Quiz# 2

Q1: Your company has been using a notification app to send notifications to users via email. The EmailNotificationService is responsible for sending emails, and it always sends a standard email format: "Dear [Name], you have a new message." However, the company now wants to introduce an SMSNotificationService to send notifications via SMS in a shorter format: "[Name], you have a new SMS!" The company also realized that using only email for notifications created high coupling between the NotificationApp and the EmailNotificationService. As a good software engineering practice, the company decided to introduce an abstraction for notifications and reduce the coupling by using dependency injection. Update the following code to introduce these changes.

```
public class EmailNotificationService {

    public String notifyUser(String name) {

        return "Dear " + name + ", you have a new message.";

    }

}
```

```
// Abstraction for Notification Service

public interface NotificationService { 3

    String notifyUser(String name);

}


// Formal email notification service

public class EmailNotificationService implements NotificationService { 2

    @Override

    public String notifyUser(String name) {

        return "Dear " + name + ", you have a new message.";

    }

}


// Informal SMS notification service

public class SMSNotificationService implements NotificationService { 3
```

```
public class NotificationApp {
    private final EmailNotificationService emailNotificationService;


 public NotificationApp() {
        this.emailNotificationService = new EmailNotificationService();
    }

    public void notifyUser(String name) {
        String notification = emailNotificationService.notifyUser(name);
        System.out.println(notification);
    }

    public static void main(String[] args) {
        NotificationApp notificationApp = new NotificationApp();
        notificationApp.notifyUser("Sarah");
```

```
    @Override
    public String notifyUser(String name) {
        return name + ", you have a new SMS!";
    }
}

// Notification App with Dependency Injection
public class NotificationApp {
    private final NotificationService notificationService; 1

    // Constructor Dependency Injection
    public NotificationApp(NotificationService notificationService) { 3
        this.notificationService = notificationService;
    }

    public void notifyUser(String name) {
        String notification = notificationService.notifyUser(name);
        System.out.println(notification);
    }

    public static void main(String[] args) {
        // Example 1: Using EmailNotificationService
        NotificationApp emailApp = new NotificationApp(new
EmailNotificationService()); 1
```

```
        notificationApp.notifyUser("Ali");  // TODO: Should receive an SMS
notification.
    }
}
```

```
        emailApp.notifyUser("Sarah"); 0.5

        // Example 2: Using SMSNotificationService
        NotificationApp smsApp = new NotificationApp(new
SMSNotificationService()); 1
        smsApp.notifyUser("Ali"); 0.5
    }
}
```