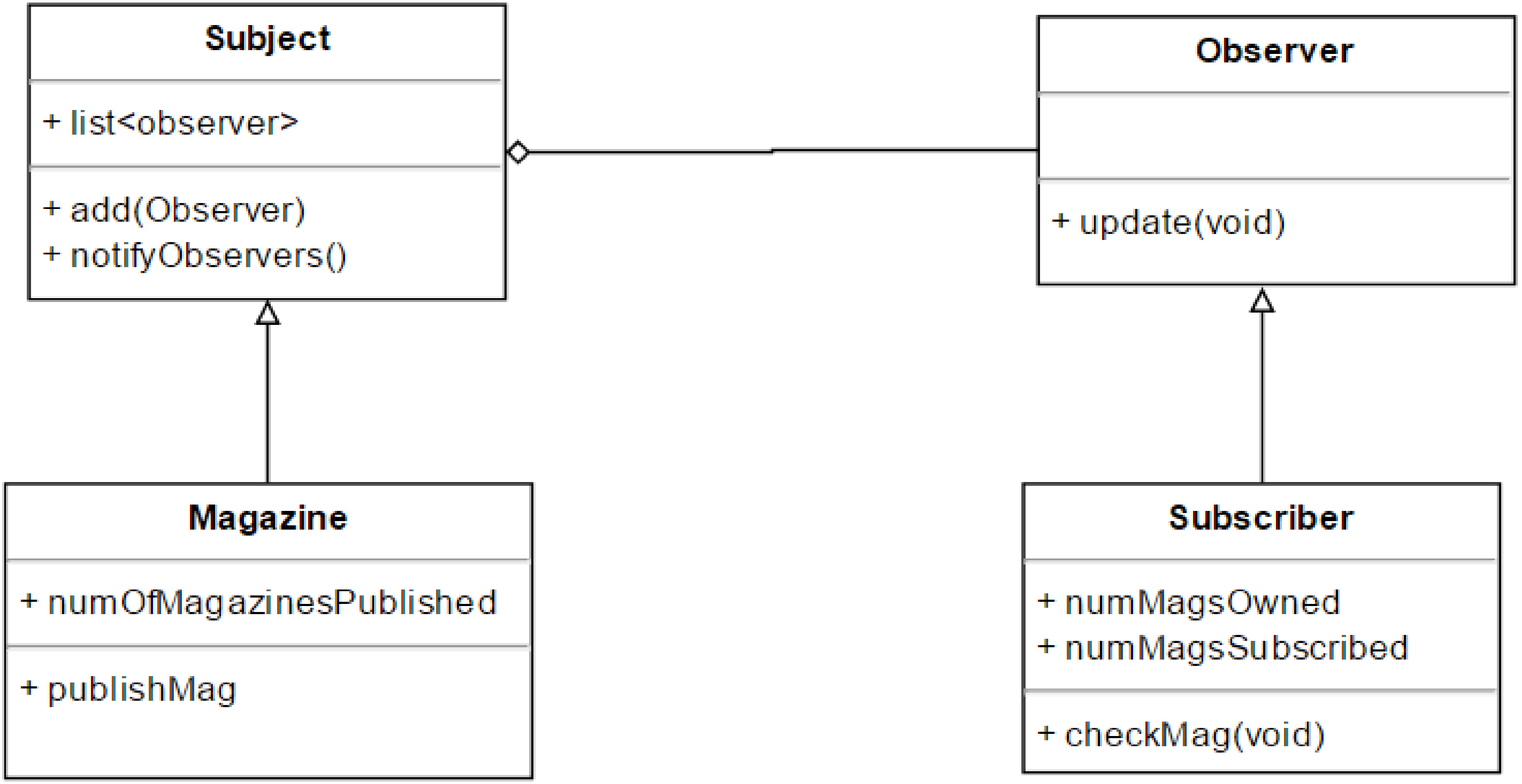
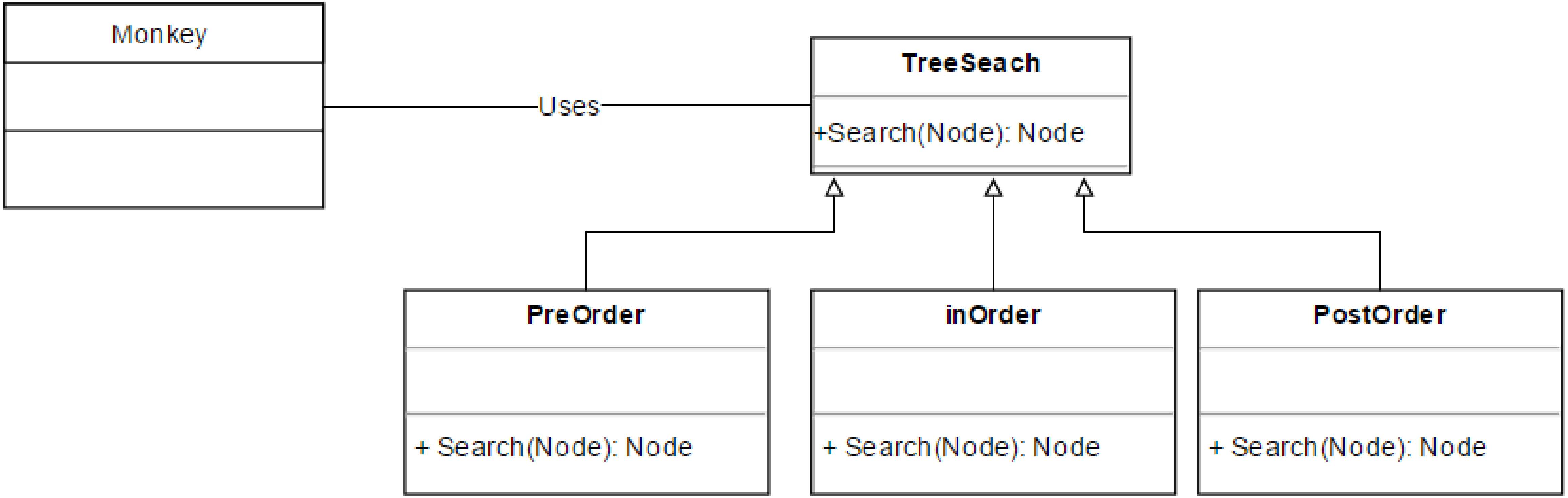
OO Design Patterns

