1. Mohsin needs to create various user objects for his University learning platform. What is the act of creating an object called?

1 / 1 point

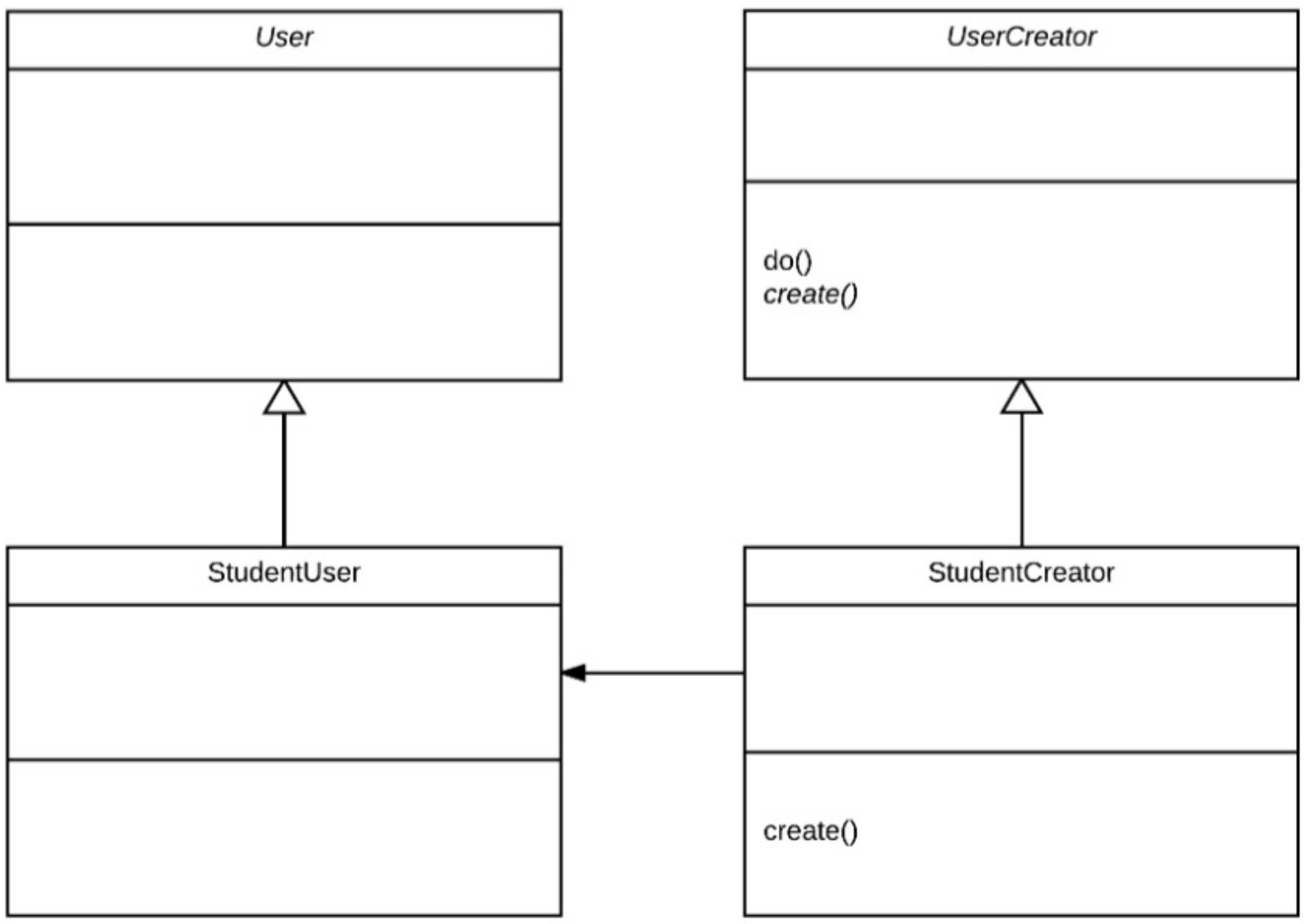
- (●) concrete instantiation
- ( ) object invocation
- ( ) object realization
- ( ) class creation

Correct

Correct! Concrete instantiation is when an object of a class is actually created.

1 / 1 point

2. Mohsin has a superclass that performs various operations on these user objects - Student, Professor, Assistant, for example. He wants the subclass to determine which object is created. This is sketched below in a UML diagram for the StudentUser class. What is this design pattern called?



- Template Pattern
- Factory Method Pattern
- Simple Factory
- Composite Pattern

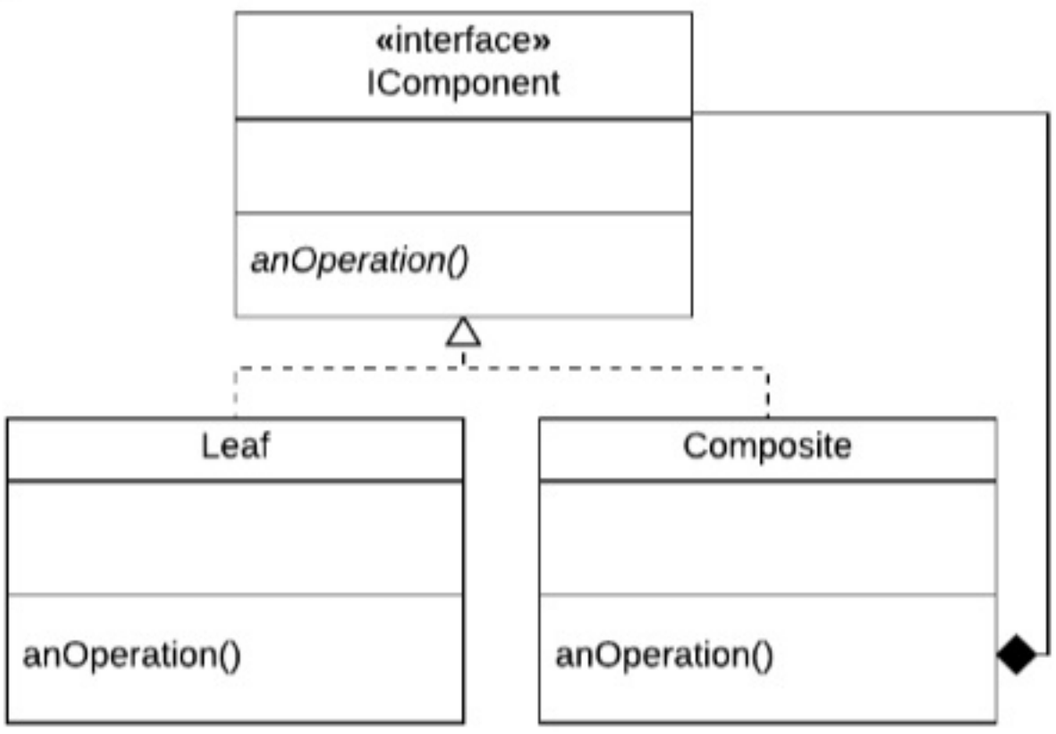
Correct

Correct! The Creator superclass in the Factory Method pattern has operations that operate on an object, but has the actual creation of that object outsourced to an abstract method that must be defined by the subclass.

