Object-Oriented Methodology Quiz 09

2024 Fall Semester

21 CST H3Art

Final Score: 95/100

1. Which class defines the createPizza() method as abstract in the Factory Method example?

A. SimplePizzaFactory
B. NYPizzaStore
C. Pizza
D. PizzaStore

2. What is the role of the CondimentDecorator class in the example?

A. It holds the size of the beverage.
B. It extends the behavior of beverages.
C. It manages inventory.
D. It represents a beverage type.

3. Which of the following is NOT a factory pattern?

A. Abstract Factory
B. Simple Factory
C. Singleton Pattern
D. Factory Method

4. What is the advantage of using the Factory Pattern?

A. It reduces the dependency of clients on concrete classes.
B. It makes code harder to understand.
C. It increases the coupling between components.
D. It simplifies the creation of objects.

5. How does the Decorator Pattern achieve type matching between decorators and the objects they decorate?

A. By modifying existing code
B. By using interfaces
C. By subclassing the abstract component class
D. By using composition and delegation

6. What is the purpose of the createPizza() method in the PizzaStore example?

A. To bake the pizza.
B. To box the pizza.
C. To prepare the pizza.
D. To instantiate a concrete pizza class.

7. What is the Dependency Inversion Principle?

A. Both high-level and low-level components should depend on abstractions.
B. High-level components should depend on low-level components.

C. Components should not depend on each other.
D. Low-level components should depend on high-level components.

8. What is the main purpose of the Factory Pattern?

A. To create objects directly.
B. To define an interface for a single product.
C. To encapsulate object creation.
D. To provide a static method for creating objects.

9. In the Factory Method Pattern, which class is responsible for creating objects?

A. The Concrete Product class
B. The Creator class
C. The Factory class
D. The Concrete Creator class

10. What is the purpose of the FilterInputStream class in Java I/O?

A. It manages file permissions.
B. It handles output streams.
C. It implements the read() method for all input streams.
D. It serves as an abstract decorator for input streams.

11. What design principle does the Decorator Pattern follow?

A. Liskov Substitution Principle
B. Open-Closed Principle
C. Interface Segregation Principle
D. Single Responsibility Principle

12. Which of the following statements about the Abstract Factory Pattern is TRUE?

A. It provides an interface for creating families of related products.
B. It can only create a single product.
C. It does not decouple clients from concrete classes.
D. It is implemented using inheritance.

13. What is a potential downside of the Decorator Pattern?

A. It eliminates the need for interfaces.
B. It can result in a large number of small classes.
C. It reduces the number of classes needed.
D. It increases code readability.

14. What is the main problem that Starbuzz Coffee faces when trying to update their ordering systems?

A. Difficulty in adding new beverage options
B. Not enough customers
C. Overuse of inheritance leading to complex designs
D. Lack of funding

15. What happens when the cost() method is called on a decorated beverage?

A. It adds the cost of all decorators and the beverage.
B. It returns the cost of the beverage only.

C. It throws an exception.
D. It returns the cost of the innermost object.

16. How do decorators typically change the behavior of the objects they decorate?

A. By adding new functionality before or after method calls
B. By modifying existing code
C. By overriding methods
D. By adding new methods

17. What is the primary goal of the Decorator Pattern?

A. To add responsibilities to objects dynamically
B. To reduce class explosion
C. To enforce encapsulation
D. To create immutable objects

18. What is the role of the PizzaIngredientFactory interface in the Abstract Factory example?

A. To define methods for creating a family of pizza ingredients.
B. To provide implementations for creating pizza ingredients.
C. To instantiate concrete pizza classes.
D. To define methods for creating concrete pizza classes.

19. Which of the following is a key characteristic of the Decorator Pattern?

A. It removes functionality from existing objects
B. It violates the Open-Closed Principle
C. It requires the use of interfaces
D. It adds behavior to objects at runtime

20. What does the LowerCaseInputStream decorator do?

A. It sets the size of the input stream.
B. It counts the number of lines read.
C. It converts uppercase characters to lowercase.
D. It adds buffering to input streams.

21. Factories are a powerful technique for coding to interfaces, not implementations.

✅

❌ **22. The CondimentDecorator class extends the Beverage class to add new behavior.**

✅

Correct Answer: ❌

❌ **23. The Decorator Pattern requires the use of inheritance to achieve type matching.**

❌

Correct Answer: ✅

24. The Simple Factory is considered a real design pattern.

❌

25. The Beverage class in the example is an abstract class.

> ✅

26. The Decorator Pattern allows objects to be extended at runtime without modifying existing code.

> ✅

27. The Dependency Inversion Principle states that high-level components should depend on low-level components.

> ❌

28. The PizzaIngredientFactory interface in the Abstract Factory example defines methods for creating families of pizza ingredients.

> ✅

29. Decorators can be used to add multiple new responsibilities to objects dynamically.

> ✅

30. Decorators are typically transparent to the client code that uses them.

> ✅

31. The Java I/O libraries make extensive use of the Decorator Pattern.

> ✅

32. The Abstract Factory Pattern relies on inheritance to create objects.

> ❌

33. The Factory Pattern is used to directly instantiate objects.

> ❌

34. The Dependency Inversion Principle guides us to depend on abstractions, not concrete classes.

> ✅

35. The getDescription() method of a decorated beverage returns a description that includes all decorators.

> ✅

36. The Decorator Pattern violates the Open-Closed Principle.

> ❌

37. The Factory Method Pattern allows clients to be decoupled from the concrete classes they use.

> ✅

38. In the PizzaStore example, the orderPizza() method creates concrete pizza objects.

> ❌

39. The Factory Pattern promotes tight coupling between objects.

> ❌

40. The PizzaStore example demonstrates the use of the Factory Method Pattern.

> ✅