Coding Cleverly

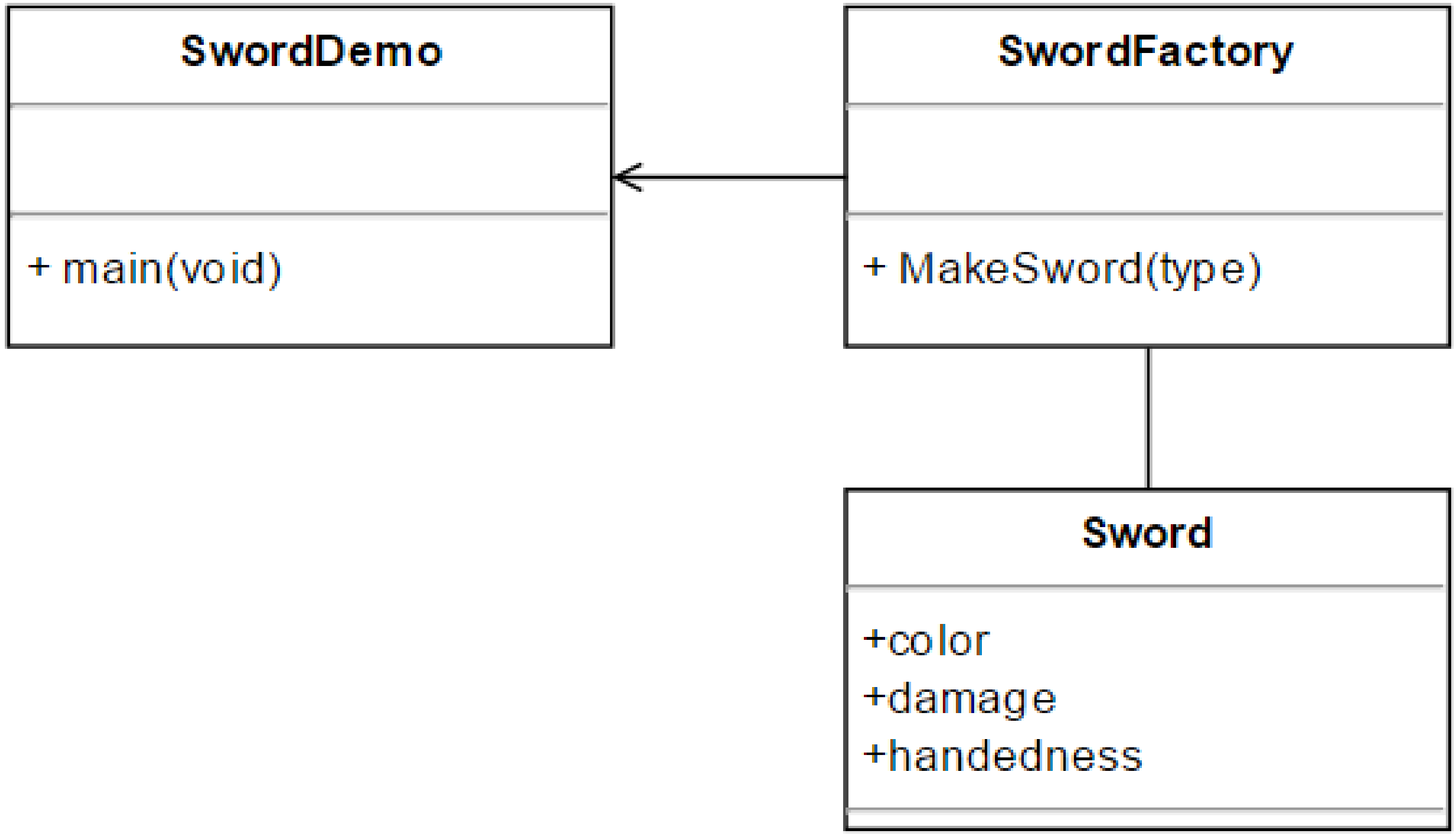
Observer


Subject
+ list<observer>
+ add(Observer)
+ notifyObservers()
Observer
+ update(void)
Magazine
+ numOfMagazinesPublished
+ publishMag
Subscriber
+ numMagsOwned
+ numMagsSubscribed
+ checkMag(void)