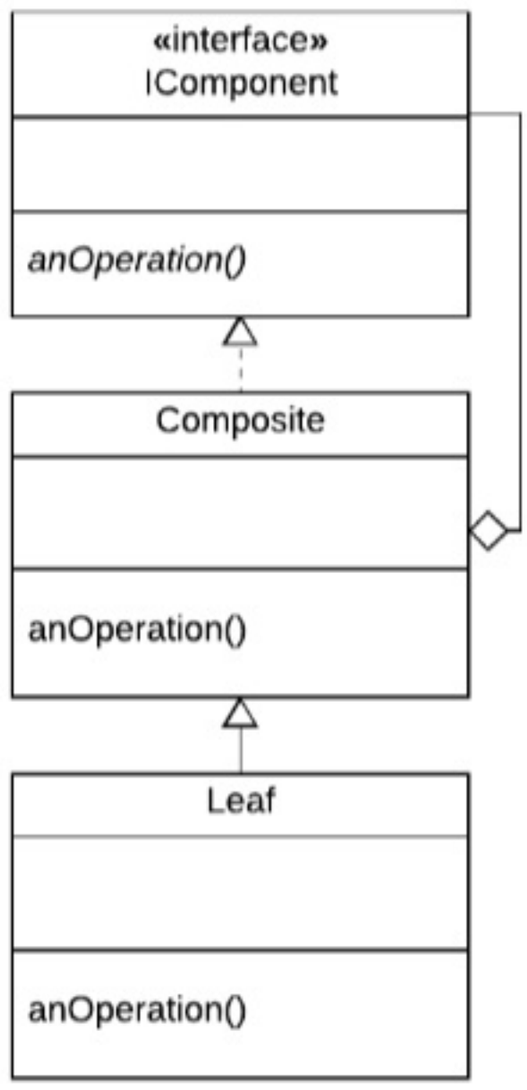
3. Select the correct UML class diagram representation of the Composite Pattern:

1 point

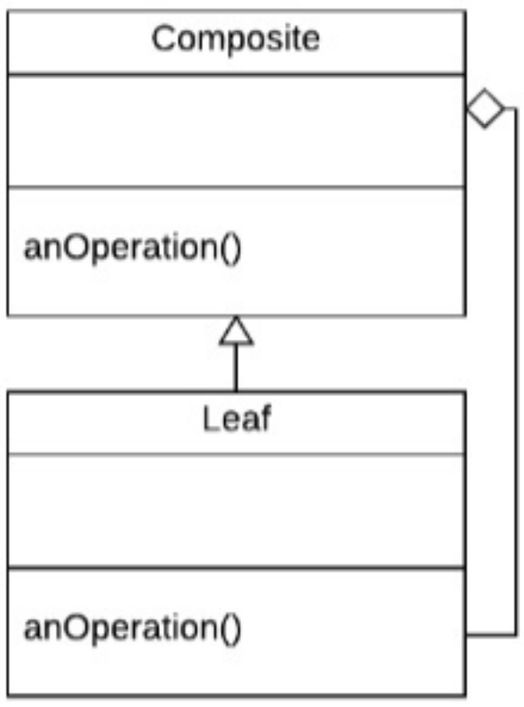
a)



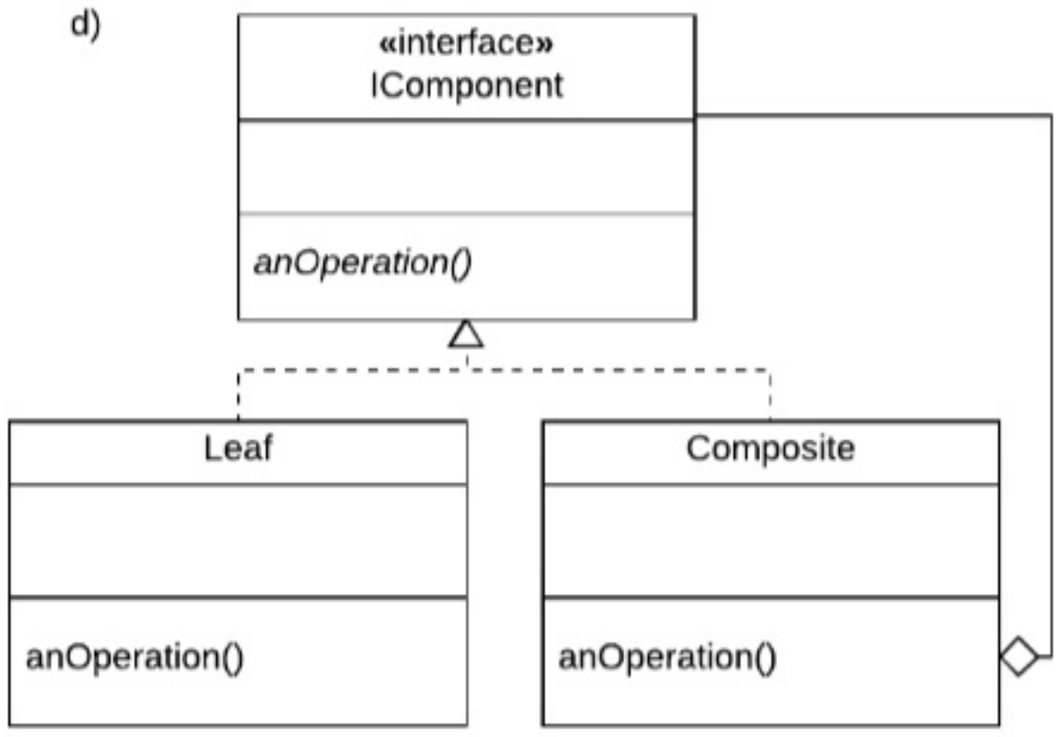
b)



c)



d)



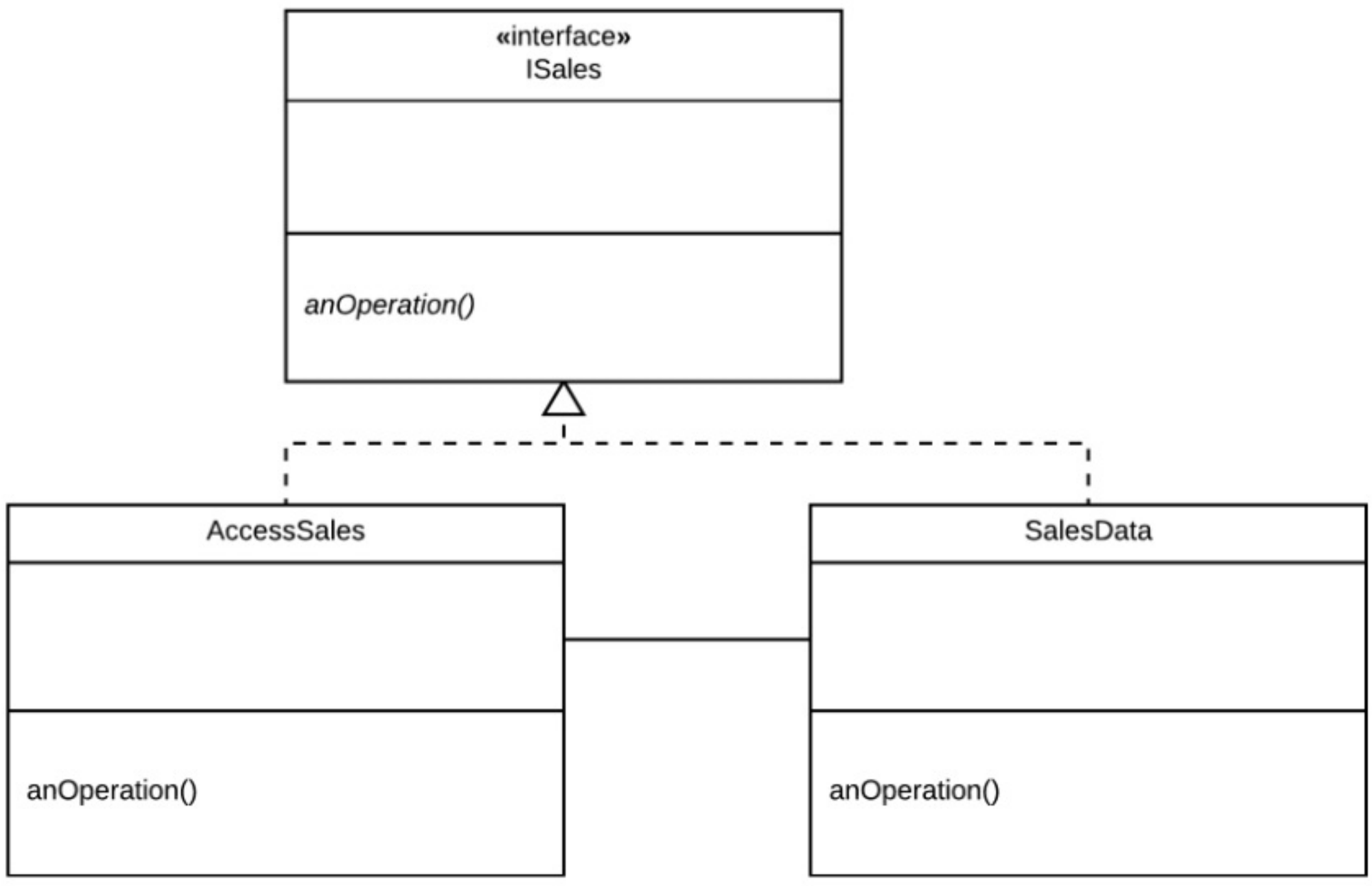
- ○ a)
- ◉ b)
- ○ c)
- ○ d)

Incorrect

Incorrect. This would require Leaf to inherit behaviour from the Composite class, which is not usually a desirable feature! It is also not clear from this diagram that the Composite class can contain Composite objects.

