**Notes**

SOLID Principles

SOLID is an acronym representing five key principles of object-oriented design:

1. **Single Responsibility Principle (SRP)**
   * A class should have only one reason to change.
   * Promotes focused and maintainable classes.
2. **Open/Closed Principle (OCP)**
   * Software entities should be open for extension but closed for modification.
   * Encourage adding new features without changing existing code.
3. **Liskov Substitution Principle (LSP)**
   * Objects of a derived class should be substitutable for objects of the base class without affecting the correctness of the program.
   * Ensures that inheritance relationships are used appropriately.
4. **Interface Segregation Principle (ISP)**
   * Clients should not be forced to depend on interfaces they do not use.
   * Promotes smaller, more focused interfaces.
5. **Dependency Inversion Principle (DIP)**
   * High-level modules should not depend on low-level modules. Both should depend on abstractions.
   * Abstractions should not depend on details. Details should depend on abstractions.
   * Encourages loose coupling and flexibility.

Benefits of SOLID Principles

* **Maintainability:** Makes your code easier to understand, modify, and extend.
* **Testability:** Promotes writing unit tests by encouraging loose coupling and dependency injection.
* **Flexibility:** Makes your code adaptable to changes in requirements.
* **Reusability:** Encourages the creation of reusable components.

**Interview Tips**

* **Understanding:** Be able to explain each principle clearly and concisely.
* **Examples:** Provide real-world or code examples that demonstrate how to apply each principle.
* **Benefits:** Articulate the advantages of adhering to SOLID principles.
* **Trade-offs:** Acknowledge that there might be trade-offs and complexities in applying these principles in certain situations.
* **Practical Application:** Discuss how you have used or would use SOLID principles in your own projects.

**Example Code (Conceptual)**

1. // SRP (Single Responsibility Principle)
2. public class ProductService
3. {
4. // Handles product-related logic, like adding or retrieving products.
5. }

8. public class OrderService
9. {
10. // Handles order-related logic, like creating or processing orders.
11. }

14. // OCP (Open/Closed Principle)
15. public interface IPaymentProcessor
16. {
17. void ProcessPayment(PaymentDetails details);
18. }

21. public class CreditCardPaymentProcessor : IPaymentProcessor { /\* ... \*/ }
22. public class PayPalPaymentProcessor : IPaymentProcessor { /\* ... \*/ }
24. // LSP (Liskov Substitution Principle)
25. public class Rectangle
26. {
27. public virtual int Width { get; set; }
28. public virtual int Height { get; set; }
29. // ...
30. }

33. public class Square : Rectangle
34. // Violates LSP
35. {
36. public override int Width
37. {
38. get => base.Width;
39. set
40. {
41. base.Width = value;
42. base.Height = value; // Setting width also sets height
43. }
44. }
46. public override int Height
47. {
48. get => base.Height;
49. set
50. {
51. base.Height = value;
52. base.Width = value; // Setting height also sets width
53. }
54. }
55. }

58. // ISP (Interface Segregation Principle)
59. public interface IPrinter
60. {
61. void Print();
62. }
64. public interface IScanner
65. {
66. void Scan();
67. }
69. public class PrintScanMachine : IPrinter, IScanner { /\* ... \*/ }
71. // DIP (Dependency Inversion Principle)
72. public class OrderProcessor
73. {
74. private readonly IPaymentProcessor \_paymentProcessor;
76. public OrderProcessor(IPaymentProcessor paymentProcessor)
77. {
78. \_paymentProcessor = paymentProcessor;
80. }
82. // ...
83. }