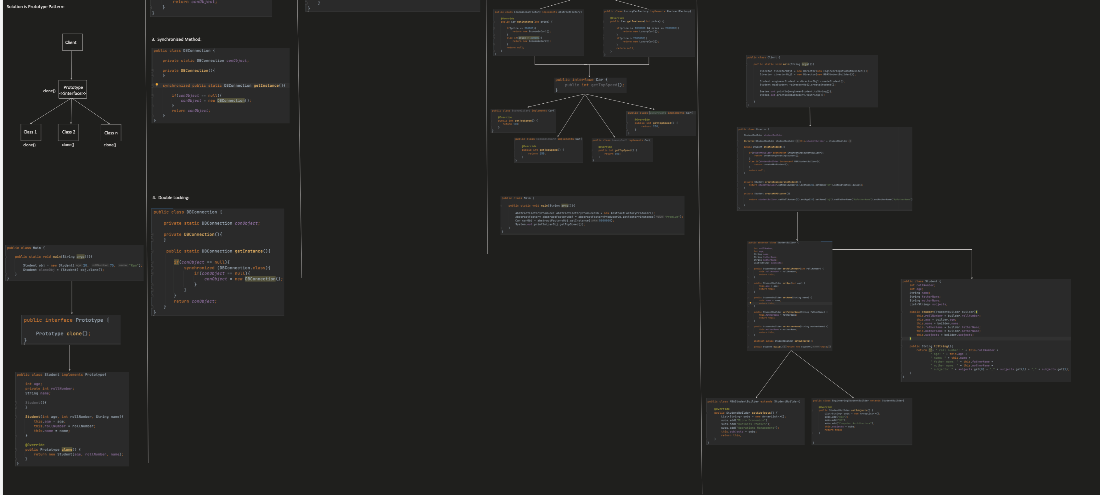
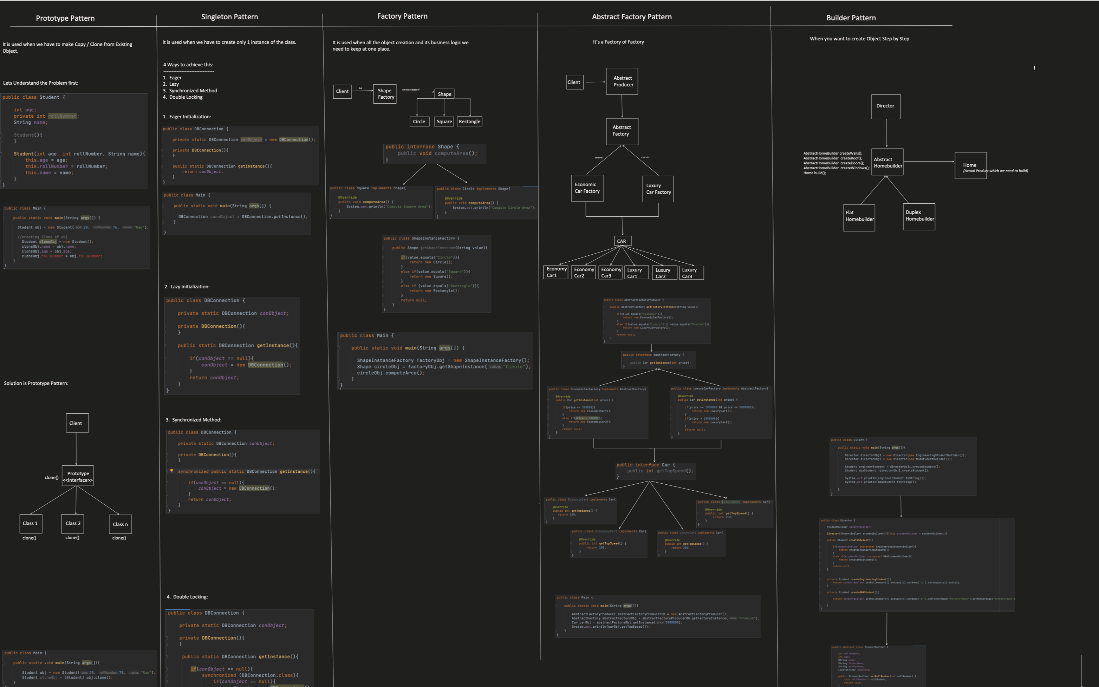
All Creational Patterns