4. Yola is programming for a grocery store system. She has a complex SalesData class that updates inventories and tracks sales figures, and a lightweight AccessSales class that will give select sales data to a user, depending on their credentials. AccessSales delegates to SalesData when more complex data is needed. This situation is shown below. Which Pattern is this?

1 / 1 point



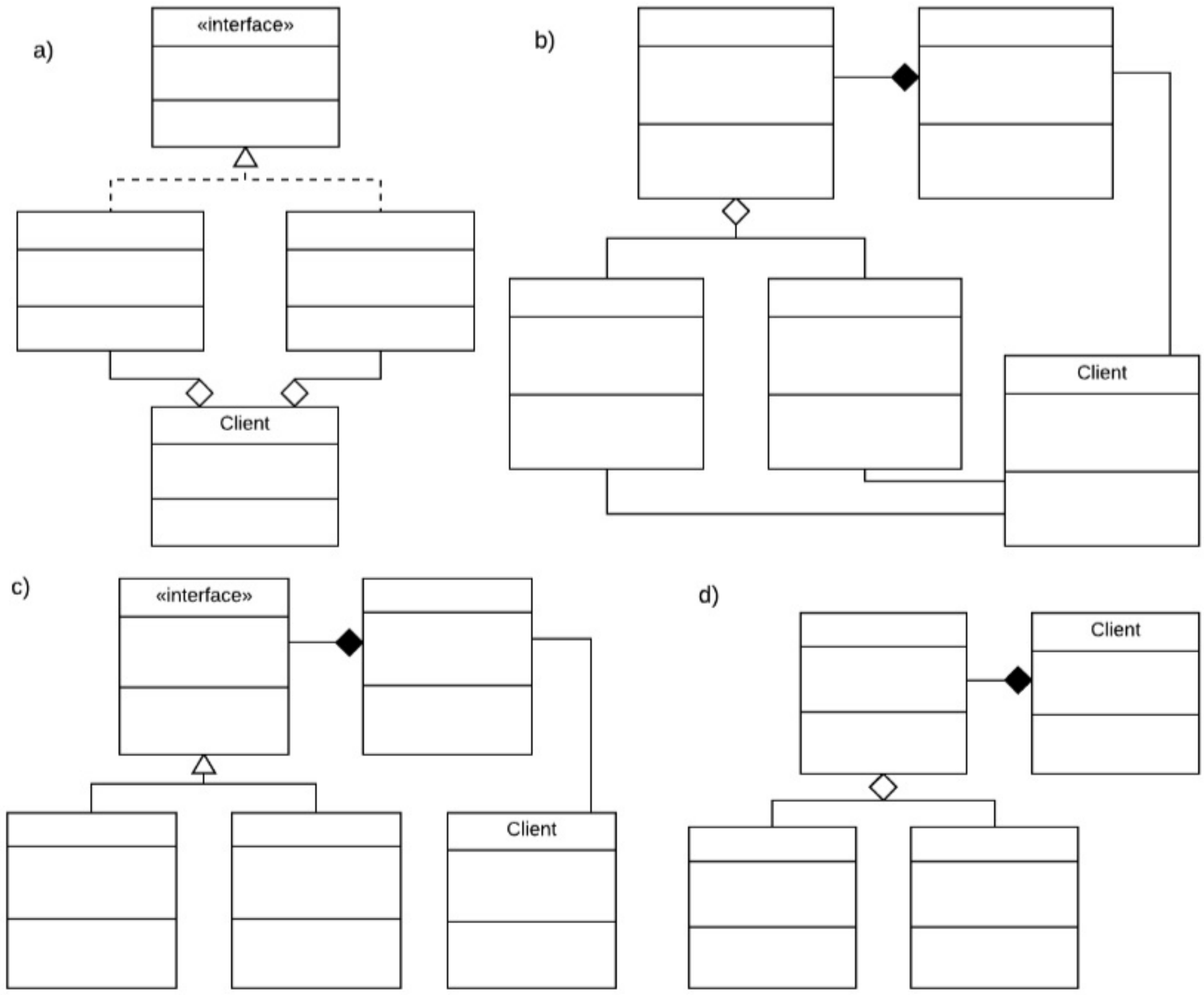
- Singleton Pattern
- Decorator Pattern
- Facade Pattern
- Proxy Pattern

Correct

Correct! This is a proxy. The AccessSales object acts as a lightweight version of the SalesData class.

5. Which of these UML class diagrams shows the Facade pattern?

1 / 1 point



- a)
- b)
- c)
- d)

Correct
Correct! The client interacts with only the Facade. The Facade then manages the subsystem.

6. What is the difference between the Factory Method and a Simple Factory?

1 / 1 point

- In the factory method pattern, the factory itself must be instantiated before it starts creating objects. This is usually done with a dedicated method.
- In Factory Method, concrete instantiation is done in a designated method, where a Simply Factory creates objects for external clients
- Simple factories cannot be subclassed.
- A simple factory instantiates only one kind of object.

Correct
Correct! This is a pretty good short definition of a factory method.

7. José wants to build behaviours by stacking objects and calling their behaviours with an interface. When he makes a call on this interface, the stack of objects all perform their functions in order, and the exact combination of behaviours he needs depends what objects he stacked and in which order. Which Design Pattern best fits this need?

1 / 1 point

- ○ Singleton Pattern
- ○ Composite Pattern
- ○ Factory Method Pattern
- ◉ Decorator Pattern

Correct
Correct! Decorator is a great pattern when you need to add behaviours with aggregation.

8. You need to connect to a third-party library, but you think it might change later, so you want to keep the connection loosely coupled by having your object call a consistent interface. Which Design Pattern do you need?

1 / 1 point

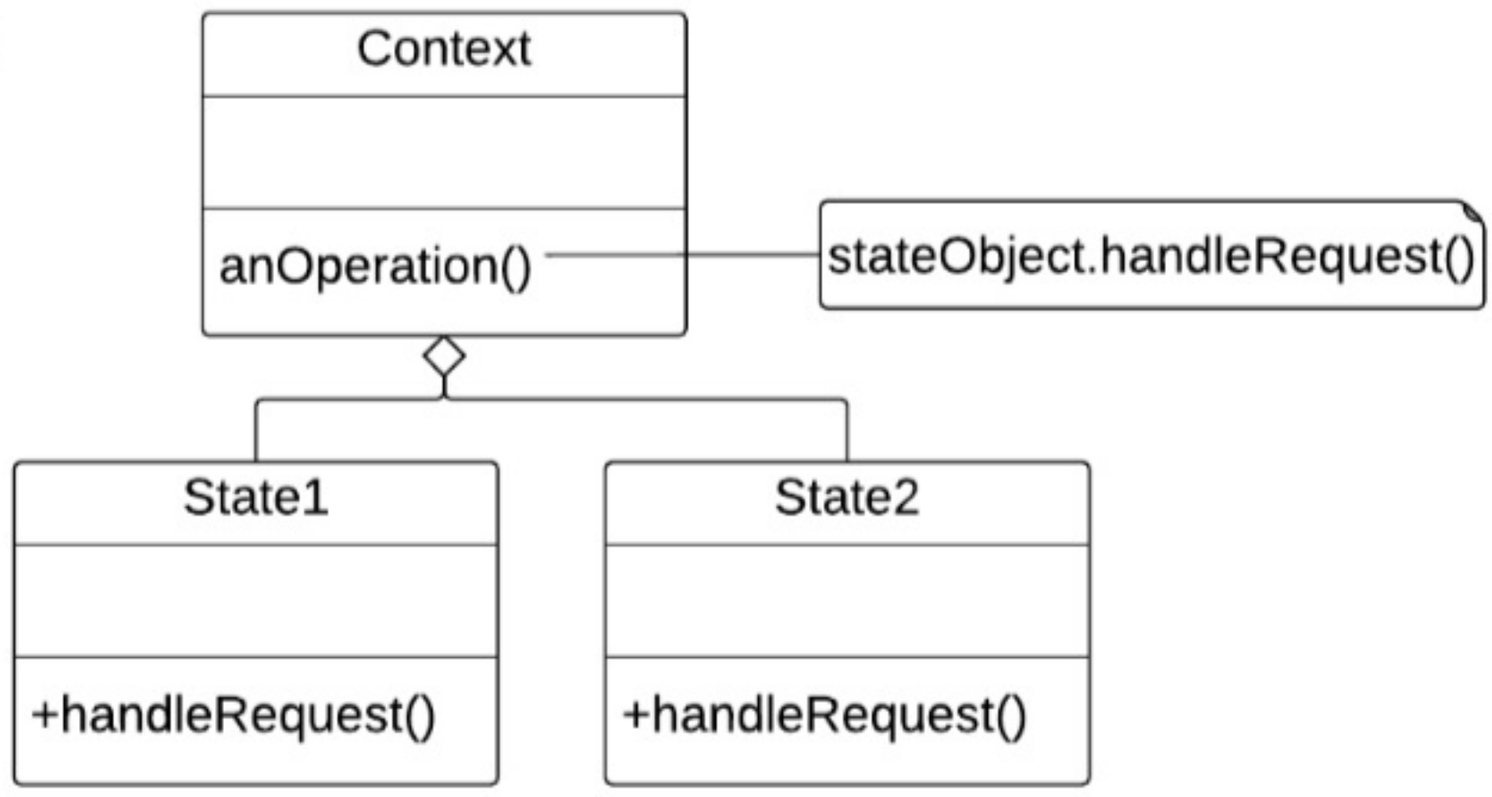
- ◉ Adapter
- ○ Proxy
- ○ Facade
- ○ Decorator

