**Title: SOLID Principles Quiz - With Answers Highlighted**

**SOLID Principles Quiz**

**1. What does the 'S' in SOLID stand for?**

❌ A) Single Structure Principle  
✅ **B) Single Responsibility Principle**  
❌ C) Software Reliability Principle  
❌ D) Systematic Refactoring Principle

**2. Which of the following violates the Single Responsibility Principle (SRP)?**

✅ **A) A class handling both user authentication and email notifications**  
❌ B) A class that manages only database operations  
❌ C) A class with multiple helper methods for different modules  
❌ D) A class following the MVC pattern

**3. The Open/Closed Principle (OCP) suggests that:**

❌ A) Classes should be modified frequently to accommodate new features  
❌ B) Classes should be open for modification and closed for extension  
✅ **C) Classes should be open for extension but closed for modification**  
❌ D) Classes should not be extended in any case

**4. How do we best implement the Open/Closed Principle in Laravel?**

❌ A) Using if conditions to handle multiple scenarios in a single class  
❌ B) Creating a single large service class for all operations  
✅ **C) Using interfaces and extending functionality through separate implementations**  
❌ D) Embedding logic directly in controllers

**5. What is a key requirement for Liskov Substitution Principle (LSP)?**

❌ A) A subclass must implement all methods of its parent class  
✅ **B) A subclass should be able to replace the parent class without altering expected behavior**  
❌ C) Subclasses should extend multiple parent classes  
❌ D) Subclasses should modify the behavior of inherited methods

**6. Which of the following is an example of violating Liskov Substitution Principle?**

✅ **A) A Bird class with a fly() method, and a Penguin subclass overriding fly() with an exception**  
❌ B) A Shape class with calculateArea(), and Circle and Rectangle subclasses implementing it correctly  
❌ C) A NotificationService class with a send() method, and EmailNotification and SMSNotification implementing it  
❌ D) Using interface segregation to define smaller interfaces

**7. Interface Segregation Principle (ISP) helps avoid:**

❌ A) Too many interfaces in a system  
✅ **B) Classes implementing methods they do not use**  
❌ C) Subclasses modifying inherited behaviors  
❌ D) Redundant database queries

**8. Which scenario follows the Interface Segregation Principle?**

❌ A) Defining a UserActionsInterface with login(), register(), and deleteUser() for all users  
✅ **B) Splitting user responsibilities into LoginInterface, RegisterInterface, and AdminActionsInterface**  
❌ C) Having a single interface for all system actions  
❌ D) Using an abstract class instead of interfaces

**9. What does the Dependency Inversion Principle (DIP) advocate?**

❌ A) High-level modules should depend on low-level modules  
✅ **B) High-level modules should depend on abstractions, not concrete implementations**  
❌ C) Concrete classes should be used instead of interfaces  
❌ D) Dependency Injection should be avoided

**10. In Laravel, how do we best follow Dependency Inversion Principle?**

❌ A) Instantiate services directly inside controllers  
✅ **B) Bind interfaces to concrete implementations using Laravel’s Service Container**  
❌ C) Use static classes instead of dependency injection  
❌ D) Store dependencies in global variables

🎯 **Congratulations! This document serves as a structured quiz with correct answers highlighted to reinforce your understanding of SOLID principles. 🚀**