Friday, November 3, 2023

3:57 PM



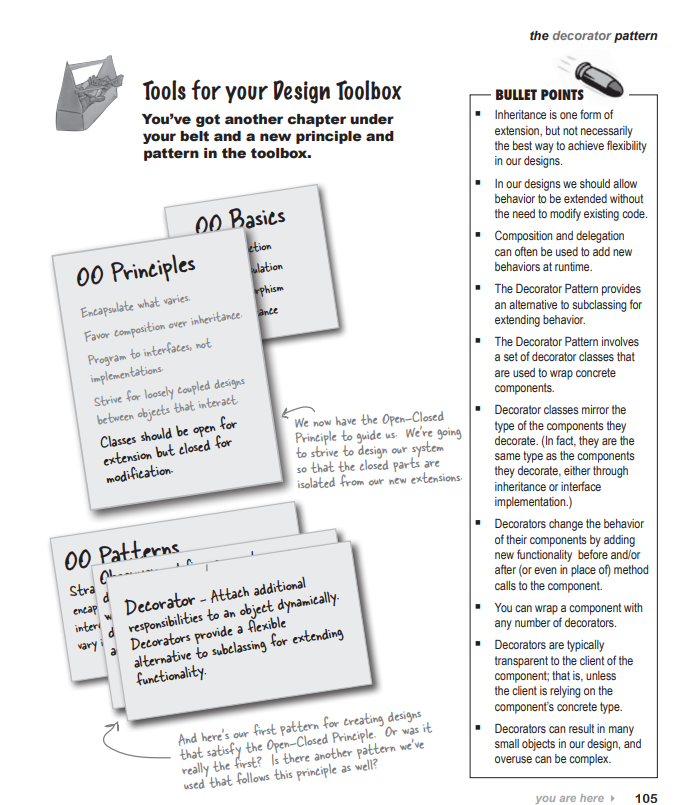


Decorator Pattern

Wednesday, January 3, 2024

11:55 AM

The Decorator Pattern attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.