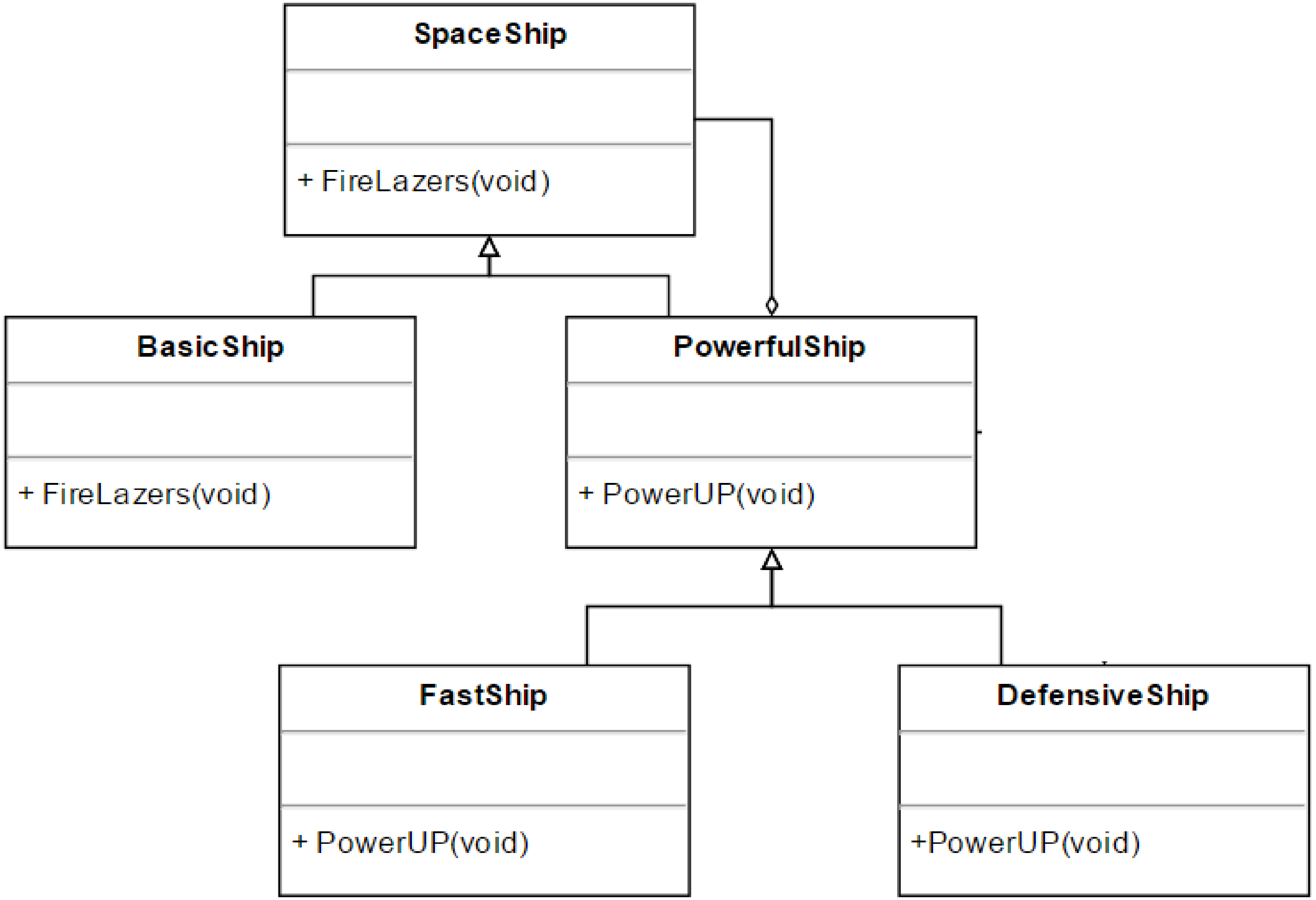
Strategy


Monkey
Uses
TreeSeach
+Search(Node): Node
PreOrder
+ Search(Node): Node
inOrder
+ Search(Node): Node
PostOrder
+ Search(Node): Node

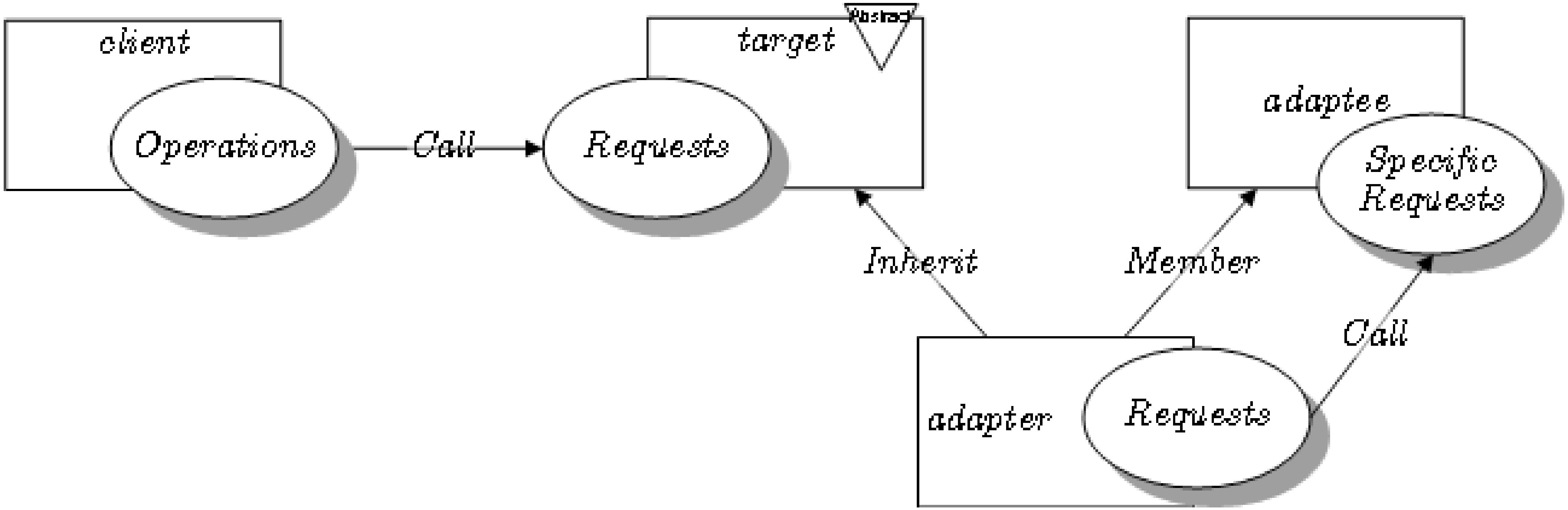
Factory


SwordDemo
+ main(void)
SwordFactory
+ MakeSword(type)
Sword
+color
+damage
+handedness

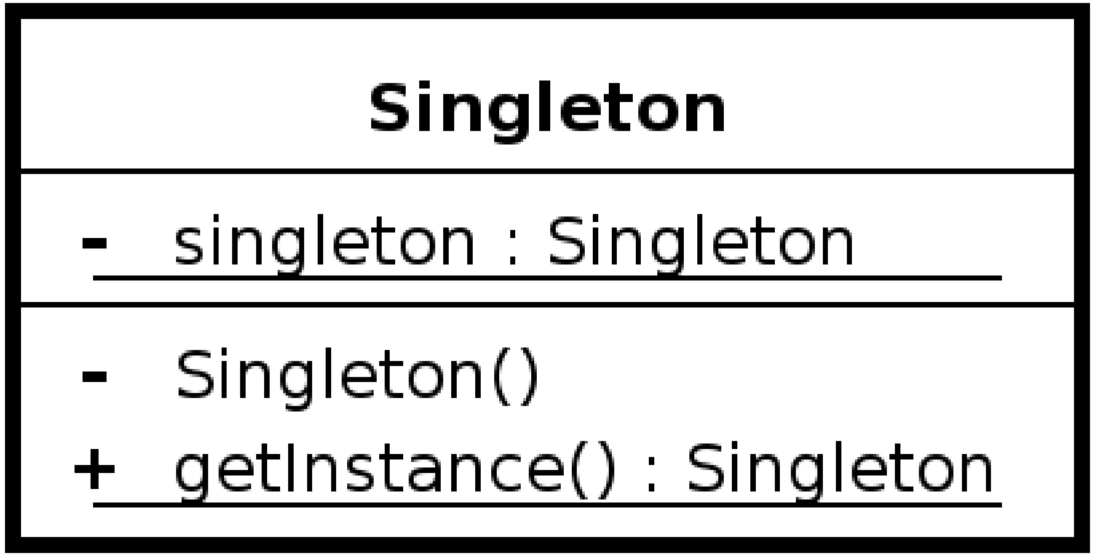
Decorator Pattern



Adapter


client
Operations
Call
Requests
target
Abstract
Inherit
Member
adaptee
Specific Requests
Call
adapter
Requests