Correct
Correct! The adapter pattern keeps loose coupling between the client and the interface in question. If either changes, only the adaptor needs to be changed.

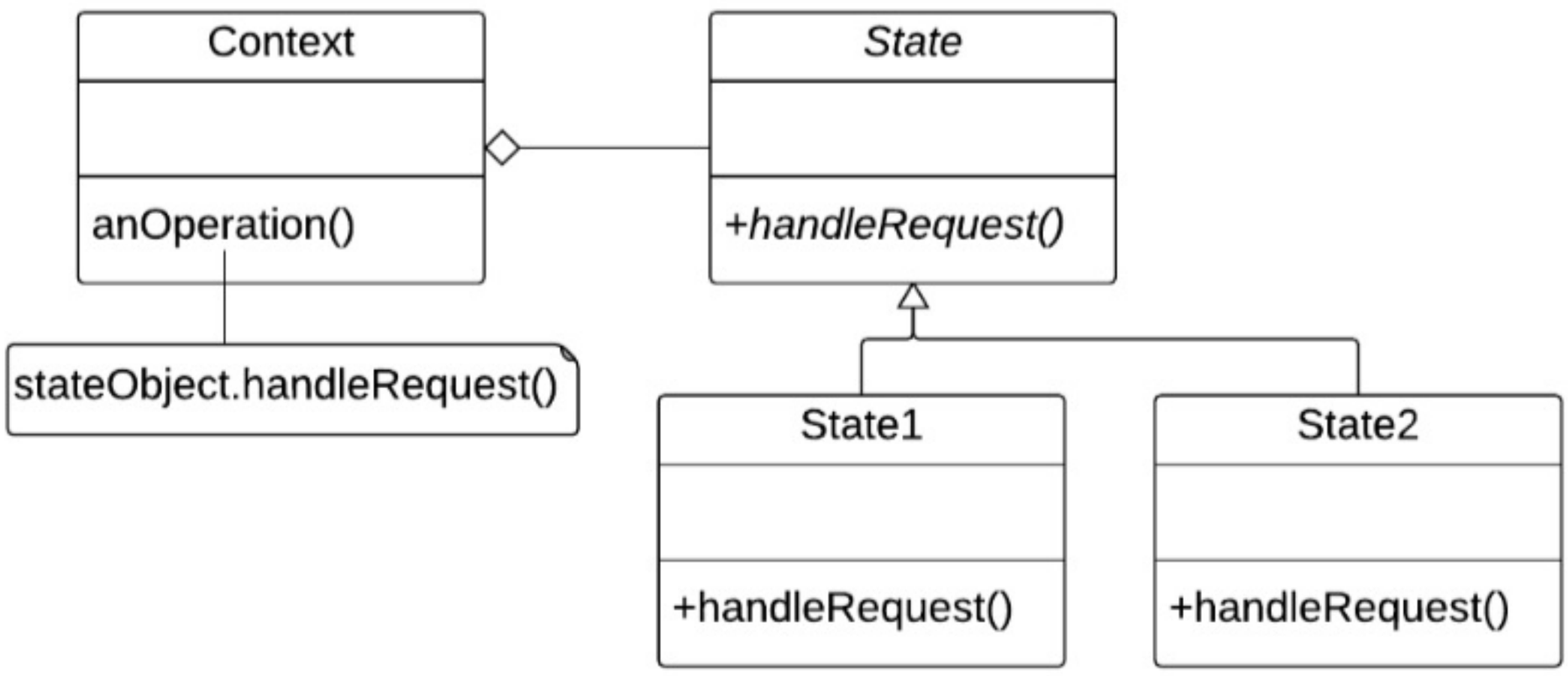
9. Which of these diagrams shows the State pattern?

1 point

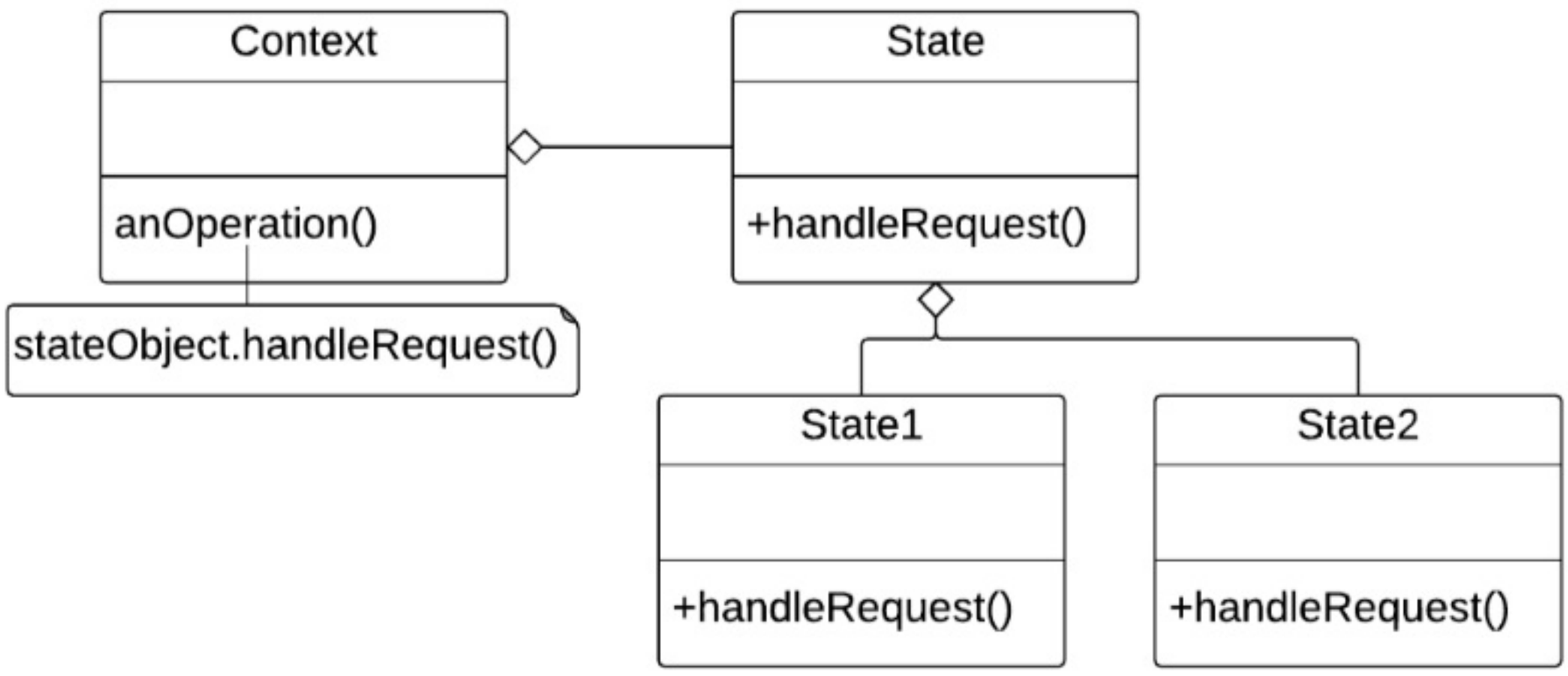
a)