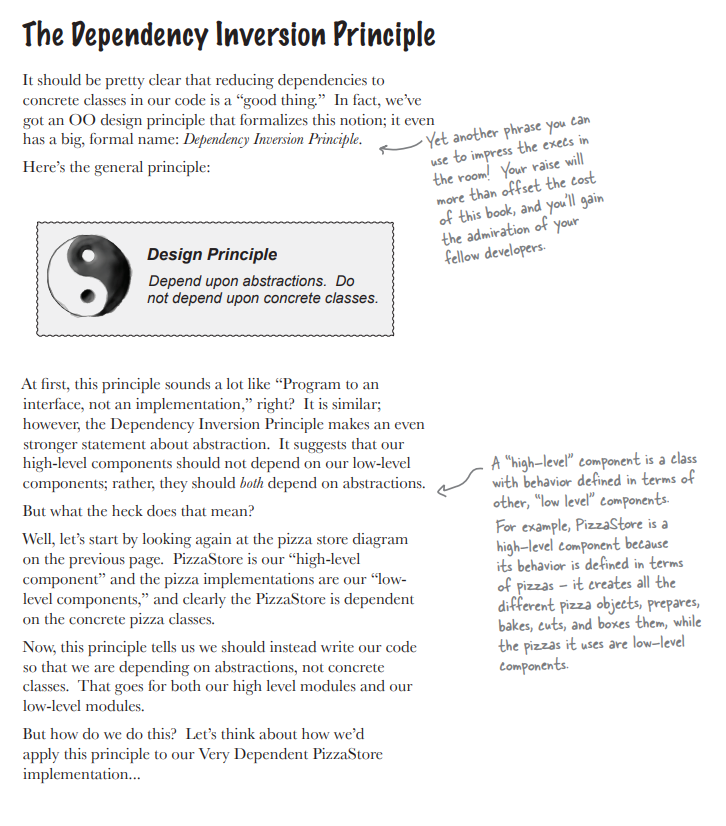
Factory Pattern

Monday, January 8, 2024

4:03 PM

Code to interfaces not implementations

Dependency inversion principle

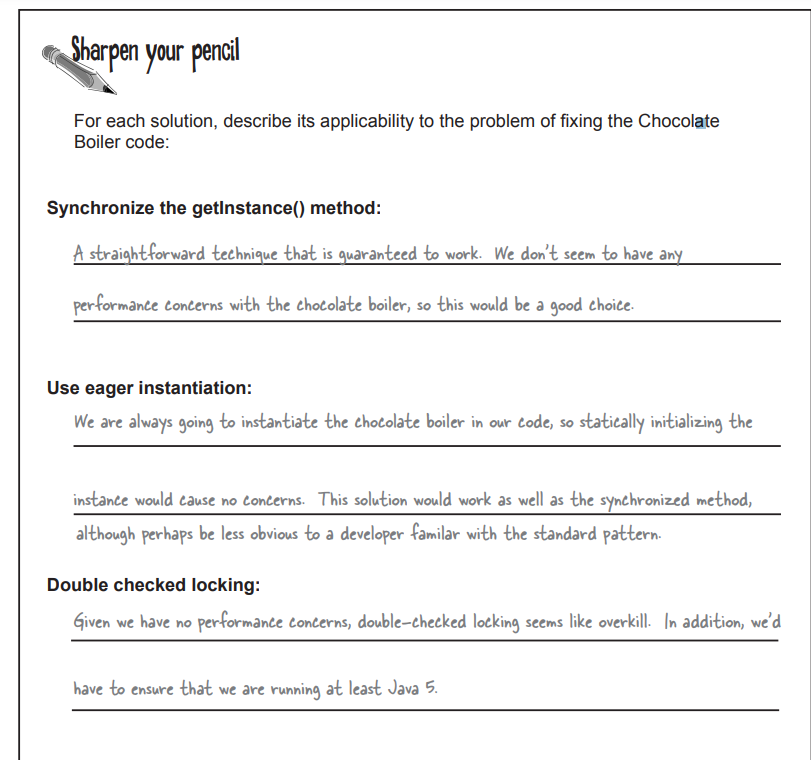


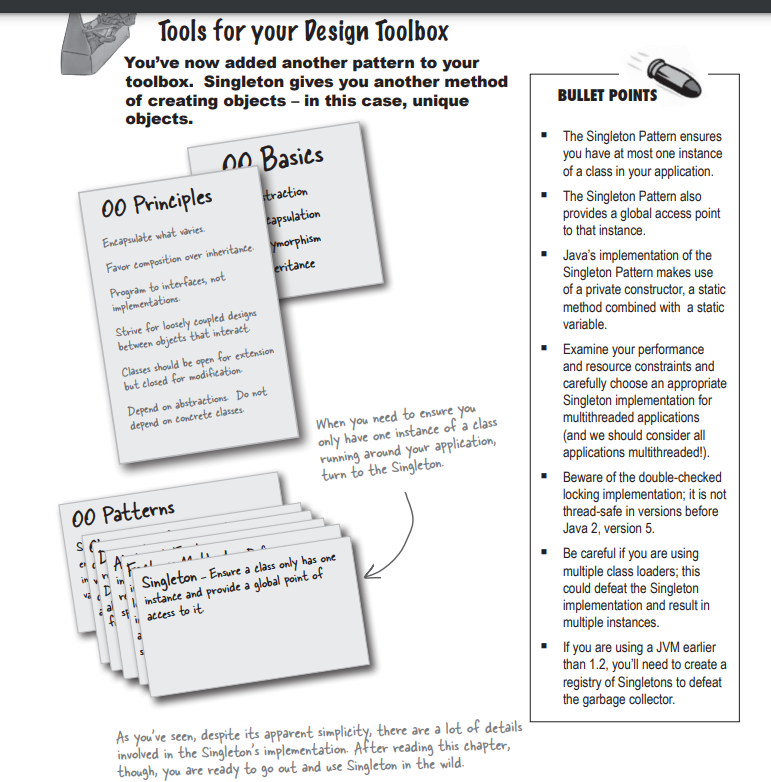
Singleton pattern

Thursday, January 18, 2024

11:53 PM

The Singleton Pattern ensures a class has only one instance, and provides a global point of access to it.





The Command pattern

Monday, February 5, 2024

3:47 PM

Here the requester would be the remote control and the object that performs the action would be an instance of one of your vendor classes.

