Multiple-Choice Questions:

1. Which of the following terms does this text best define?

   The extent to which one component depends on other components

   a. Cohesion

   b. Concern

   c. Coupling

   d. Crossover

   e. None of the above

2. Given classes *A* and *B*, which of the following is <u>not</u> a common type of coupling in object-oriented software?

   a. *A* is a direct or an indirect subclass of *B*

   b. A method parameter or local variable in *A* references *B*

   c. *A* has an instance variable that refers to *B*

   d. *A* invokes methods of *B*

   e. None of the above

3. All else being equal, which is more desirable?

   a. Higher/tighter coupling

   b. Lower/looser coupling

   c. None of the above is more desirable than the others

4. Class `Gear` has which of the following dependencies (i.e., things that if changed force a change in class `Gear`)?

```
class Gear
…
  def gear_inches
    ratio * Wheel.new(rim, tire).diameter
  end
…
end
```

   a. A class named `Wheel` must exist

   b. `Wheel.new` must take two parameters, `rim` and `tire`

   c. The first argument for `Wheel.new` must be `rim`, and the second must be `tire`

   d. All of the above

   e. None of the above

5. Which of the following is <u>true</u> about design patterns?

   a. Represent the best practices used by experienced object-oriented software developers

   b. Solutions to general problems that developers commonly face during software development

   c. Obtained by trial and error of numerous software developers over a substantial period of time

   d. All of the above

   e. None of the above

6. Which pattern automatically notifies dependent objects when a subject object is modified?

   a. Adapter

   b. Observer

   c. Mediator

   d. Memento

   e. None of the above

7. Which pattern encapsulates how a set of objects interact?

    a. Adapter

    b. Observer

    c. Mediator

    d. Memento

    e. None of the above

8. Which of the following are true about the Mediator Pattern? Circle all that apply.

    a. Reduces interdependencies by spreading interaction logic throughout objects

    b. Promotes loose coupling by keeping objects from referring to each other explicitly

    c. Allows you to vary the interaction between objects independently

    d. Tightly couples objects together to make them more maintainable

    e. Uses indirection to keep objects from directly referring to each other

Solutions:

1. c
2. e
3. b
4. d
5. d
6. b
7. c
8. b, c, e

Consider these figures when answering the following question.

