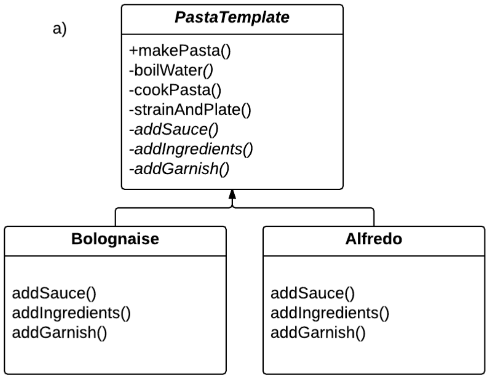
1. Choose the most appropriately implemented Template pattern.
   Some UML reminders that will help you:

1 / 1 point

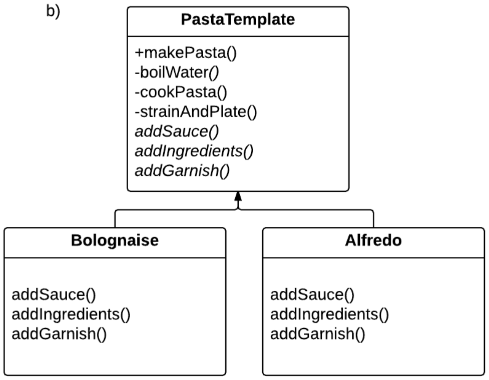
1. a private method or variable is denoted by a - as in -boilWater().

   2. a method, variable, or class that is abstract is denoted by italics (as in PastaTemplate)

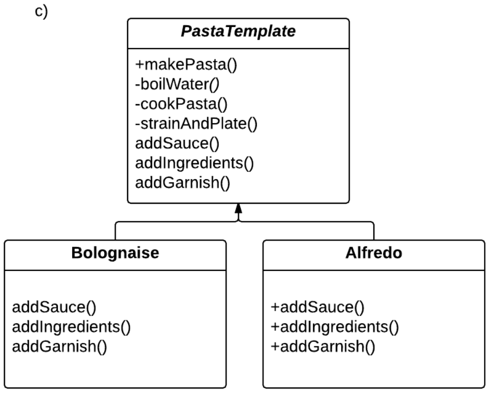
a)


PastaTemplate
+makePasta()
-boilWater*()*
-cookPasta()
-strainAndPlate()
-addSauce()
-addIngredients()
-addGarnish()
Bolognaise
addSauce()
addIngredients()
addGarnish()
Alfredo
addSauce()
addIngredients()
addGarnish()

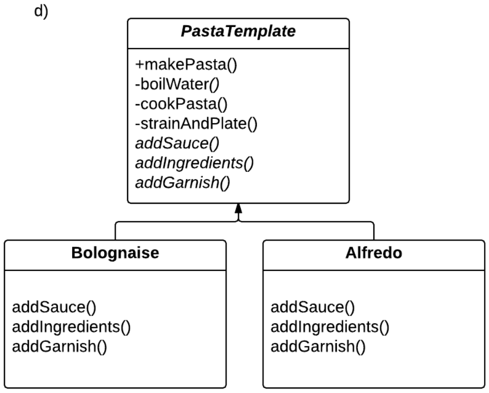
b)


PastaTemplate
+makePasta()
-boilWater*()*
-cookPasta()
-strainAndPlate()
addSauce()
addIngredients()
addGarnish()
Bolognaise
addSauce()
addIngredients()
addGarnish()
Alfredo
addSauce()
addIngredients()
addGarnish()

c)


PastaTemplate
+makePasta()
-boilWater*()*
-cookPasta()
-strainAndPlate()
addSauce()
addIngredients()
addGarnish()
Bolognaise
addSauce()
addIngredients()
addGarnish()
Alfredo
+addSauce()
+addIngredients()
+addGarnish()

d)


PastaTemplate
+makePasta()
-boilWater*()*
-cookPasta()
-strainAndPlate()
addSauce()
addIngredients()
addGarnish()
Bolognaise
addSauce()
addIngredients()
addGarnish()
Alfredo
addSauce()
addIngredients()
addGarnish()

○ a)
○ b)
○ c)
● d)

✓ Correct
Correct! The key aspect of the template pattern is that some parts of the Template are defined by the subclasses. These are the addSauce(), addIngredient() and addGarnish() methods, and they are in italics because they are abstract in the superclass.

2. What is the correct situation for the use of a Chain of Responsibility pattern?

1 / 1 point

○ You need a set of objects to each contribute information on responding to a request.
● You have multiple potential handlers, but only one will deal with the request.
○ You need to pass a message to multiple receivers.
○ You need to delegate a set of tasks to a hierarchy of objects.