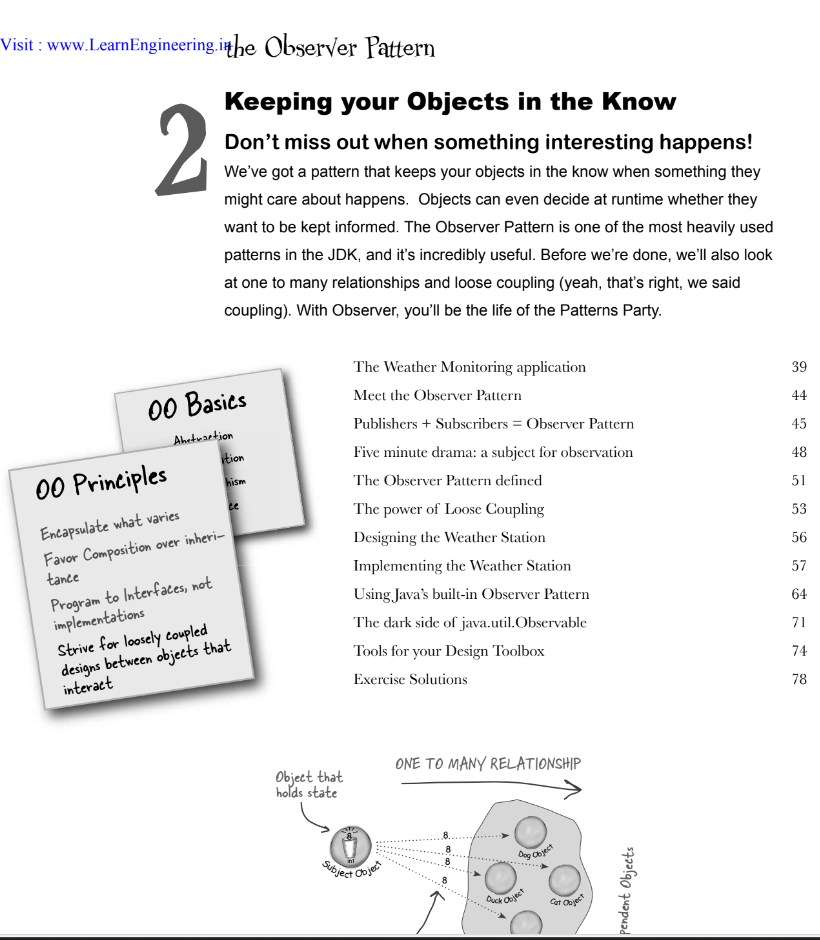
It is possible by introducing command objects into your design, a command object encapsulates a request to do something on a specific object, so if we press the remote it does not have any idea what object will be called I just knows that command object is called on button press, which knows how to talk to right object to get the work done. In this way remote is decoupled from the light object.

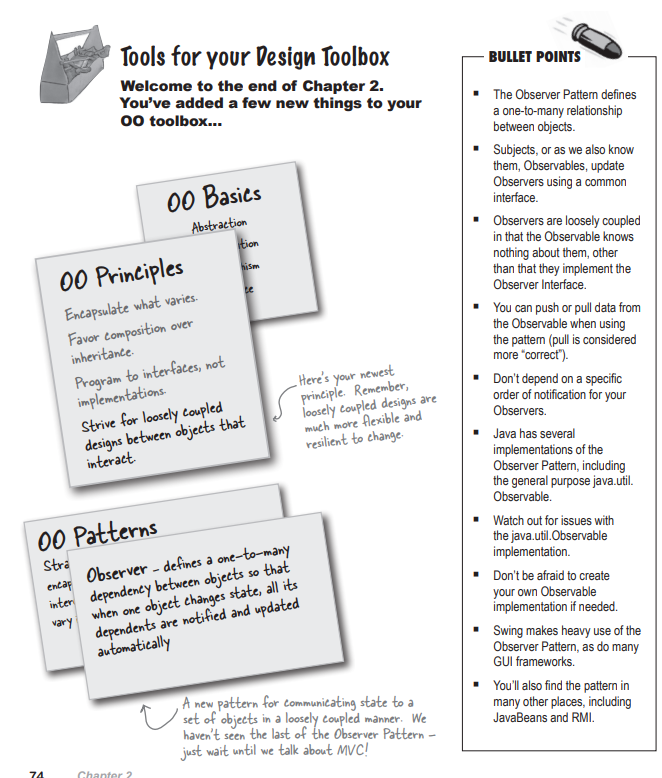
Definition : The Command Pattern encapsulates a request as an object, thereby letting you parameterize other objects with different requests, queue or log requests, and support undoable operations.

Observer Pattern

Monday, May 20, 2024

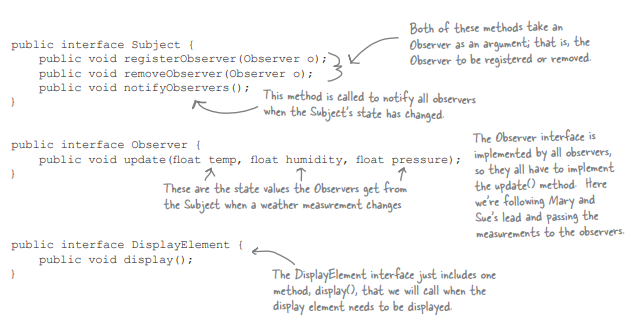
11:01 PM





Here we have a topic also called subject and multiple observer which subscribes to that subjects. Each observer subscribes to the observer and in case of event change all the observers are notified.