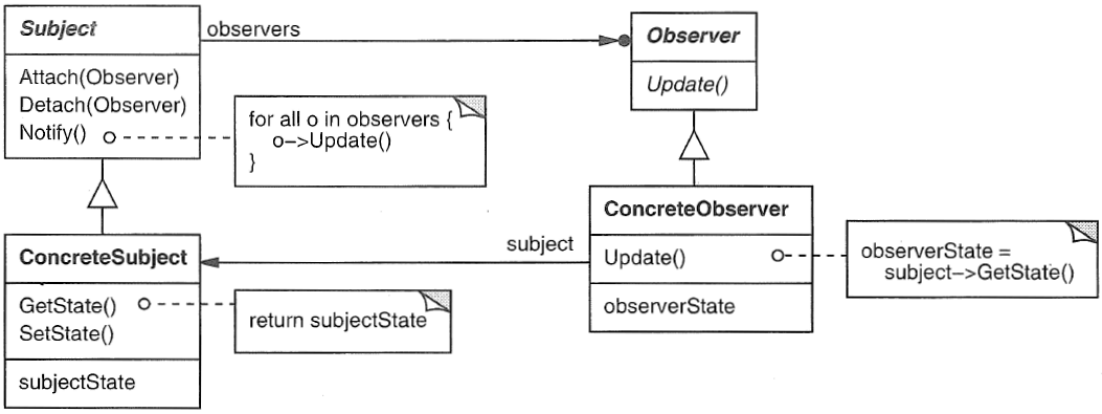


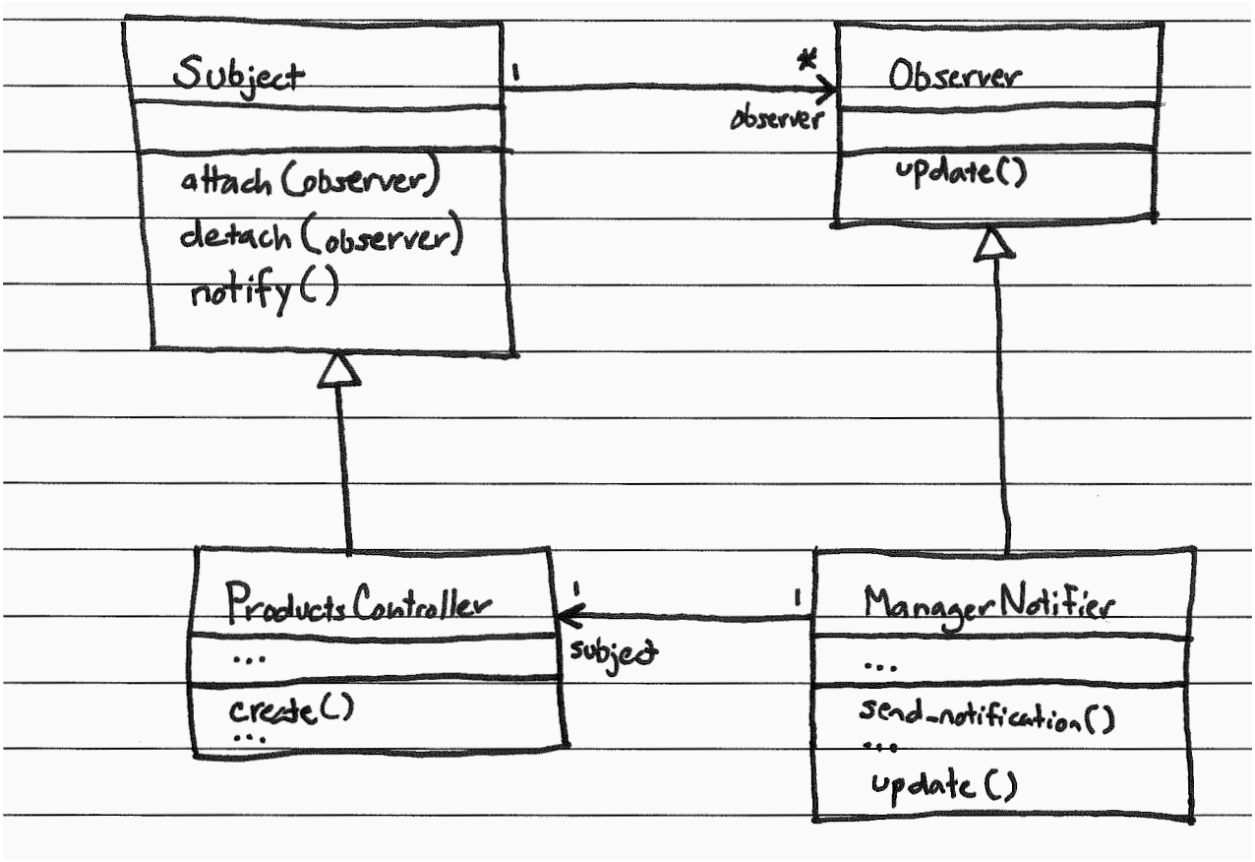
Figure 1. Observer Pattern from the "Gang of Four" book. (Note that the book uses an outdated class diagram notation.)

Figure 2. Classes for product-supply system.

Problem:

Recall the Observer Design Pattern depicted in Figure 1. Imagine that you are designing a web app for product-supply business. Figure 2 depicts the classes that you have so far. In particular, you have designed a ProductsController class that records product information. As part of this controller's responsibilities, it must create new product entries when new products are added. You have also designed a ManagerNotifier that is capable of sending notification messages to managers at the company. The design problem you need to solve is how to make a ManagerNotifier "listen" for when a ProductsController creates a new product, and to send a notification to a manager whenever that happens. Draw a class diagram that applies the Observer Design Pattern to solve this problem. Use the same names used in the design pattern as much as possible (except make Ruby style). You must include all the classes from Figure 2 in your diagram (i.e., your changes should be additive). In particular, I expect that you will be adding classes, operations, inheritance relationships, and associations.

Solution:


Subject
attach(observer)
detach(observer)
notify()
1
*
observer
Observer
update()
Products Controller
...
create()
...
1
1
subject
Manager Notifier
...
send_notification()
...
update()

Consider these figures when answering the following question.