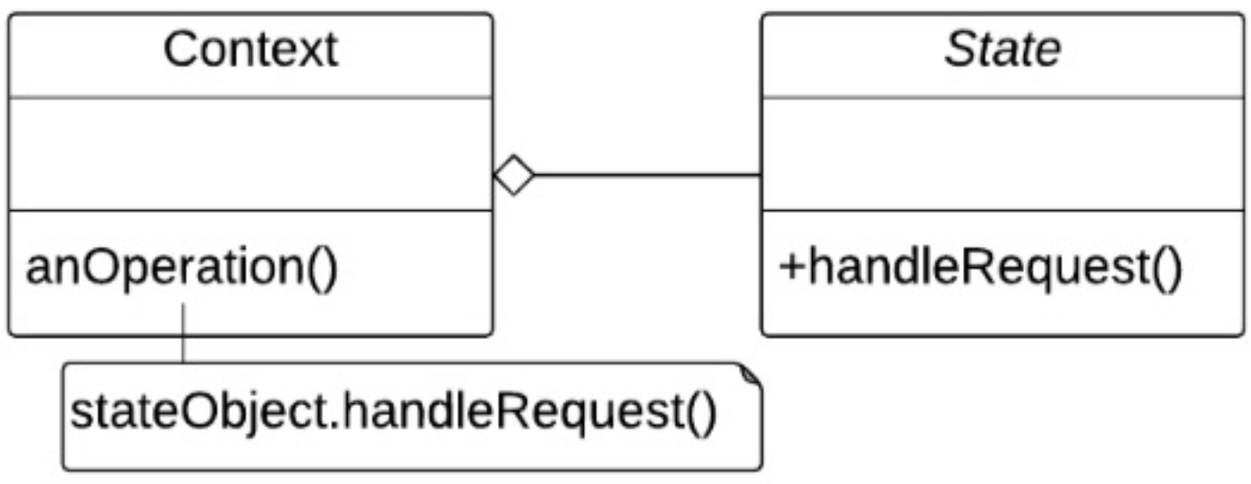
b)



c)



d)



- ○ a)
- ○ b)
- ○ c)
- ◉ d)

Incorrect

Incorrect. This doesn't work at all! The State of your context cannot be set using an object.

10. Which of these design principles best describes the Proxy pattern?

1 / 1 point

- ○ separation of concerns, because the Proxy object has different concerns from the subject
- ○ decomposition, because the Proxy object has different concerns than the subject
- ◉ encapsulation, because the Proxy hides some of the detail of the subject
- ○ generalization, because a proxy is a general version of the real subject

Correct
Correct! The Proxy encapsulates some behaviour of the subject in a simpler way, and delegates to the subject when needed.

11. Ashley has a method in her class that needs to makes a request. This request could be handled by one of several handlers. Which design pattern does she need?

1 / 1 point

- ○ Facade
- ○ Decorator
- ◉ Chain of Responsibility
- ○ Template

Correct
Correct! The Chain of Responsibility is a pattern for passing a request down a line until one of the handlers can handle it.

12. Colin is designing a class for managing transactions in software for a banking machine software. Each transaction has many of the same steps, like reading the card, getting a PIN, and returning the card. Other steps are particular to the type of transaction. Which pattern does he need?

1 / 1 point

- ◉ Template
- ○ State
- ○ MVC
- ○ Mediator

Correct
Correct! The Template method is used for situations in which the same general set of steps are followed, but some steps are different in their specifics.

13. Which of these is **NOT** a good use of the Command pattern?

1 / 1 point

- ○ Building a user-interface that can be used to perform operations
- ○ Building macros, for example in an image manipulation program
- ○ Supporting undo/redo queues of commands
- ◉ Sending a command to a third-party service or library

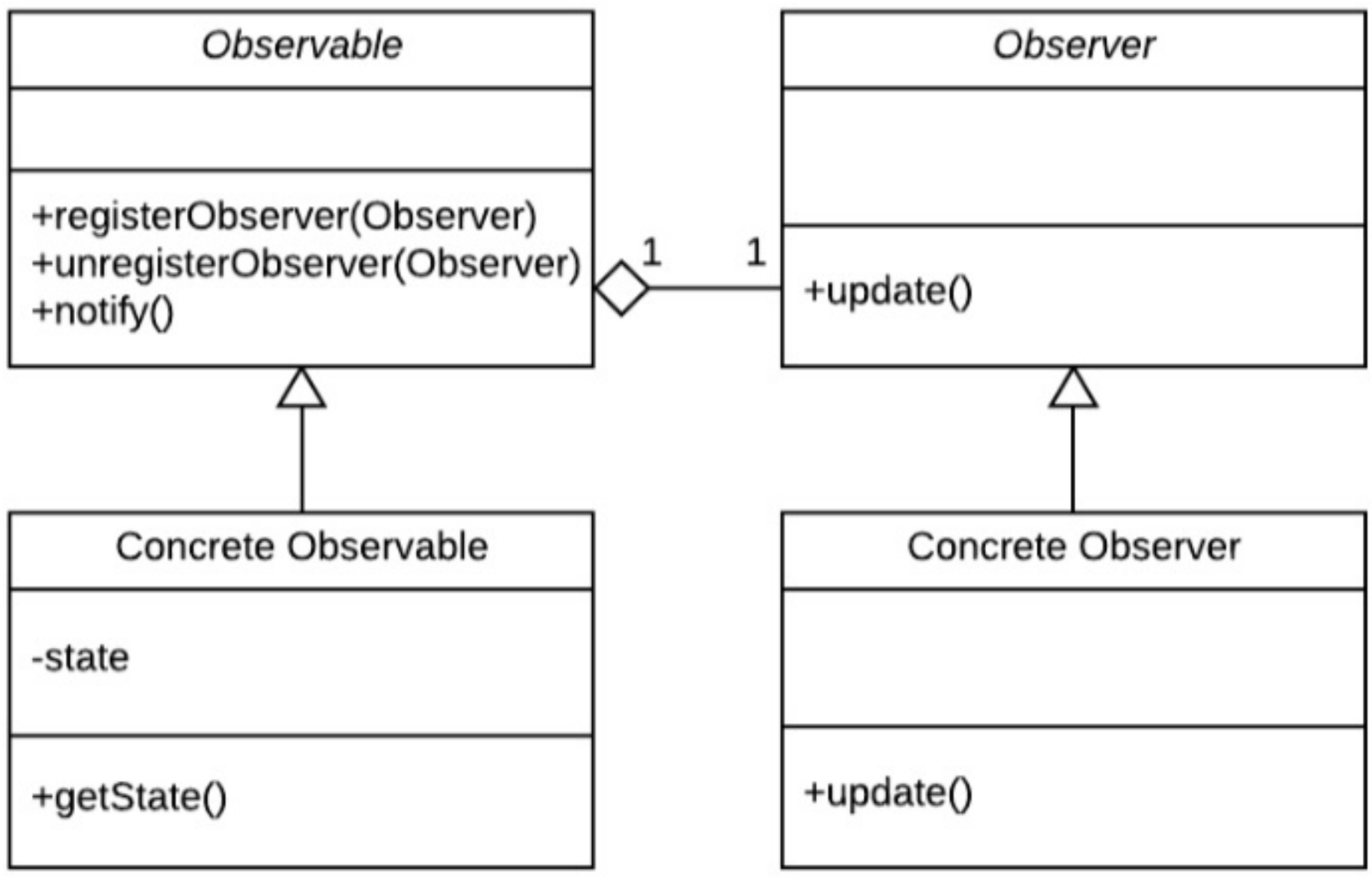
Correct

Correct! This better describes the Facade or Adapter pattern.