Each observer can register and unregister itself.

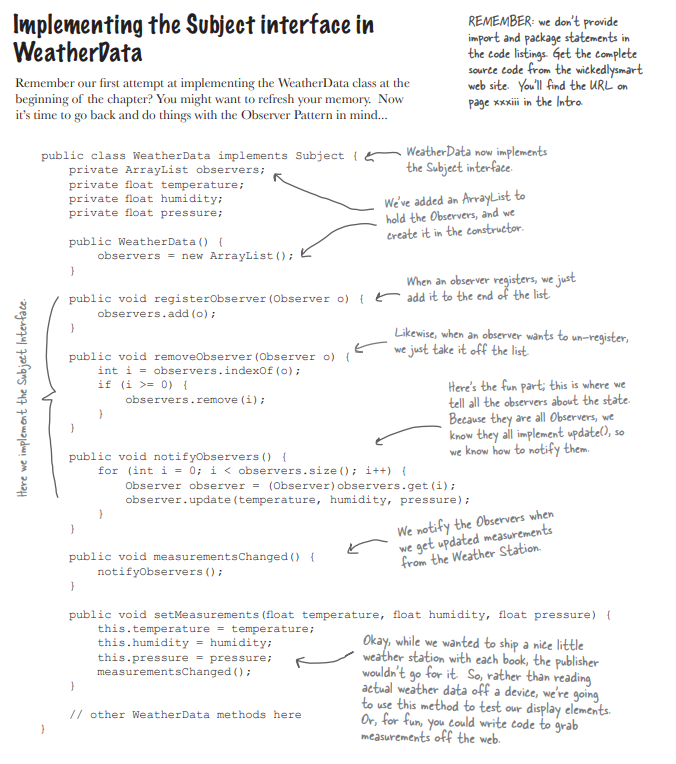


Subject/Observer has methods registerObserver, removeObserver, notifyObserver.

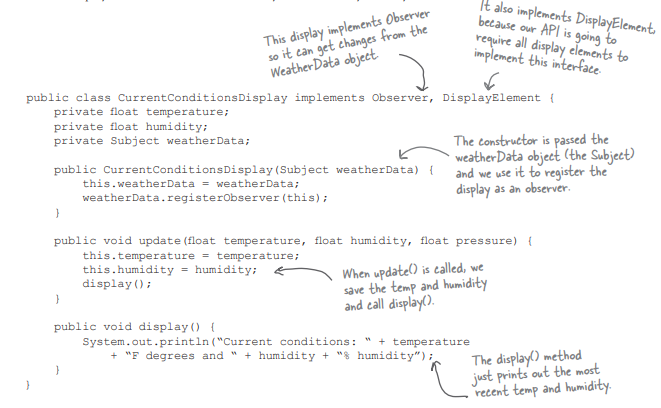
Observer has update method.

DisplayElement has display method.

All the classes which will implement the Subject interface will be called subjects.



Each Observer will implement the observer interface and display interface, we are segregating the display interface as there can be various different display screens.



Java.Util package also consists of Observable and Observer class which has built-in implementation of observer pattern.

