Comprehensive Design Patterns - Special Practice Test

Multiple Choice Questions (30 questions)

1. **Which design pattern allows you to change the behavior of an object when its internal state changes?**
    - A) State Pattern
    - B) Observer Pattern
    - C) Command Pattern
    - D) Proxy Pattern
2. **In which pattern do you encapsulate a request as an object, allowing parameterization of clients with different requests?**
    - A) Strategy Pattern
    - B) Chain of Responsibility Pattern
    - C) Command Pattern
    - D) Mediator Pattern
3. **Which design pattern is used to create families of related or dependent objects without specifying their concrete classes?**
    - A) Abstract Factory Pattern
    - B) Factory Method Pattern
    - C) Builder Pattern
    - D) Singleton Pattern
4. **Which of the following patterns is most useful when you need to decouple an abstraction from its implementation?**
    - A) Proxy Pattern
    - B) Decorator Pattern
    - C) Bridge Pattern
    - D) Composite Pattern
5. **In the ______________*Composite Pattern*______________, how are individual objects and their compositions treated?**
    - A) They are treated as two separate entities with different interfaces.
    - B) They are treated uniformly through a common interface.
    - C) Only the composite objects are treated uniformly.
    - D) Each object must implement a different interface.
6. **Which design pattern allows an object to create new instances by copying an existing object, often used in scenarios where object creation is complex?**
    - A) Singleton Pattern

- B) Prototype Pattern
- C) Builder Pattern
- D) Factory Method Pattern

7. **Which of the following patterns is used to ensure that a class has only one instance and provides a global point of access to it?**
    - A) Factory Method Pattern
    - B) Prototype Pattern
    - C) Singleton Pattern
    - D) Strategy Pattern
8. **What is the purpose of the _______________*Adapter Pattern*_______________?**
    - A) To provide a simplified interface to a complex system
    - B) To allow incompatible interfaces to work together
    - C) To provide a dynamic mechanism for adding behavior to objects
    - D) To define a family of algorithms.
9. **Which design pattern defines a structure that can be extended without changing its code?**
    - A) Decorator Pattern
    - B) Strategy Pattern
    - C) Composite Pattern
    - D) Template Method Pattern
10. **In the _______________*Visitor Pattern*_______________, what role does the visitor object play?**
    - A) It creates new elements in the structure.
    - B) It performs operations on elements of the structure without changing the structure itself.
    - C) It defines a new way to create an object.
    - D) It stores the state of the visited objects.
11. **Which of the following patterns allows for a consistent interface for accessing elements of an aggregate object?**
    - A) Observer Pattern
    - B) Iterator Pattern
    - C) Chain of Responsibility Pattern
    - D) Proxy Pattern
12. **In the _______________*Facade Pattern*_______________, what does the facade class provide?**
    - A) A complex interface to a subsystem
    - B) A simplified interface to a set of complex subsystems
    - C) A mediator between client and subsystem
    - D) A way to manage dependencies between objects.

13. **In the ______________*Bridge Pattern*______________, what is decoupled to allow more flexibility?**
    - A) The object and its behavior
    - B) The abstraction and the implementation
    - C) The strategy and the context
    - D) The command and the invoker
14. **Which design pattern is used when an object needs to be created, but the exact type is unknown or varies?**
    - A) Abstract Factory Pattern
    - B) Singleton Pattern
    - C) Factory Method Pattern
    - D) Prototype Pattern
15. **In the ______________*Chain of Responsibility Pattern*______________, what determines the handler of a request?**
    - A) The type of request
    - B) The first handler in the chain
    - C) The priority of the handler
    - D) The context of the request
16. **Which design pattern allows the creation of complex objects step by step, and provides flexibility in their construction?**
    - A) Prototype Pattern
    - B) Factory Method Pattern
    - C) Builder Pattern
    - D) Singleton Pattern
17. **Which design pattern defines a family of algorithms and allows the client to choose between them dynamically?**
    - A) Command Pattern
    - B) Strategy Pattern
    - C) Observer Pattern
    - D) Memento Pattern
18. **In the ______________*State Pattern*______________, what allows an object to change its behavior when its internal state changes?**
    - A) A strategy object
    - B) A proxy object
    - C) A state object
    - D) A memento object
19. **What is the main purpose of the ______________*Decorator Pattern*______________?**

- A) To change the state of an object dynamically
- B) To add new functionality to an object without modifying its structure
- C) To create new instances of objects
- D) To control the creation of objects