Singleton



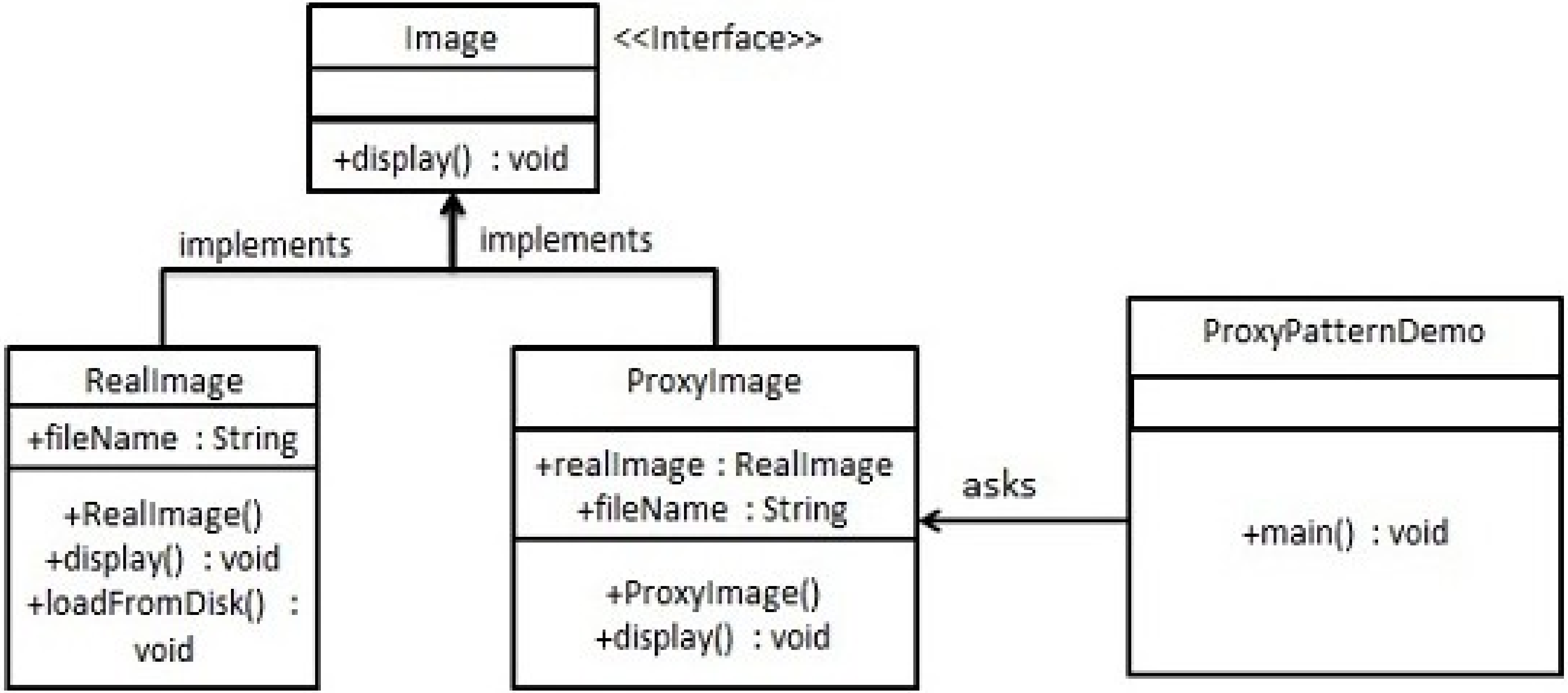
Singleton

- singleton : Singleton

- Singleton()