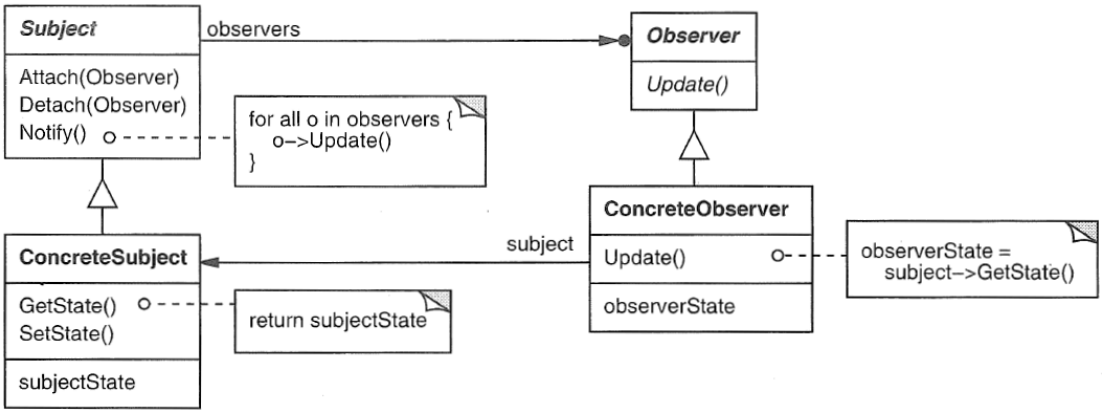


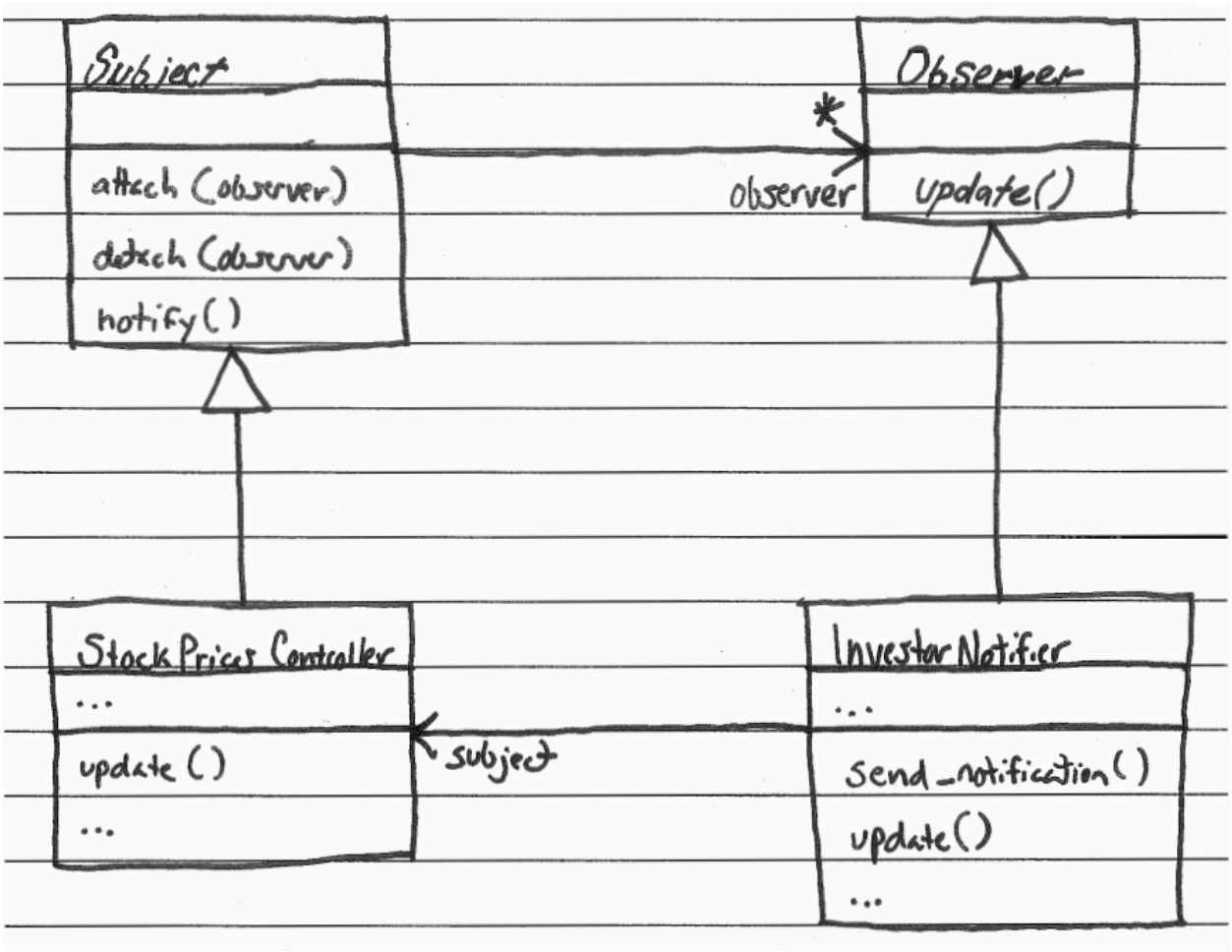
Figure 3. Observer Pattern from the "Gang of Four" book. (Note that the book uses an outdated class diagram notation.)

Figure 4. Classes for investment company web app.