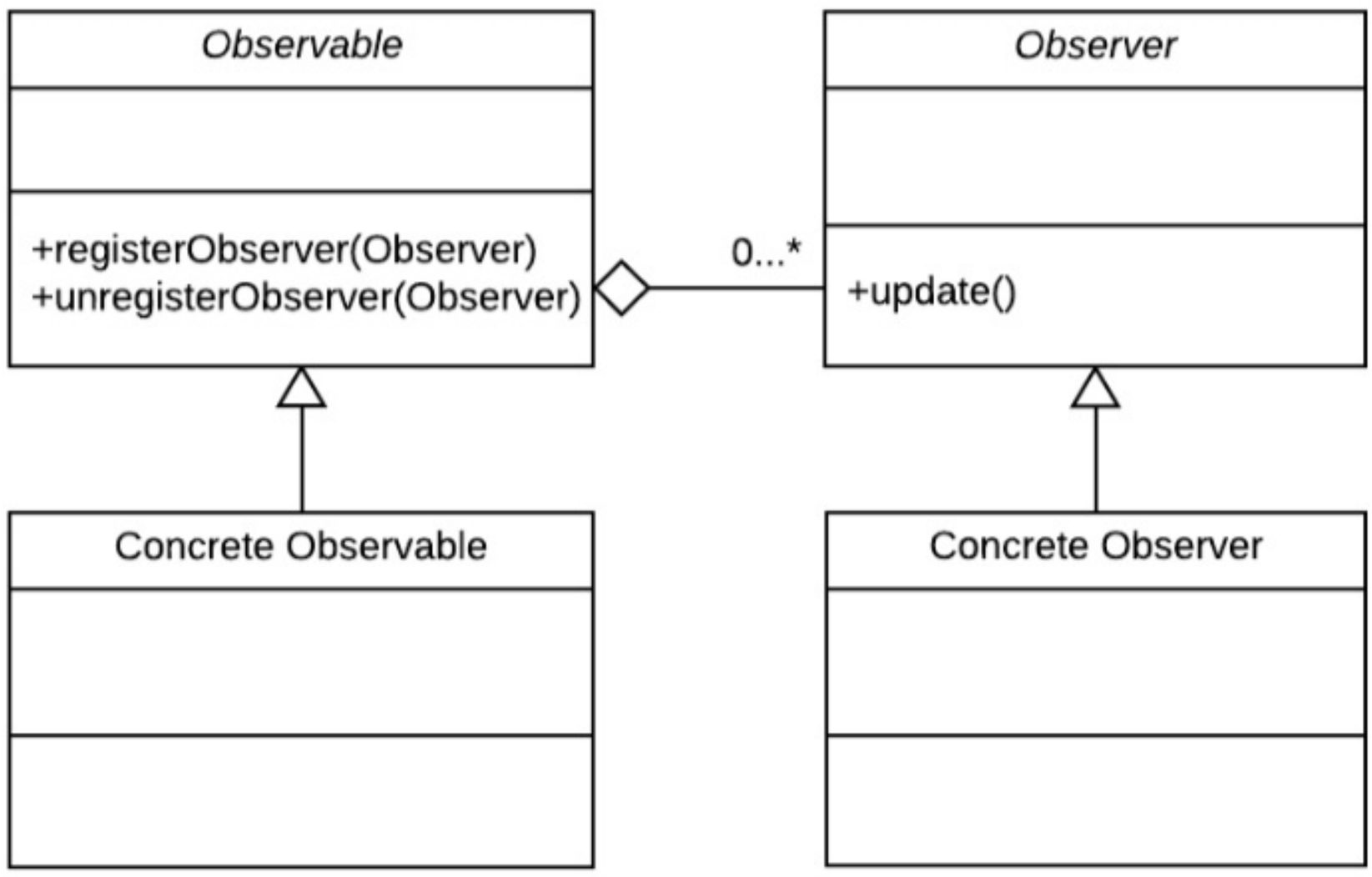
14. Choose the correct UML class diagram representation of the Observer pattern:

1 / 1 point

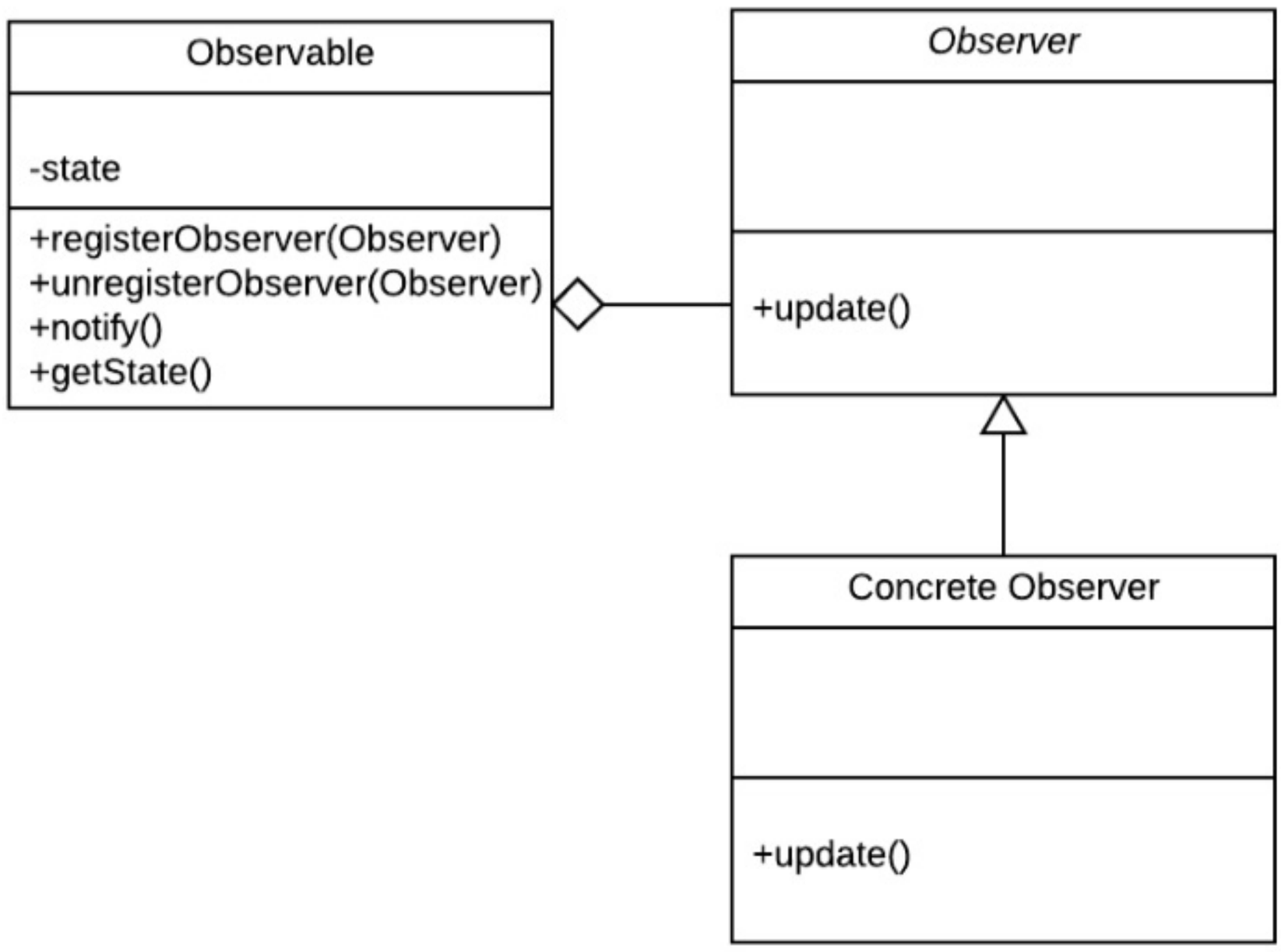
a)



