UML and Design Patterns

Table of contents

1 Multiple Choice Questions

1. **What is the primary purpose of Creational Design Patterns?**
   - A. To manage application state
   - B. To simplify object creation and increase flexibility
   - C. To handle system security
   - D. To optimize application performance
2. **Which Design Pattern delegates the creation of objects to subclasses?**
   - A. Singleton Pattern
   - B. Factory Method Pattern
   - C. Abstract Pattern
   - D. Builder Pattern
3. **In the Factory Method Pattern, what is the primary benefit of polymorphism?**
   - A. Increased security
   - B. Improved performance

- C. Flexibility in object creation
- D. Simplified code structure

4. **Which of these is NOT a main category of Design Patterns according to the slides?**

   - A. Creational Patterns
   - B. Structural Patterns
   - C. Functional Patterns
   - D. Behavioral Patterns

5. **What does the Abstract Factory Pattern provide?**

   - A. An interface for creating individual objects
   - B. A way to create families of related objects
   - C. A method for deleting objects
   - D. A pattern for building complex objects step-by-step

6. **Which pattern is often used in game development for creating different types of enemies?**

   - A. Factory Method Pattern
   - B. Singleton Pattern
   - C. Builder Pattern
   - D. Prototype Pattern

7. **In the context of Factory Patterns, what does 'Encapsulation' primarily achieve?**

   - A. It enhances the performance of the application
   - B. It hides the creation logic of objects
   - C. It makes the application more secure
   - D. It simplifies the debugging process

8. **The Factory Method Pattern in Java often uses which of the following concepts?**

   - A. Inheritance
   - B. Serialization
   - C. Multithreading
   - D. Recursion

9. **In a UML diagram for the Factory Method Pattern, which relationship typically exists between 'Factory' and 'ConcreteFactory'?**

   - A. Aggregation
   - B. Composition
   - C. Inheritance

- D. Association

10. **The 'Product' in the Factory Method Pattern is usually represented as:**
    - A. A concrete class
    - B. An abstract class or interface
    - C. A static method
    - D. A global variable
11. **In the Abstract Factory Pattern, 'ConcreteFactory1' and 'ConcreteFactory2' typically:**
    - A. Create the same type of products
    - B. Create different families of related products
    - C. Act as Singleton classes
    - D. Implement the same concrete class
12. **The Factory Method Pattern is mainly concerned with:**
    - A. Creating complex objects step by step
    - B. Delegating instantiation to subclasses
    - C. Managing global state across the application
    - D. Ensuring a class has only one instance

2 Short Answer Questions

1. Explain the Factory Method Pattern with appropriate UML diagrams.
2. Discuss the differences between the Factory Method and Abstract Factory Patterns citing examples.
3. Explain the concept of polymorphism in the context of the Factory Method Pattern with a code snippet.
4. Discuss how the Abstract Factory Pattern can be used for theme switching in an application, with a brief code example.

3 UML Questions

1. Given an online bookstore system with functionalities for browsing books, adding books to a cart, and checking out, design a UML class diagram representing the key classes and their relationships.

2. Develop a UML use case diagram for a car rental system where users can search for available cars, book a car, and return a car. Include actors such as Customer and Admin.
3. Illustrate the sequence of interactions between a user, a web application, and a database when the user logs into the application.
4. Design a UML state diagram for an online order that transitions through states: Created, Processed, Shipped, and Delivered.
5. Based on the class diagram you created for the online bookstore system (Question 1), draw a UML object diagram representing specific instances of books and a cart at a particular moment.
6. Design a UML package diagram to organize the modules of a university management system into packages such as Enrollment, Grading, and Scheduling.

4 Design Pattern Questions

1. **Document Management System**

   Imagine you are designing a document management system for a company. This system should allow employees to create various types of documents such as `TextDocuments`, `SpreadsheetDocuments`, and `PresentationDocuments`. Each document type has its unique properties and behaviors, but they share some common operations like `open()`, `save()`, `edit()`, and `close()`.

   The system should be extensible, allowing for the addition of new document types in the future without modifying the core system. Employees don't need to know the details of document creation; they just request a document of a particular type, and the system should provide it.

4.1 Question

How can you design the document management system using the Factory Method design pattern to allow for flexibility and extensibility in creating various types of documents? Implement a solution that demonstrates the creation of different document types through a unified interface but delegates the instantiation of document objects to subclasses corresponding to each document type (give your source code and appropriate UML diagrams in your response).

2. **Problem Scenario: Cross-Platform UI Widget Toolkit**

   Imagine you are tasked with designing a cross-platform UI (User Interface) toolkit for a software development company. This toolkit should enable developers to create applications that can run on multiple operating systems (OS) like Windows, MacOS, and Linux without altering the codebase for UI components. Each OS has a distinct look and feel for its UI components, such as Buttons, TextFields, and CheckBoxes.

   The toolkit should provide a unified way to create these UI components so that they are rendered correctly according to the OS on which the application is running. Moreover, the system should be extensible to support new types of UI components or new operating systems without significant changes to the existing codebase.

4.2 Question

How can you design the cross-platform UI toolkit using the Abstract Factory pattern to ensure the system can produce families of related objects (UI components) for different operating systems without specifying their concrete classes? Your design should allow for easy extension to accommodate new UI components and new operating systems (give your source code and appropriate UML diagrams in your response).