Problem:

Recall the Observer Design Pattern depicted in Figure 3. Imagine that you are designing a web app for an investment company. Figure 4 depicts the classes that you have so far. In particular, you have designed a StockPricesController class that records price changes to stocks. As part of this controller's responsibilities, it must update stock price entries as they change. You have also designed a InvestorNotifier that is capable of sending notification messages to investors. The design problem you need to solve is how to make a InvestorNotifier "listen" for when a StockPricesController updates a stock price, and to send a notification to affected investors whenever that happens. Draw a class diagram that applies the Observer Design Pattern to solve this problem. Use the same names used in the design pattern as much as possible (except make Ruby style). You must include all the classes from Figure 4 in your diagram (i.e., your changes should be additive). In particular, I expect that you will be adding classes, operations, inheritance relationships, and associations.

Solution:


Subject
attach (observer)
detach (observer)
notify()
Observer
*
observer
update()
StockPrices Controller
...
update()
...
subject
InvestorNotifier
...
send_notification()
update()
...

In answering the next question, consider this application of the Observer Pattern. In the application, there is a products controller that can create new products in the system. Manager notifiers observe the products controller, and send notifications to managers when new products are created.

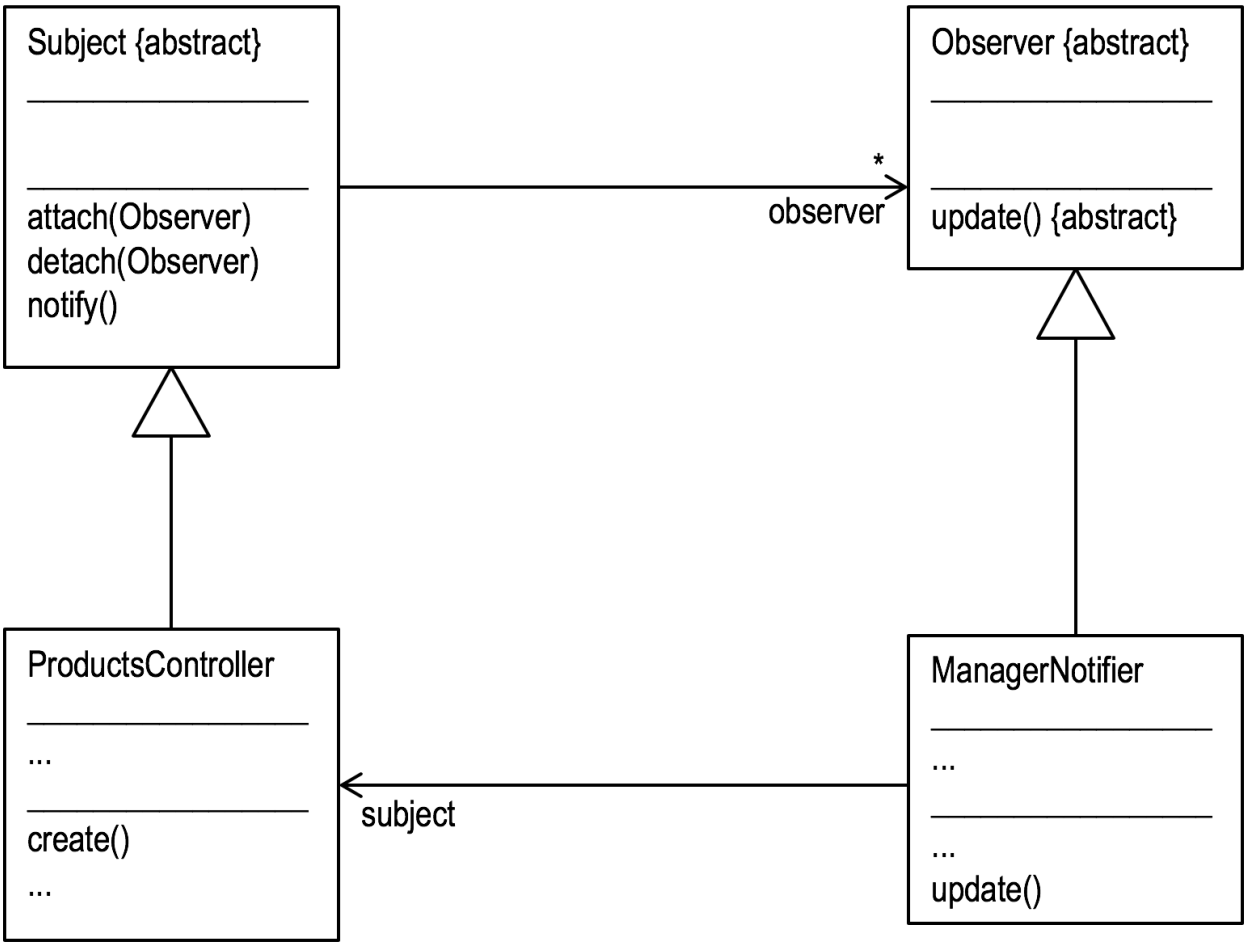
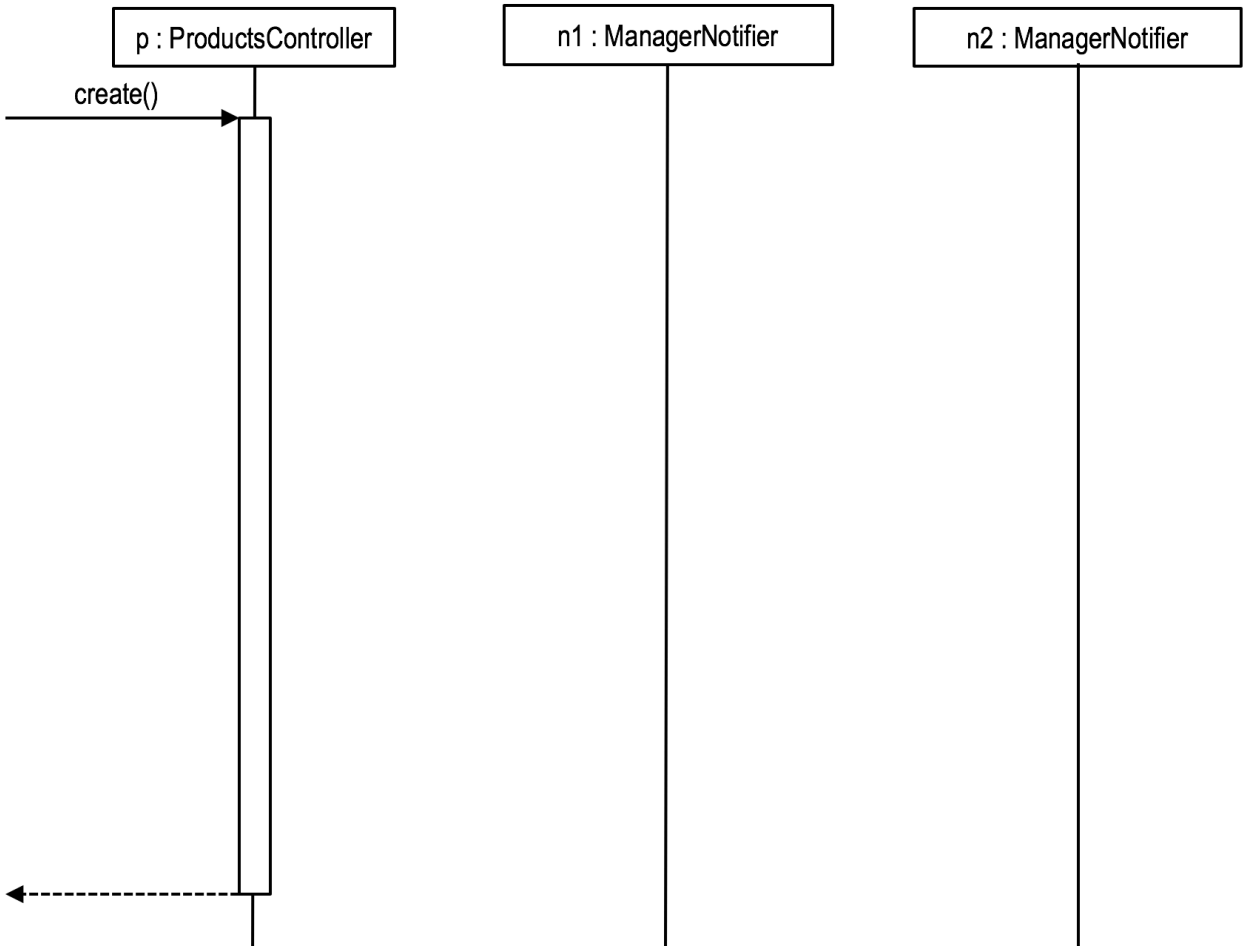