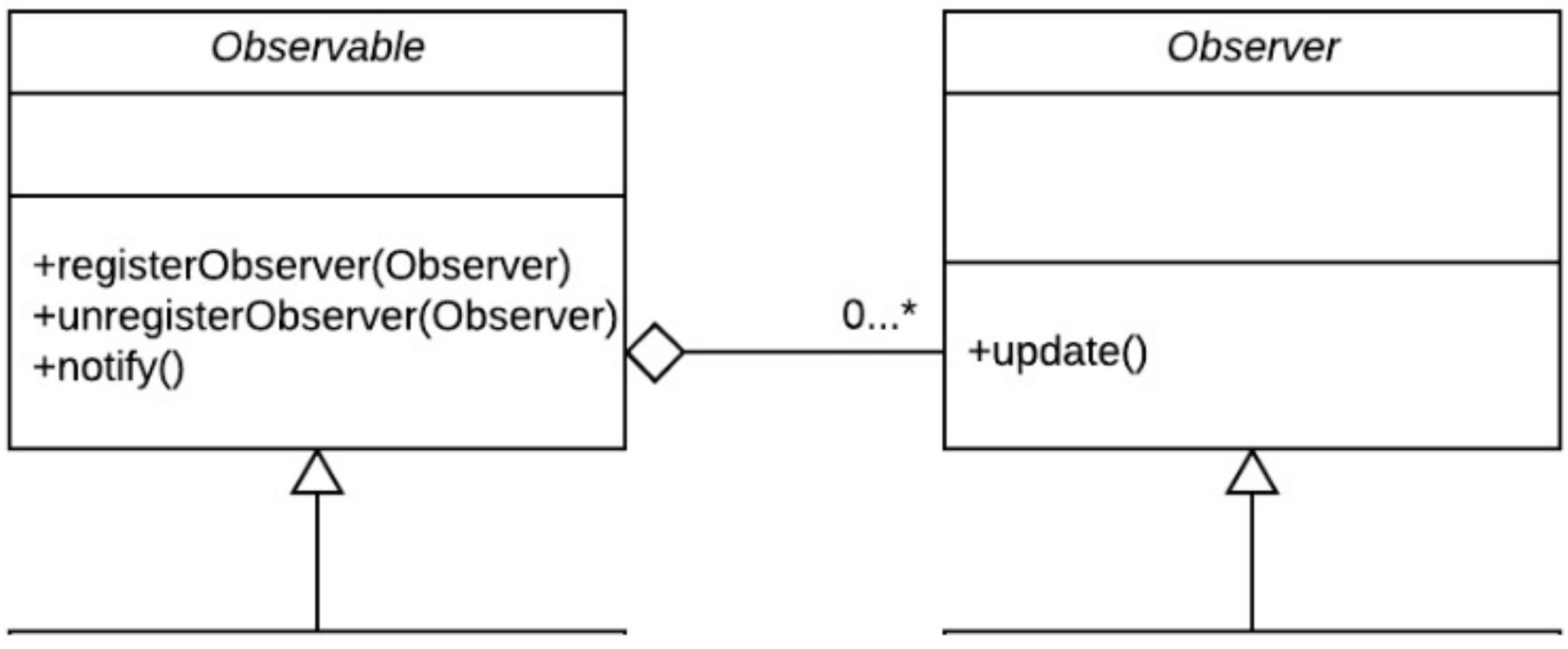
b)

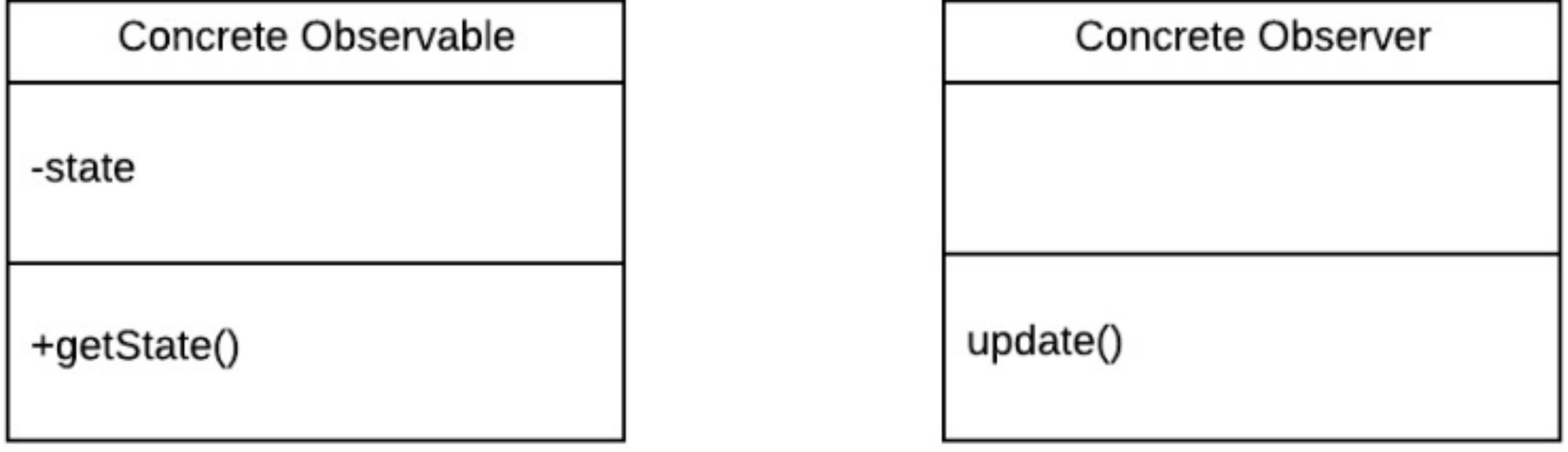


c)



d)





- a)
- b)
- c)
- d)

Correct
Correct! This diagram has all the correct elements of an Observer pattern.

15. Which code smell may become a problem with the Mediator design pattern?

1 / 1 point

- Speculative Generality
- Large Class
- Inappropriate Intimacy
- Refused Bequest

Correct
Correct! The Mediator class can quickly become very large, which means it might have this or related code smells, like Divergent Change or Long Method.

16. Hyun-Ji is developing a program. She wants to create a Student class that behaves differently based on if the student has not registered for classes, is partially registered, fully registered, or fully registered and paid. Which design pattern does she need?

1 / 1 point

- Mediator
- Proxy
- State
- Template Method

Correct
Correct! The State of the student will determine its responses to various requests. Exactly what she needs.

17. Which of these methods is found in a typical Observer class?

1 / 1 point

- ○ notify()
- ○ getState()
- ◉ update()
- ○ addObserver()

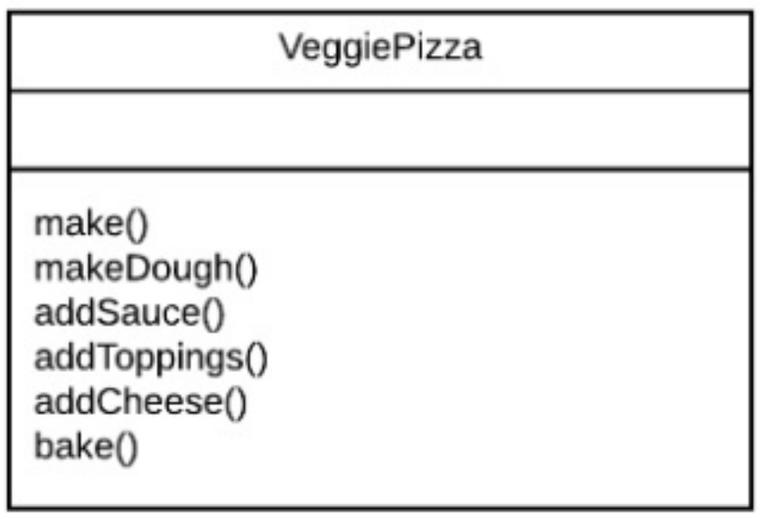
Correct

Correct! The Observer class needs to update itself.