20. **Which pattern is most effective for situations where an object needs to delegate requests to other objects while controlling access?**
    - A) Command Pattern
    - B) Proxy Pattern
    - C) State Pattern
    - D) Builder Pattern
21. **Which of the following patterns is used to manage different algorithms dynamically by selecting one at runtime?**
    - A) Observer Pattern
    - B) Strategy Pattern
    - C) Memento Pattern
    - D) Visitor Pattern
22. **In the _______________*Memento Pattern*_______________, what object is responsible for capturing and restoring the state of an originator object?**
    - A) Caretaker
    - B) Client
    - C) Memento
    - D) Originator
23. **Which design pattern provides an object with an interface for operations on a tree structure, allowing for recursive operations?**
    - A) Command Pattern
    - B) Composite Pattern
    - C) Proxy Pattern
    - D) State Pattern
24. **In the _______________*Prototype Pattern*_______________, how does the client create new objects?**
    - A) By modifying an existing object
    - B) By cloning an existing object
    - C) By calling a constructor
    - D) By using a factory method
25. **Which design pattern simplifies object creation by delegating the construction to specialized builder classes?**
    - A) Factory Method Pattern

- B) Singleton Pattern
- C) Builder Pattern
- D) Memento Pattern

26. **Which design pattern allows a system to handle requests in a chain, where each handler either processes the request or passes it to the next handler?**
    - A) Composite Pattern
    - B) Chain of Responsibility Pattern
    - C) State Pattern
    - D) Command Pattern
27. **In the _______________*Observer Pattern*_______________, when the state of the subject changes, it ________________ its observers.**
    - A) Creates
    - B) Registers
    - C) Notifies
    - D) Removes
28. **In the _______________*Strategy Pattern*_______________, the algorithm is encapsulated in a separate class, and the context uses this class for ________________.**
    - A) Selecting the algorithm
    - B) Creating the algorithm
    - C) Storing the algorithm
    - D) Modifying the algorithm
29. **In the _______________*State Pattern*_______________, the context object can switch between states by changing the ________________ object it uses.**
    - A) Strategy
    - B) Proxy
    - C) State
    - D) Memento
30. **Which pattern allows an object to alter its behavior based on its internal state and changes its behavior dynamically?**
    - A) Strategy Pattern
    - B) Command Pattern
    - C) State Pattern
    - D) Composite Pattern

Fill-in-the-Blanks (30 questions)

1. In the **Memento Pattern**, the ______________ is responsible for storing and retrieving mementos but does not modify the memento's state.
2. The ______________ **Pattern** allows you to create new objects by copying an existing one rather than creating them from scratch.
3. The ______________ **Pattern** is used to decouple the abstraction of an object from its implementation, allowing both to evolve independently.
4. The ______________ **Pattern** is used when you need to control access to an object and add behavior dynamically without modifying the original object's class.
5. The ______________ **Pattern** provides a simple interface to a complex subsystem, hiding the complexity from the client.
6. The ______________ **Pattern** involves creating a new object by copying an existing one, which serves as the prototype.
7. The ______________ **Pattern** defines a family of algorithms, encapsulates them, and allows the client to choose which algorithm to use at runtime.
8. In the ______________ **Pattern**, a central mediator facilitates communication between objects without them needing to refer to each other directly.
9. The ______________ **Pattern** allows you to add new functionality to an object without modifying its structure, often used to extend behavior dynamically.
10. The ______________ **Pattern** allows objects to be created step-by-step and provides the flexibility to construct a variety of objects with different representations.
11. The ______________ **Pattern** allows an object to change its behavior when its internal state changes, simulating state dependent behavior.
12. The ______________ **Pattern** encapsulates a request as an object, allowing parameterization of clients with different requests and queuing of requests.
13. The ______________ **Pattern** is used to represent part-whole hierarchies where both individual objects and composites of objects can be treated uniformly.
14. The ______________ **Pattern** provides a mechanism for undo/redo by storing the internal state of an object and restoring it when needed.
15. The ______________ **Pattern** helps in managing object creation, making it easier to manage families of related objects by providing an interface to create them.
16. The ______________ **Pattern** is used when there is a need to create complex objects, often involving multiple steps or configurations.
17. The ______________ **Pattern** treats individual objects and composite objects uniformly, making it easier to work with tree-like structures.
18. In the ______________ **Pattern**, objects are created based on the prototype of an existing object, and new objects can be created by cloning the prototype.
19. The ______________ **Pattern** allows you to change the behavior of an object at runtime by using different strategies encapsulated in separate classes.
20. The ______________ **Pattern** allows you to separate the creation of objects from the actual logic that uses the objects, making it easier to modify their creation process.