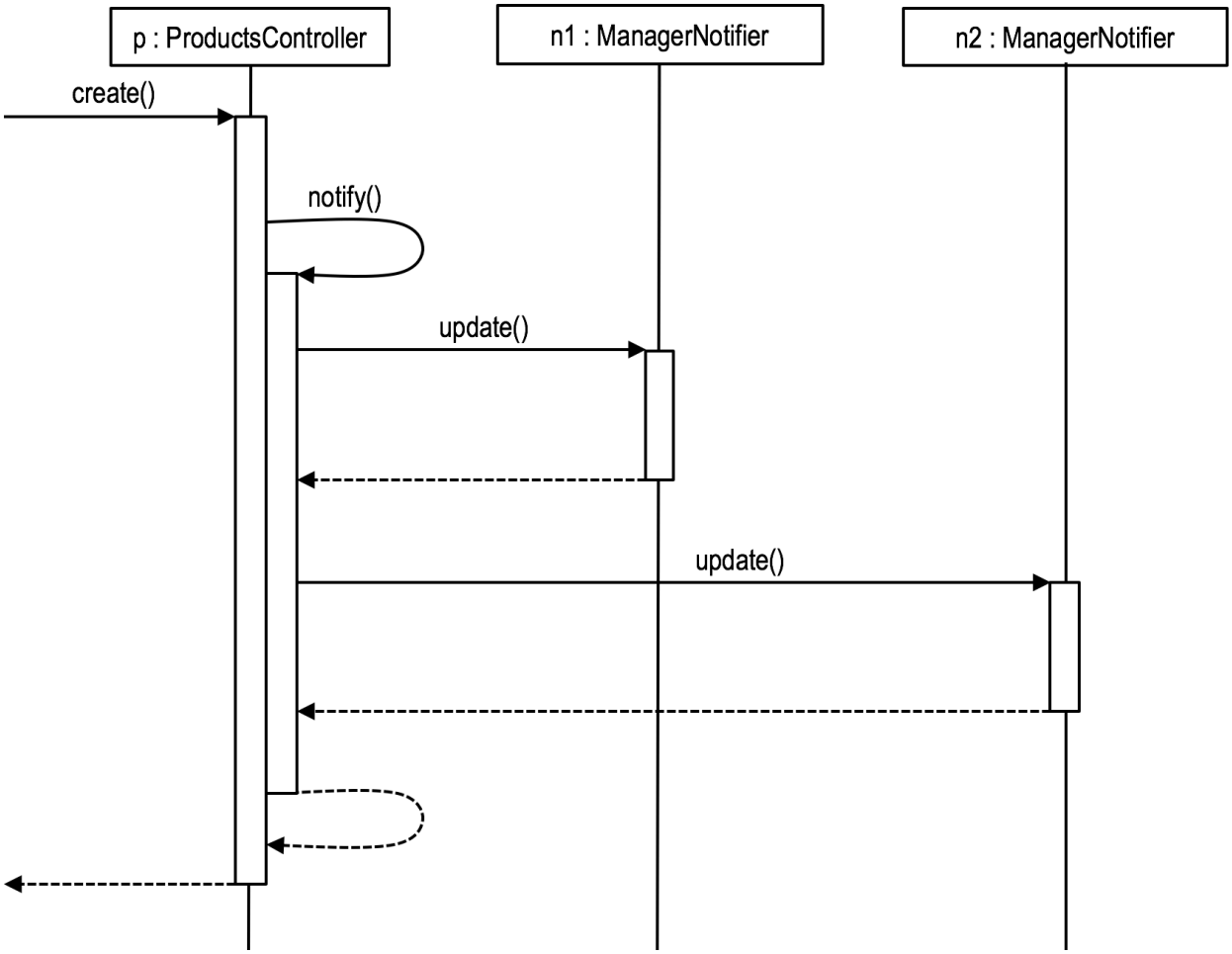
Figure 5. Application of Observer Pattern for a product management application that notifies managers whenever a new product is created in the system.

Problem:

The partially completed sequence diagram below depicts a ProductsController object (*p*) and two ManagerNotifier objects (*n1* and *n2*). The ManagerNotifier objects are already attached to the ProductsController object (although it is not depicted explicitly in the sequence diagram). Complete the sequence diagram such that, as per the Observer Pattern, it shows the method calls and returns triggered by the products controller creating a new product. Show only calls to methods that are depicted in the class diagram.



Solution:


p : ProductsController
n1 : ManagerNotifier
n2 : ManagerNotifier
create()
notify()
update()
update()

Part 2

1. Which of the following best defines *software design pattern*?

    a) Application programming interface for a code library

    b) General, reusable solution to a commonly occurring design problem

    c) Reusable code library for a particular problem domain

2. List two key benefits that design patterns provide. (Three benefits were discussed in class.)

3. Which book popularized software design patterns?

    a) *Pattern-Oriented Software Architecture* (aka the "POSA" book)

    b) *Design Patterns: Elements of Reusable Object-Oriented Software* (aka the "GoF" book)

    c) *Head First Design Patterns* (aka the "HFDP" book)

4. How is the structure of a design pattern typically expressed?

    a) Component diagram

    b) Dataflow diagram

    c) Class diagram

Consider the Observer design pattern diagram in Figure 6.