✓ Correct
Correct! The handlers pass the message down until one can handle it or the end of the chain is reached.

3. What is the purpose of encapsulating state in an object in the State Pattern? Choose the three that are correct.

1 / 1 point

☑ it turns the context into a client of the state.

✓ Correct
Correct! This allows the context to easily make requests of the state.

☑ it allows the current state object to decide how to achieve behaviours specific to the state of the context.

✓ Correct
Correct! The subclasses of state provide that actual implementation of the behaviours.

□ it allows the current state to be copied from one instance to another
☑ it removes large conditionals that are difficult to maintain.

✓ Correct
Correct! The state pattern outsources those "ifs" to a State object - a subclass of State - which decides how to handle requests.

4. What design principles is the Command Pattern using?

1 / 1 point

○ Generalization, information hiding, loose coupling
○ Encapsulation, information hiding, loose coupling
● Encapsulation, generalization, loose coupling
○ Encapsulation, generalization, information hiding

✓ Correct
Correct! The command pattern encapsulates a request as an object, provides a general command interface for managing command objects, and allows you to have looser coupling between the participants.

5. Which are the minimum requirements of the Observer pattern? Choose the three that are correct.

1 / 1 point

☑ update method in observers

✓ Correct
Correct! When the observers are informed that a change has been made, they update themselves accordingly.

☑ method to notify observers

✓ Correct
Correct! The subject has a method for notifying the observers that a change has been made

□ a state variable to determine if observers have been notified.
☑ methods to add or remove observers

✓ Correct

Correct! There must be a way to track which observers are associated with a subject.

6. When are you most likely to need a Mediator pattern? 1 / 1 point

○ When your class is sending a request that might be handled by one of several handlers.
○ When you have two classes with different interfaces that you must connect.
○ When you want to de-couple a class that is requesting a service from one that is providing it.
● When you are coordinating the activities of a set of related classes.

✓ Correct
Correct! Use a Mediator pattern to coordinate the activities of many, relatively simple classes

7. 1 / 1 point

Marlon is coding part of the software that follows a similar sequence of steps. Depending on the type of object, these steps will be implemented in slightly different ways, but their order is always the same. Which design pattern could Marlon use?

● Template pattern
○ Mediator pattern
○ Command pattern
○ State pattern

✓ Correct

Correct! The Template pattern specifies a general 'recipe' and some common steps in the superclass, but allows the implementation of many of the steps in that recipe to the subclasses.

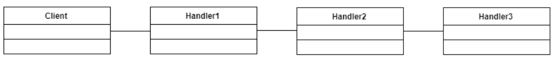
8. What are the important roles in the Command Pattern? 1 / 1 point

○ Command, Queue, Receiver
● Command, Receiver, Invoker
○ Delegate, Command, Requester
○ Sender, Receiver, Invoker

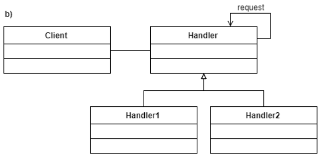
✓ Correct
Correct! These are the three roles that must be implemented in a Command pattern.

9. Select the best UML class diagram representation of the Chain of Responsibility pattern. 1 / 1 point

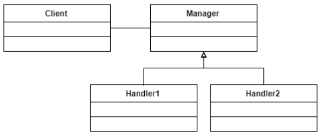
a)



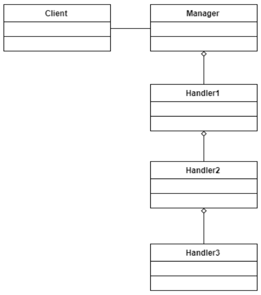
b)



c)



d)



○ a)
● b)
○ c)
○ d)

✓ Correct
Correct! The Handler superclass manages the logic for determining where to send the request next. Each subclass tried to handle the request.

10. You have a machine performing a complex manufacturing task, with different sensors and different components of the machine represented by different classes. Which design pattern will you use to arrange the parts?

1 / 1 point

○ Template
○ Command
● Mediator
○ Chain of Responsibility

✓ Correct
Correct! The Mediator pattern is useful when coordinating the activities of many interrelated classes.

11. You have a security system class, and it has 3 modes: normal, lockdown, and open. Which pattern would you use to model the behaviour in these different modes?

1 / 1 point

● State
○ Template
○ Observer
○ Mediator

✓ Correct
Correct! This pattern is useful when a class has a collection of behaviours that are a little bit different while the system is in different states.

12. One of your classes represents a mailbox, while another is the owner of the mailbox. The person would like to know when new

1 / 1 point

mail arrives. Which design pattern will you probably use?

○ State
○ Command
○ Mediator
● Observer

✓ Correct
Correct! Observer pattern is like subscribing. The Owner is alerted when the mailbox has new mail.