexte : Vous travaillez sur un site e-commerce qui intègre diverses méthodes de paiement. Le site a déjà intégré un système de paiement externe appelé `OldPaymentGateway`. Cependant, un nouveau système appelé `NewPaymentProcessor` a été choisi pour remplacer l'ancien en raison de ses fonctionnalités améliorées. L'interface du nouveau système est différente de celle de l'ancien système.

Objectif : Créer un adaptateur qui permettra au site d'utiliser `NewPaymentProcessor` avec le même code client qui utilisait `OldPaymentGateway`.

- `OldPaymentGateway` a une méthode `makePayment(String accountNumber, double amount)`.
- `NewPaymentProcessor` utilise deux méthodes : `authenticate(String apiKey)` et `sendPayment(double amount)`.
- Créer une classe `PaymentAdapter` qui implémente l'interface `OldPaymentGateway` et utilise une instance de `NewPaymentProcessor` pour exécuter les paiements.

Exercice 2 : Adaptateur pour Système de Notification

Contexte : Une application possède un module de notification qui utilise actuellement un service de messagerie pour envoyer des alertes via SMS. Avec l'évolution des besoins, un nouveau service de notification par email, `EmailService`, a été développé.

Objectif : Adapter le `EmailService` pour qu'il puisse être utilisé à la place du service de messagerie SMS sans modifier le code existant du module de notification.

- Le service de messagerie SMS existant (`SmsService`) a une méthode `sendSms(String number, String message)`.
- Le nouveau service (`EmailService`) a une méthode `sendEmail(String email, String subject, String body)`.
- Créer une classe `NotificationAdapter` qui implémente l'interface de `SmsService` mais envoie les notifications via `EmailService`. Assumez que chaque numéro de téléphone a un email correspondant stocké qui peut être récupéré.