5. Which of the following best characterizes a `Subject` in the pattern?

    a) Object that listens for changes to another object and reacts to those changes

    b) Object that encapsulates both subscriber and publisher objects

    c) Object that is being watched for state changes

6. Which of the following best characterizes an `Observer` in the pattern?

    a) Object that listens for changes to another object and reacts to those changes

    b) Object that encapsulates both subscriber and publisher objects

    c) Object that is being watched for state changes

7. Which of the following best describes the purpose of the `Subject#attach` operation?

    a) Operation that a subject object invokes when the subject changes state and that in turn invokes an operation on each of the associated observer objects

    b) Operation that adds an observer object the list of such objects maintained by a subject

    c) Observer-specific operation that defines how a particular observer object reacts to changes in a watched subject object

8. Which of the following best describes the purpose of the `Subject#notify` operation?

    a) Operation that a subject object invokes when the subject changes state and that in turn invokes an operation on each of the associated observer objects

    b) Operation that adds an observer object the list of such objects maintained by a subject

    c) Observer-specific operation that defines how a particular observer object reacts to changes in a watched subject object

9. Which of the following best describes the purpose of the `Observer#update` operation?

    a) Operation that a subject object invokes when the subject changes state and that in turn invokes an operation on each of the associated observer objects

    b) Operation that adds an observer object the list of such objects maintained by a subject

    c) Observer-specific operation that defines how a particular observer object reacts to changes in a watched subject object

10. Describe in plain English all the changes and additions you would need to make in order to apply the Observer Pattern to solve the design problem in Scenario 1. Such changes/additions may involve classes, attributes, methods, associations, generalization relationships, etc.

11. Which objects play the role of subject in the application of the Observer Pattern?

    a) Class `Article`

    b) Class `Editor`

    c) Class `EditorNotifier`

12. Which objects play the role of observer in the application of the Observer Pattern?

    a) Class `Article`

    b) Class `Editor`

    c) Class `EditorNotifier`

Figures

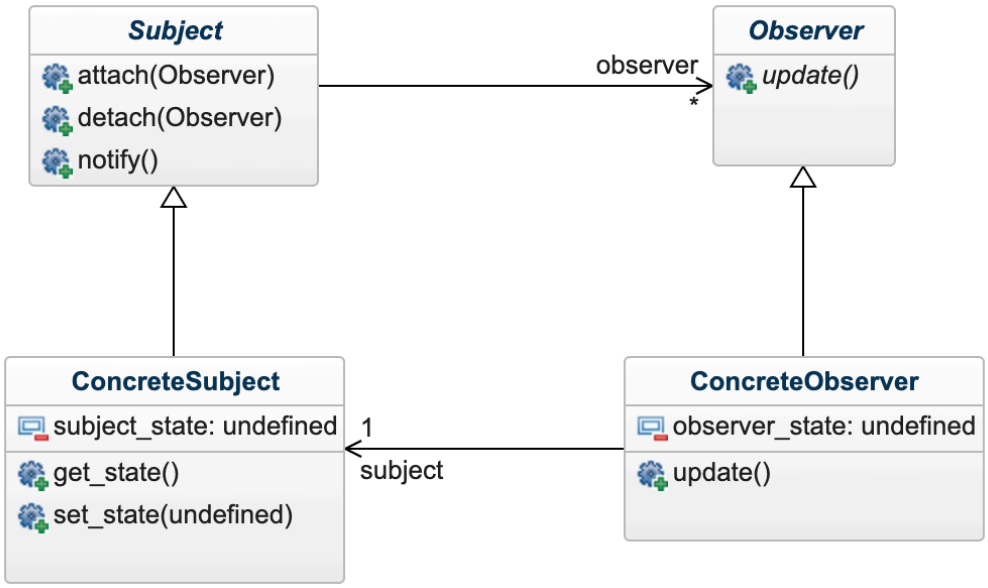
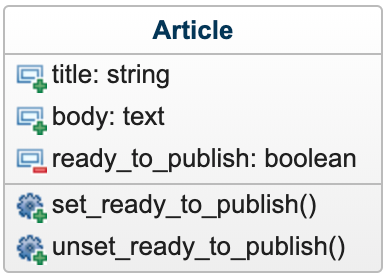


Figure 6. Observer Pattern diagram.

Scenario 1. Designing an online magazine web app.

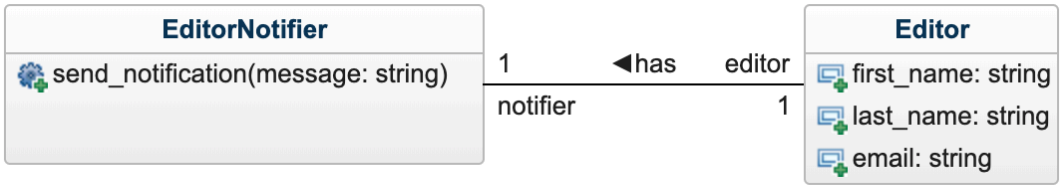
Imagine that you are designing a web app for an online magazine, and you have a design problem that you must solve.