+ getInstance() : Singleton

Proxy



Image

<<Interface>>

+display() : void

implements

implements

RealImage

+fileName : String

+RealImage()
+display() : void
+loadFromDisk() : void

ProxyImage

+realImage : RealImage
+fileName : String

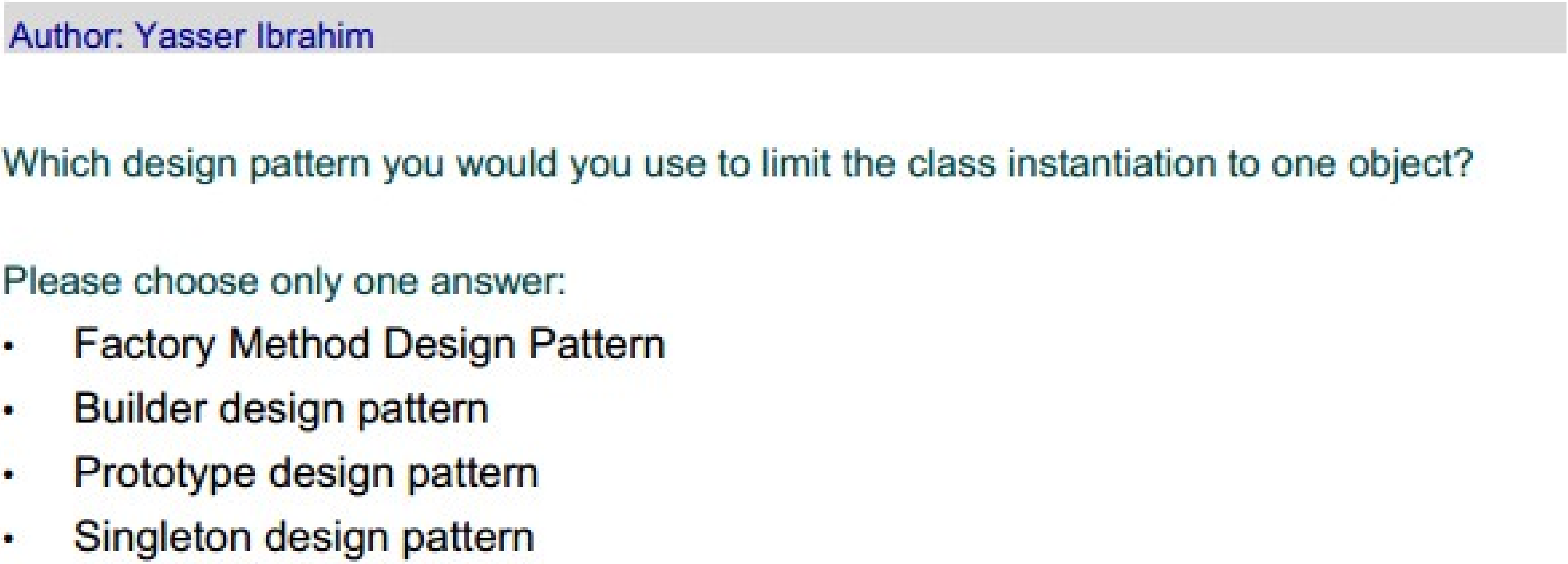
+ProxyImage()
+display() : void

asks

ProxyPatternDemo

+main() : void

Warm UP



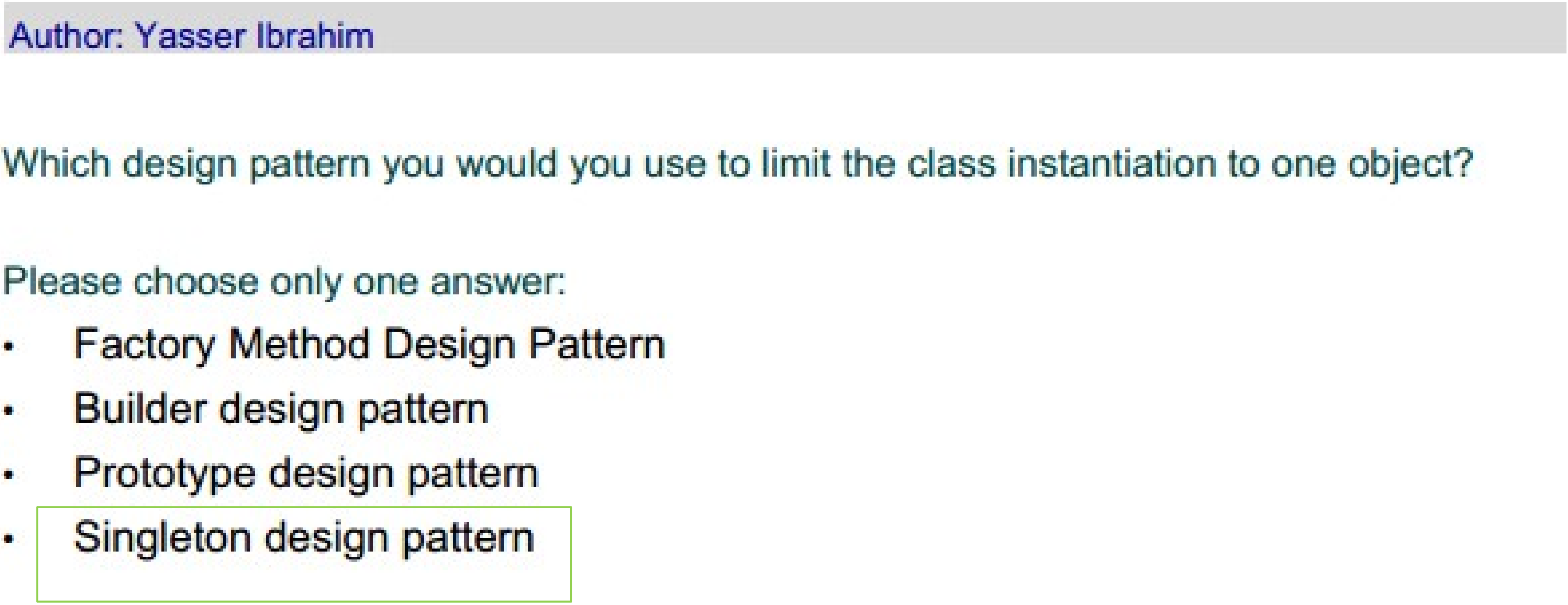
Author: Yasser Ibrahim

Which design pattern you would you use to limit the class instantiation to one object?

Please choose only one answer:

- Factory Method Design Pattern
- Builder design pattern
- Prototype design pattern
- Singleton design pattern

Warm UP



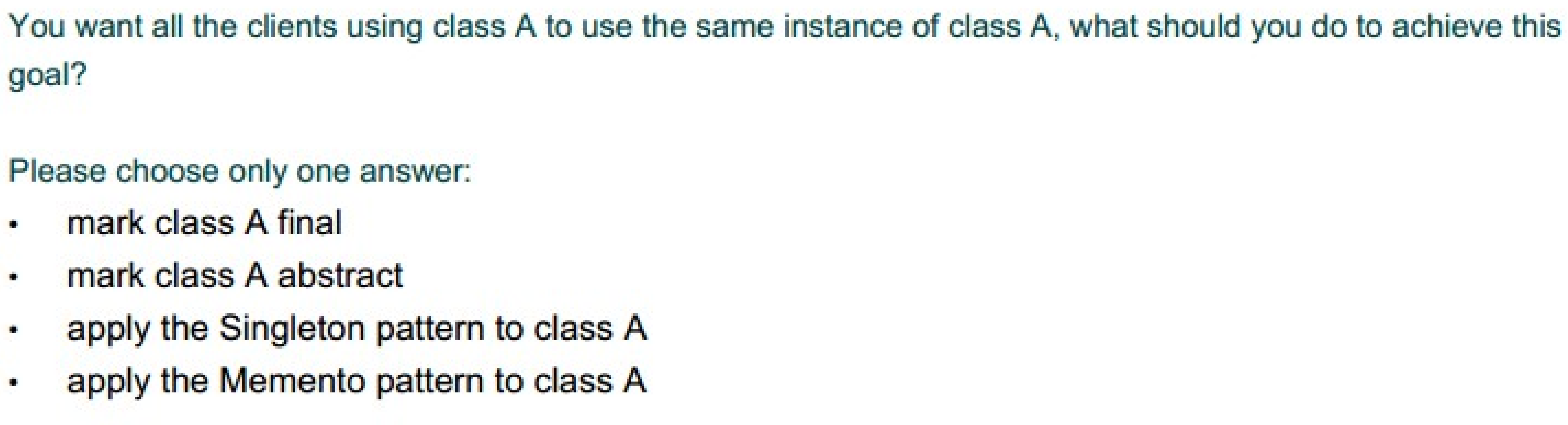
Author: Yasser Ibrahim

Which design pattern you would you use to limit the class instantiation to one object?