Strategy Design Pattern

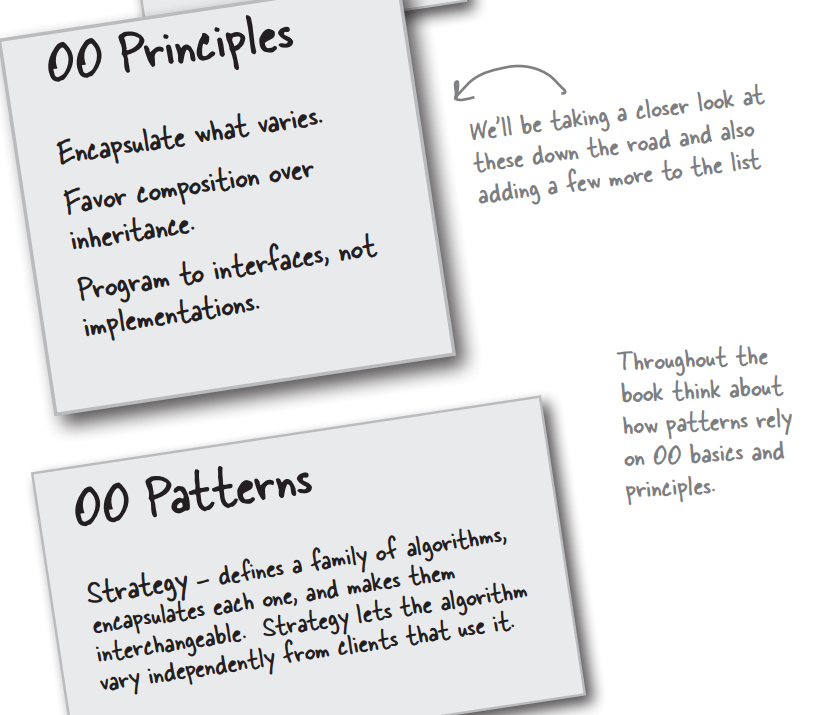
* 2. encapsulate
  3. Composition
  4. .
  5. .
  6. .
  7. Principles
  8. .
  9. .
  10. .
  11. .
  12. .
  13. .
  14. .
  15. Interface
  16. .
  17. Brain
  18. .
  19. .
  20. Vocabulary

Friday, December 29, 2023

12:50 PM

BULLET POINTS Knowing the OO basics does not make you a good OO designer. Good OO designs are reusable, extensible and maintainable. Patterns show you how to build systems with good OO design qualities. Patterns are proven objectoriented experience. Patterns don’t give you code, they give you general solutions to design problems. You apply them to your specific application. Patterns aren’t invented, they are discovered. Most patterns and principles address issues of change in software. Most patterns allow some part of a system to vary independently of all other parts. We often try to take what varies in a system and encapsulate it. Patterns provide a shared language that can maximize the value of your communication with other developers.

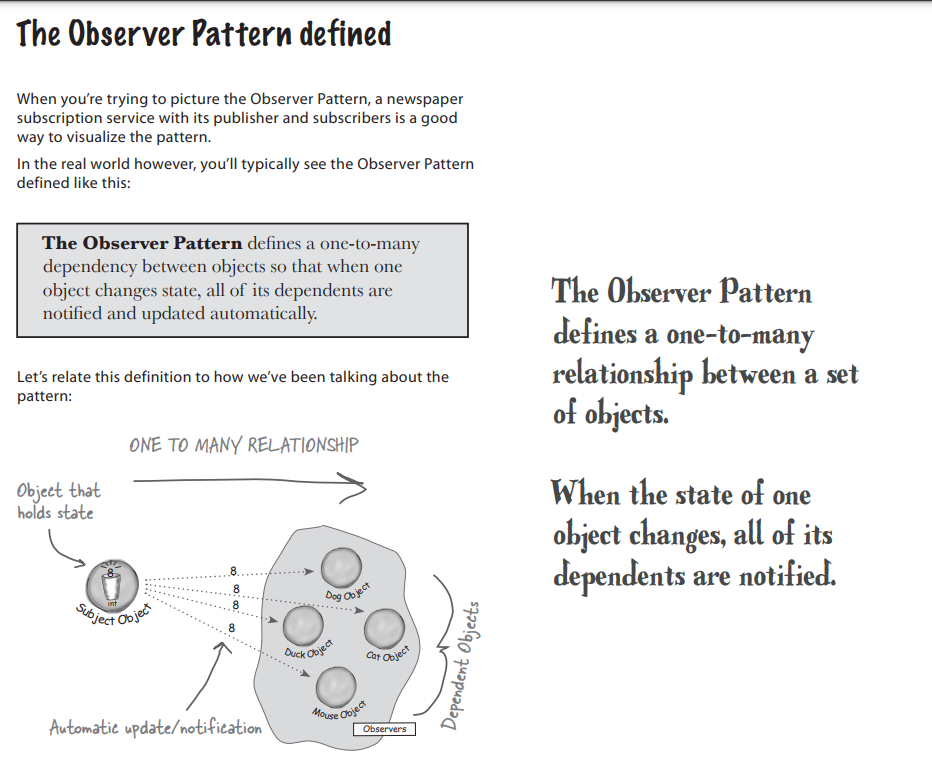
.



Observer Design Pattern

Friday, December 29, 2023

1:24 PM



Ans 1:

We are coding to concrete implementations, not interfaces.

For every new display element we need to alter code.

The display elements don't implement a common interface

We haven't encapsulated the part that changes.

We are voilating encapsulation of the weatherData class.