Your app currently has the following model class `Article` that represents a magazine article:



When an article writer first creates an article, the `ready_to_publish` attribute is set to `false`. When the author has completed the article, they use the app to set the attribute to `true`. In particular, the system uses the method `Article#set_ready_to_publish` to set the attribute to `true`.

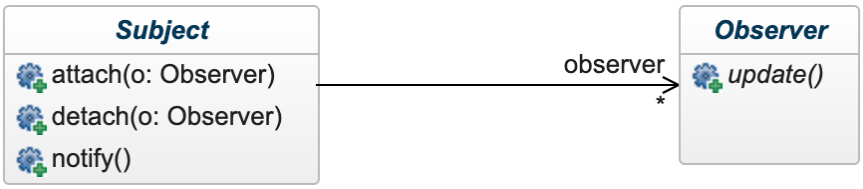
The app also has an editor-notification subsystem:



An editor is the person who reviews an article before it is actually published in the magazine. Each editor is represented by an `Editor` object in the system. Each `Editor` object has an associated `EditorNotifier` object that can be used to send email notifications to the editor by calling the `EditorNotifier#send_notification` method.

Here's the design problem that you must solve: an `EditorNotifier` object must be able to "watch" an `Article` object such that when the `ready_to_publish` attribute is set to `true`, the `EditorNotifier` object will send an email notification to the associated editor.

To solve this problem, you must apply the Observer Pattern. To assist you, a support library for the Observer Pattern has been provided. In particular, the library contains the following:



Part 2 Solutions

1. [2] Which of the following best defines *software design pattern*?

   a) Application programming interface for a code library

   b) General, reusable solution to a commonly occurring design problem

   c) Reusable code library for a particular problem domain

2. [4] List two key benefits that design patterns provide. (Three benefits were discussed in class.)

Benefit 1: Avoid reinventing solutions.

Benefit 2: Promote software quality.

Benefit 3: Facilitate communication.

3. [2] Which book popularized software design patterns?

   a) *Pattern-Oriented Software Architecture* (aka the "POSA" book)

   b) *Design Patterns: Elements of Reusable Object-Oriented Software* (aka the "GoF" book)

   c) *Head First Design Patterns* (aka the "HFDP" book)

4. [2] How is the structure of a design pattern typically expressed?

   a) Component diagram

   b) Dataflow diagram

   c) Class diagram

Consider the Observer design pattern diagram in Figure 1.

5. [2] Which of the following best characterizes a `Subject` in the pattern?

    a) Object that listens for changes to another object and reacts to those changes

    b) Object that encapsulates both subscriber and publisher objects

    c) Object that is being watched for state changes

6. [2] Which of the following best characterizes an `Observer` in the pattern?

    a) Object that listens for changes to another object and reacts to those changes

    b) Object that encapsulates both subscriber and publisher objects

    c) Object that is being watched for state changes

7. [2] Which of the following best describes the purpose of the `Subject#attach` operation?

    a) Operation that a subject object invokes when the subject changes state and that in turn invokes an operation on each of the associated observer objects

    b) Operation that adds an observer object the list of such objects maintained by a subject

    c) Observer-specific operation that defines how a particular observer object reacts to changes in a watched subject object

8. [2] Which of the following best describes the purpose of the `Subject#notify` operation?

    a) Operation that a subject object invokes when the subject changes state and that in turn invokes an operation on each of the associated observer objects

    b) Operation that adds an observer object the list of such objects maintained by a subject

    c) Observer-specific operation that defines how a particular observer object reacts to changes in a watched subject object

9. [2] Which of the following best describes the purpose of the `Observer#update` operation?

    a) Operation that a subject object invokes when the subject changes state and that in turn invokes an operation on each of the associated observer objects

    b) Operation that adds an observer object the list of such objects maintained by a subject

    c) Observer-specific operation that defines how a particular observer object reacts to changes in a watched subject object

10. [6] Describe in plain English all the changes and additions you would need to make in order to apply the Observer Pattern to solve the design problem in Scenario 1. Such changes/additions may involve classes, attributes, methods, associations, generalization relationships, etc.

(1) Add a generalization relationship such that Article implements Subject. (2) Add a generalization relationship such that EditorNotifier implements Observer. (3) Add an update() method to EditorNotifier that invokes send_notification(). (4) Modify set_ready_to_publish() so that it invokes notify(). (5) (optional) Add an association between EditorNotifier and Article.

11. [2] Which objects play the role of subject in the application of the Observer Pattern?

a) Class `Article`

b) Class `Editor`

c) Class `EditorNotifier`

12. [2] Which objects play the role of observer in the application of the Observer Pattern?

a) Class `Article`

b) Class `Editor`

c) Class `EditorNotifier`