Please choose only one answer:

- Factory Method Design Pattern
- Builder design pattern
- Prototype design pattern
- [✓ Singleton design pattern]

Warm UP

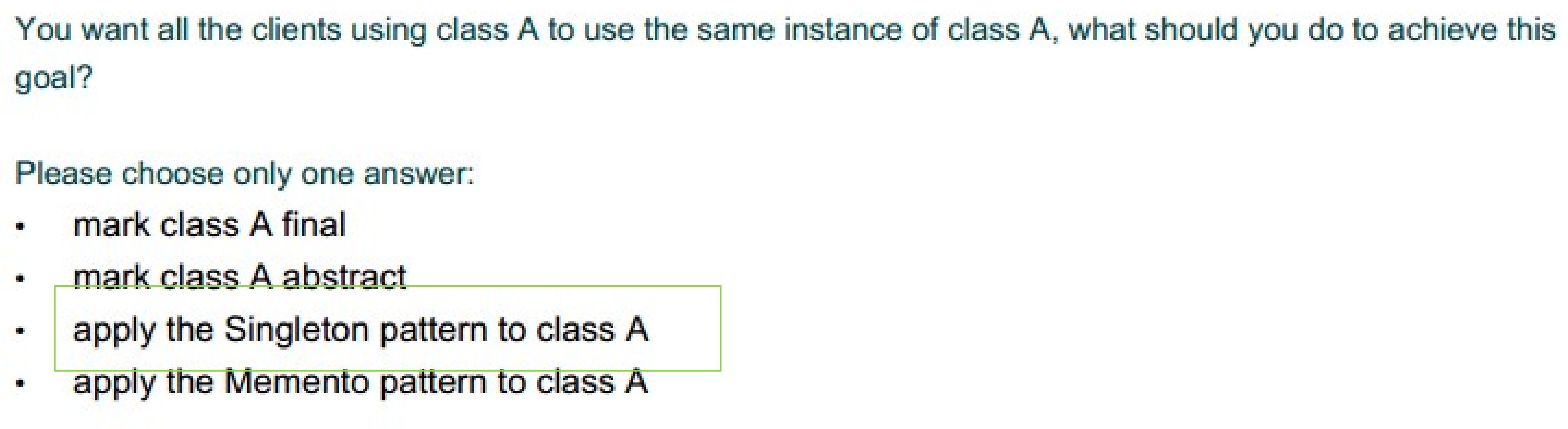


You want all the clients using class A to use the same instance of class A, what should you do to achieve this goal?

Please choose only one answer:

- mark class A final
- mark class A abstract
- apply the Singleton pattern to class A
- apply the Memento pattern to class A

Warm UP

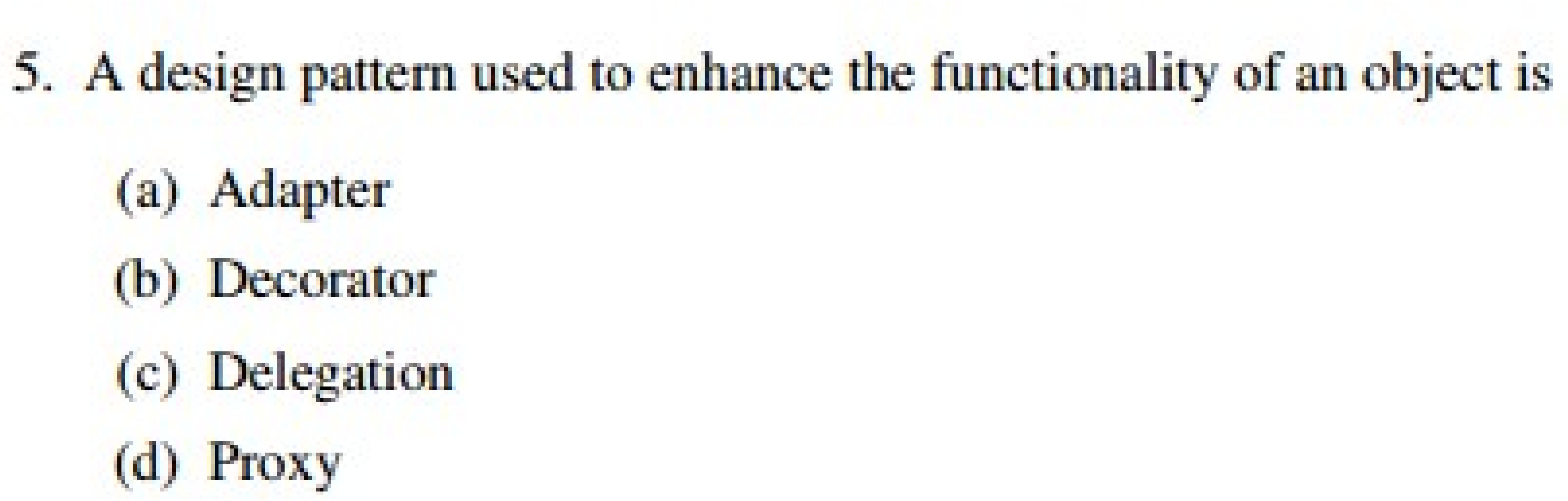


You want all the clients using class A to use the same instance of class A, what should you do to achieve this goal?

Please choose only one answer:

- mark class A final
- mark class A abstract
- [✓ apply the Singleton pattern to class A]
- apply the Memento pattern to class A