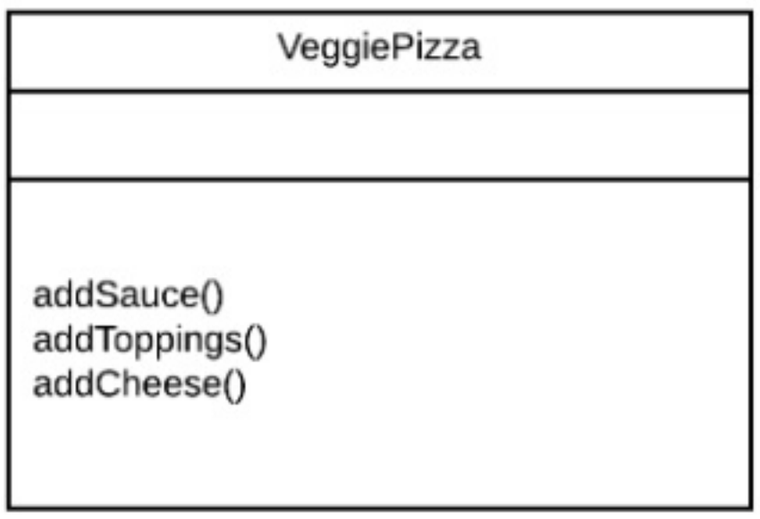
18. Fernando is making pizza objects with the Template Method pattern. The make() function is the whole process of making the pizza. Some steps are the same for every pizza - makeDough(), and bake(). The other steps - addSauce(), addToppings() and addCheese() - vary by the pizza. Which of these subclasses shows the proper way to use a template method?

1 point

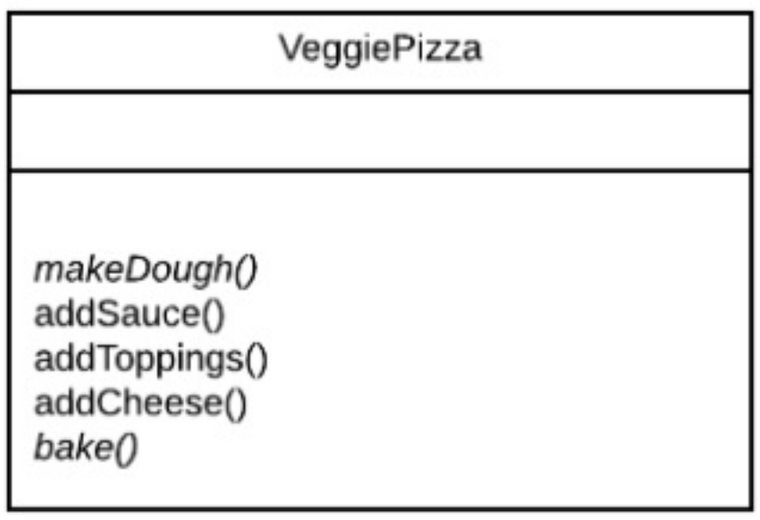
a)



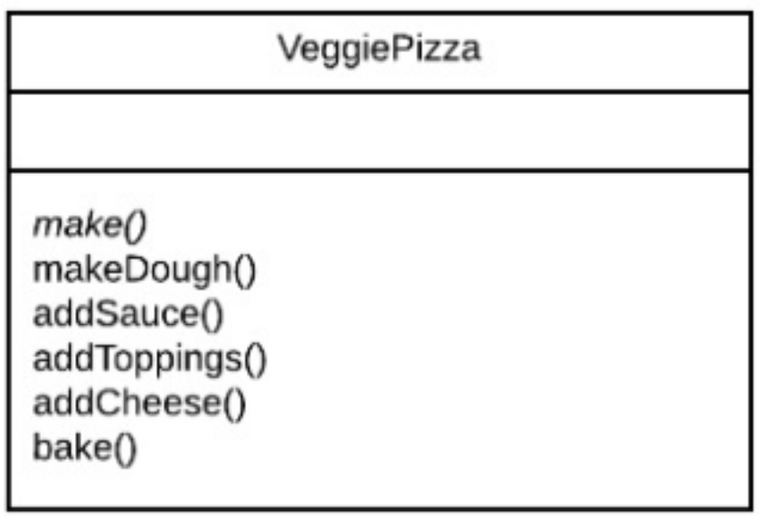
b)



c)



d)



- ○ a)
- ○ b)
- ◉ c)
- ○ d)

Incorrect

Incorrect. You're getting there, but it doesn't make sense to have abstract methods in your subclass, unless you're planning another subclass.

19. In the Mediator Pattern, which pattern is often used to make sure the Mediator always receives the information it needs from its collaborators?

1 / 1 point

- ○ Template Method
- ○ Command
- ○ Chain of Responsibility
- ◉ Observer

Correct
Correct! The Mediator can be made an Observer of all of its Collaborators.

20. In the MVC Pattern, which of these is usually made into an Observer?

1 / 1 point

- ◉ View
- ○ Model
- ○ Back-End
- ○ Controller

Correct
Correct! Views are usually subscribed to the model so that when changes are made, the views are updated.

21. Which of these answers does **NOT** accurately complete the following sentence? “A class is considered closed to modification when…”

1 point

- ◉ ...all the attributes and behaviours are encapsulated
- ○ ...it is tested to be functioning properly
- ○ ...it is proven to be stable within your system
- ○ ...its collaborators are fixed

Incorrect
Incorrect. This is an aspect of being closed to modification. It prevents unintended access.

22. How does the Dependency Inversion Principle improve your software systems?

1 / 1 point

- ○ Dependency becomes inverted by having the system depend on the client classes
- ◉ Client classes become dependent on high level generalizations rather than dependant on low level concrete classes
- ○ Client classes become dependant on low-level concrete classes, rather than dependant on high-level generalizations
- ○ Client classes use an adapter to facilitate communication between itself and the rest of the system

Correct

Correct! Being dependent on a generalization allows your system to be more flexible.

23. Allison has a search algorithm, and she would like to try a different implementation of it in her software. She tries replacing it everywhere it is used and this is a huge task! Which design principle could Allison have used to avoid this situation?

1 point

- ◉ Composing Objects Principle
- ○ Principle of Least Knowledge
- ○ Don't Repeat Yourself
- ○ Dependency Inversion

Incorrect

Incorrect. It is not clear how this would have helped her in this case.

24. Which of the code smells is shown in this code example of a method declaration?

1 / 1 point

```
private void anOperation(String colour, int x, int y, int z, int speed)
```

- ◉ Large Parameter List
- ○ Primitive Obsession
- ○ Message Chains
- ○ Long Method

Correct

Correct! A long parameter list like this is often an indication that you should define an abstract data type to contain this bundle of information.

25. Which object-oriented design principle do Long Message Chains, a code smell, usually violate?

1 / 1 point

- (●) Principle of Least Knowledge / Law of Demeter
- ( ) Open/Closed Principle
- ( ) Separation of Concerns
- ( ) Cohesion

Correct
Correct! A class should only know about a few other classes. Long message chains will make your code rigid and difficult to change.

26. Which code smell can you detect here?

1 / 1 point

```
public class Person {
    int age;
    int height;
    String hairColour;

    public int getAge() { return age; }
    ...

}
```

- ( ) Feature Envy
- ( ) Primitive Obsession
- (●) Data Class
- ( ) Data Clump

Correct
Correct! This class seems to only contain data and a getter (with presumably more getters and setters). Maybe there are some operations you could move into this class.

27. What are the components of the MVC pattern?

1 / 1 point

- ( ) Member, Vision, Controller
- ( ) Model, View, Command
- ( ) Model, Vision, Command
- (●) Model, View, Controller

Correct
Correct! Model View Controller

28. The interface segregation principle encourages you to use which of these object-oriented design principles? Choose the **2 correct** answers.

1 point

- [ ] decomposition
- [x] generalization

This should not be selected
Incorrect. Interface segregation encourages you to make more, specific interfaces (as opposed to general superclasses)

- [x] abstraction

Correct
Correct! The principle encourages you to select good abstractions for your entity.

- [ ] encapsulation

29. Interface Segregation is a good way to avoid which code smell? **Choose the best possible answer**.

1 / 1 point

- ( ) Divergent Change
- ( ) Switch Statements
- (•) Refused Bequest
- ( ) Long Method

Correct
Correct! By composing with interfaces instead of inheriting, you can avoid your classes inheriting behaviour that they will not use.

30. Which of these statements about the Decorator pattern are true?

1 point

1. The decorator classes inherit from the basic object which is being decorated
2. Decorator objects can be stacked in different order

- [x] The first statement is true
- [ ] The second statement is true
- [ ] Neither statement is true
- [ ] Both statements are true

Incorrect

Incorrect. This would mean that each decorator inherits the behaviour of the basic object. This is not good separation of concerns!