21. The ______________ **Pattern** allows clients to change the algorithm being used based on the current context without modifying the class itself.
22. The ______________ **Pattern** allows the delegation of responsibility for a request to a chain of objects, each handling the request in a specific way.
23. The ______________ **Pattern** decouples the interface from the implementation by providing a way for objects to interact via an abstract interface, reducing dependencies.
24. The ______________ **Pattern** provides a flexible way to structure classes that can perform different operations but share a common interface.
25. The ______________ **Pattern** allows a system to change its behavior based on the context of the request, handling requests in a chain.
26. The ______________ **Pattern** defines the structure of an algorithm and allows subclasses to redefine certain steps without modifying the structure of the algorithm.
27. The ______________ **Pattern** allows for dynamic object creation based on different configurations, simplifying the construction process.
28. The ______________ **Pattern** is used to allow an object to alter its behavior dynamically when its state changes, without modifying its class.
29. The ______________ **Pattern** provides a way for an object to be created by copying an existing object rather than constructing it from scratch.
30. In the ______________ **Pattern**, the context object maintains a reference to the state it is in, delegating behavior to the state object.

Design Patterns - Case Study Analysis (10 questions)

Case Study 1: Payment Processing System

Scenario:
You are building an online payment processing system where users can choose from multiple payment methods (e.g., credit card, PayPal, cryptocurrency, etc.). The system needs to support adding new payment methods in the future without affecting the existing codebase.

Question:
Which design pattern would you recommend for this scenario? Explain why it fits, and how it would be implemented.

Case Study 2: Logging System

Scenario:
Your system needs to generate different types of logs based on the severity of events (e.g., info, warning, error). The logging system should allow adding new types of logging formats (e.g., JSON, XML, plain text) and handle the logging logic in different ways (e.g., log to file, log to database, or log to console).

Question:
Which design pattern would be most suitable for this logging system? Explain how it works in this context, and provide an outline of its components.

Case Study 3: Restaurant Ordering System

Scenario:
You are designing a restaurant ordering system where customers can choose from various menu items. Some menu items are customizable, such as pizzas (choose toppings) and burgers (choose size and extra toppings). The system needs to calculate the total price based on customer selections.

Question:
Which design pattern would help handle the creation and customization of these menu items? Provide a detailed explanation and steps for implementation.

Case Study 4: User Profile Management

Scenario:
Your application has a user profile management system where users can select different roles (Admin, User, Moderator, etc.). Each role has specific permissions, and the system needs to manage changes in permissions dynamically without altering the user class.

Question:
Which design pattern would be best for handling the dynamic roles and permissions system? Describe how the pattern would be used to manage different permissions for users in this system.

Case Study 5: Document Editing Application

Scenario:
You are designing a document editing application. Users can make various edits to the document, such as adding text, deleting text, or formatting it. The application needs to support undo/redo functionality so that users can easily revert or redo changes they have made to the document.

Question:
Which design pattern would be most appropriate for implementing the undo/redo functionality in this document editor? Explain the rationale behind the choice and describe how the pattern would be applied.

Case Study 6: Notification System

Scenario:
You are building a notification system for an e-commerce application. The system should notify users of different events (order shipped, discount available, etc.) through various channels (email, SMS, push notification). New notification channels might need to be added in the future.

Question:
Which design pattern should you use to implement this notification system? Explain why it is the best fit and outline how it would be implemented, including any required interfaces or components.

Case Study 7: Media Player Application

Scenario:
You are designing a media player application that supports multiple file formats (e.g., MP3, MP4, WAV). The system should allow for easy addition of new media formats without changing the core media player logic.

Question:
Which design pattern would you use to manage different media formats in the player? Explain how the pattern would work and how new formats could be added easily.

Case Study 8: Customer Support Chatbot

Scenario:
You are building a customer support chatbot that can assist customers with various tasks, such as answering frequently asked questions or processing simple orders. The chatbot should be able to handle different types of questions and tasks dynamically without requiring constant code changes.

Question:
Which design pattern would be effective for this chatbot system? Justify your choice and describe how it would help in handling different customer inquiries and tasks.

Case Study 9: Traffic Control System

Scenario:
You are tasked with designing a traffic control system where traffic lights control the flow of cars at intersections. The lights should change according to the current time of day (e.g., rush hour or late night), and the system needs to be flexible to accommodate new traffic rules or timing constraints in the future.

Question:
Which design pattern would be most effective in managing the behavior of the traffic lights based on the time of day? Explain the reasoning behind your choice and describe the structure of the system.

Case Study 10: Shopping Cart System

Scenario:
In an online store, the shopping cart system needs to handle various types of items (e.g., physical products, digital downloads, promotional discounts). Some items may require additional processing (e.g., calculating shipping costs), while others may not. The system should allow adding new types of items and processing them accordingly.

Question:
Which design pattern would you use to manage the shopping cart system? Explain how the pattern would manage different types of items and the processing required for each, including how to add new item types in the future.