Warm UP



5. A design pattern used to enhance the functionality of an object is

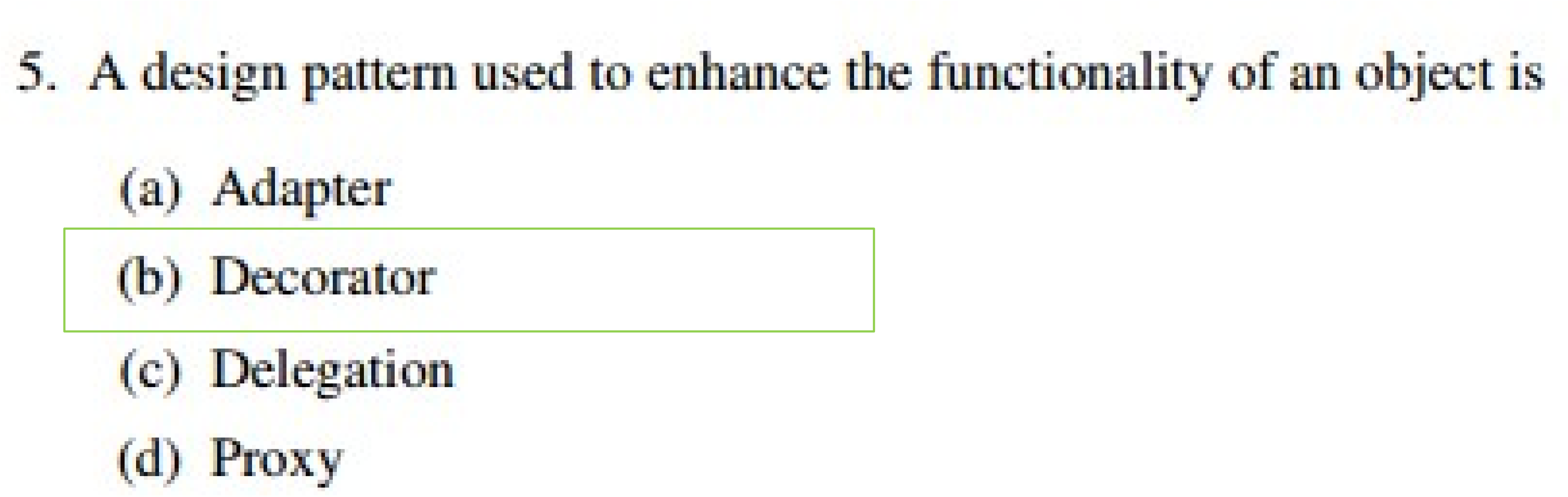
(a) Adapter

(b) Decorator

(c) Delegation

(d) Proxy

Warm UP



5. A design pattern used to enhance the functionality of an object is

(a) Adapter

[✓ (b) Decorator]

(c) Delegation

(d) Proxy

Concept

What does it mean to design for the interface versus design for the implementation?

Concept

What does it mean to design for the interface versus design for the implementation?

Following this principle leads to code that exhibits loose coupling among classes. Examples of these benefits include:

a)Better testability. Classes that are not under tests can be substituted with fakes that implement the same interface. I.e you could substitute AJAXLibrary with FakeAJAX if the class just expects anything that implements an interface RemoteCallLibrary.

b)Better code reuse. If you have a routine that can sort anything that implements Comparable interface, you can now sort almost anything pretty easily instead of writing sorts for every type of object.

Concept

When to use design patterns?

Concept

What is the difference between architecture and design?

Concept

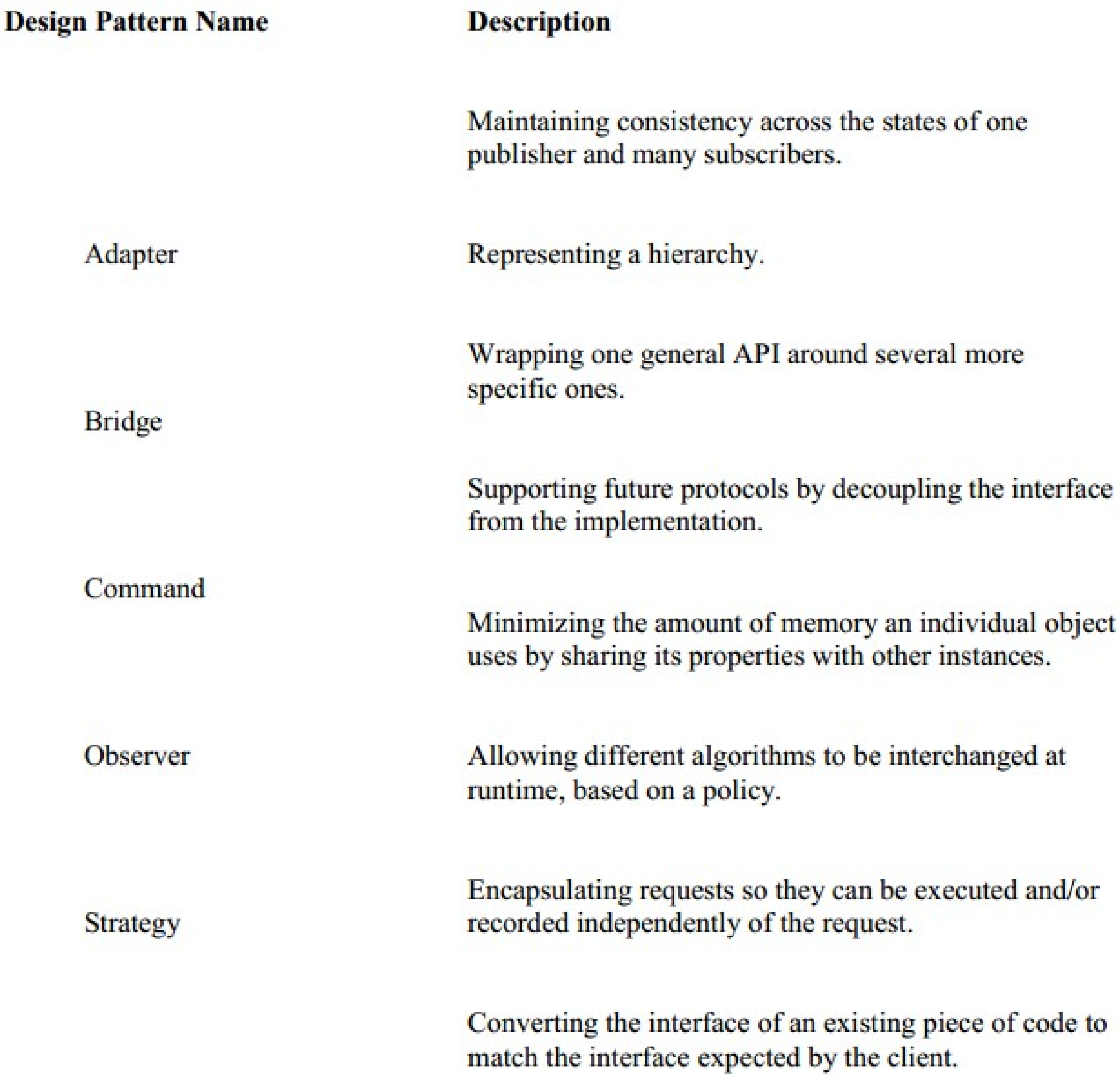
What is the difference between architecture and design?

Architectural styles: Broad approaches to solving problems (pipes & filters, object-oriented, tiers, repository, etc.) that define which underlying tools are available to be used by patterns. These must be decided early, as they provide the general approach that is implemented.

Basic design patterns: Refer to how the internals of a component are arranged in order to solve problems. These are decided on as the problem they solve arises. e.g. An iterator is introduced when an operation must be applied to every component of a data structure.

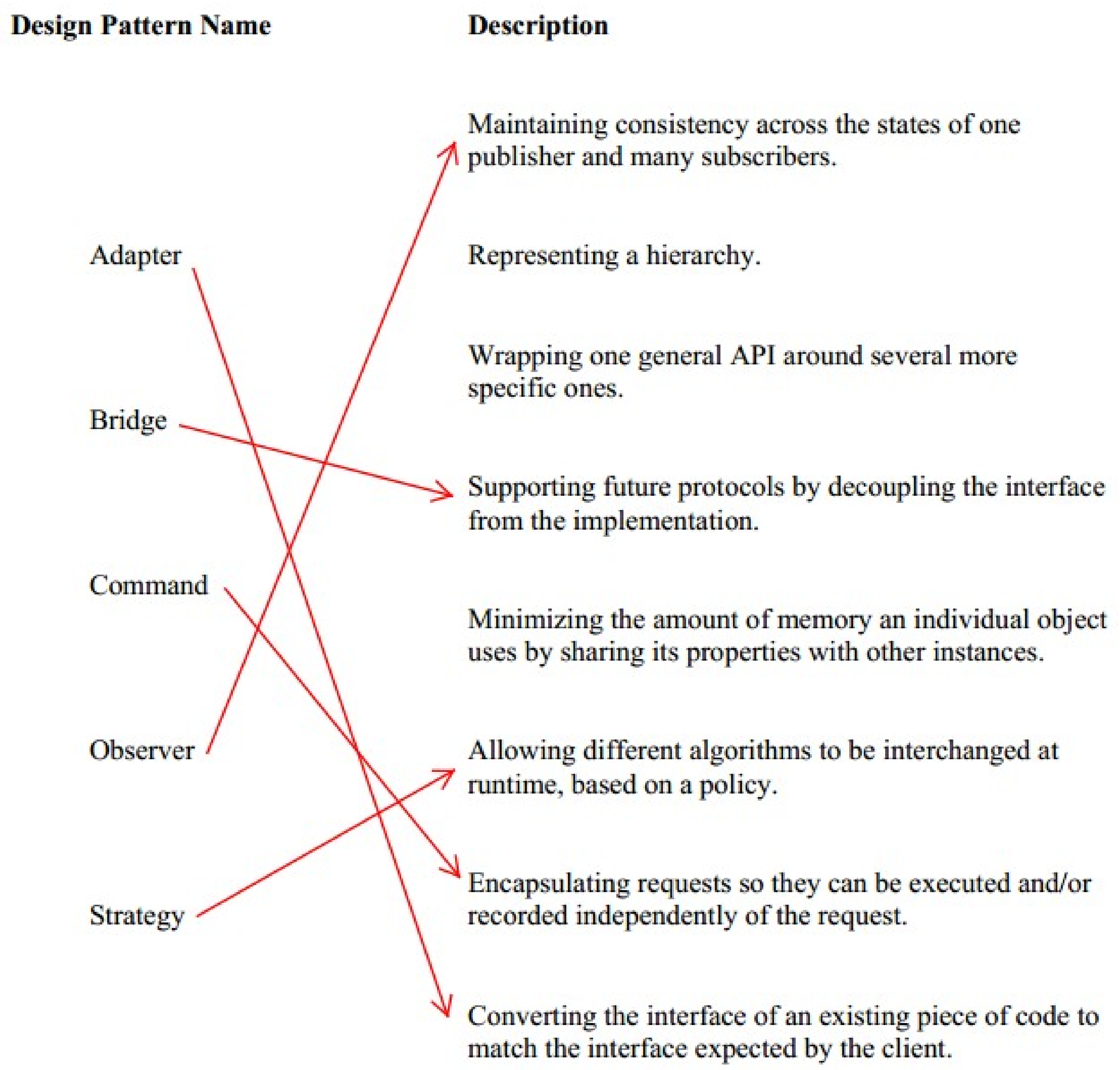
Design patterns specify a general solution schema for problems that is known to have certain desirable properties.

Matching



Design Pattern Name	Description
	Maintaining consistency across the states of one publisher and many subscribers.
Adapter	Representing a hierarchy.
Bridge	Wrapping one general API around several more specific ones.
	Supporting future protocols by decoupling the interface from the implementation.
Command	Minimizing the amount of memory an individual object uses by sharing its properties with other instances.
Observer	Allowing different algorithms to be interchanged at runtime, based on a policy.
Strategy	Encapsulating requests so they can be executed and/or recorded independently of the request.
	Converting the interface of an existing piece of code to match the interface expected by the client.

Matching



Design Pattern Name

Adapter

Bridge

Command

Observer

Strategy

Description

Maintaining consistency across the states of one publisher and many subscribers.

Representing a hierarchy.

Wrapping one general API around several more specific ones.

Supporting future protocols by decoupling the interface from the implementation.

Minimizing the amount of memory an individual object uses by sharing its properties with other instances.

Allowing different algorithms to be interchanged at runtime, based on a policy.

Encapsulating requests so they can be executed and/or recorded independently of the request.

Converting the interface of an existing piece of code to match the interface expected by the client.

Case Study

ParkingMontreal.com is a new system to track the availability of parking spots around the city of Montreal. This system will provide services both to the public, by helping them find nearby available parking spots, and to the City of Montreal parking attendants, by helping them locate parked cars whose meters have expired. Users who are registered with ParkingMontreal.com can also pay for their parking spot using their cellular phone. And, of course, the general public still has the option to pay for parking using the existing kiosk-based system.

References

1. http://www.tutorialspoint.com/design_pattern/design_pattern_overview.htm
2. Factory: https://www.youtube.com/watch?v=ub0DXaeV6hA
3. http://s3.amazonaws.com/docuum/attachments/2139/Old%20final%20winter%202008.pdf?1240434717
4. https://cs.uwaterloo.ca/~a78khan/cs446/additional-material/scribe/99-exams/1005-Final_exam-Solutions.pdf