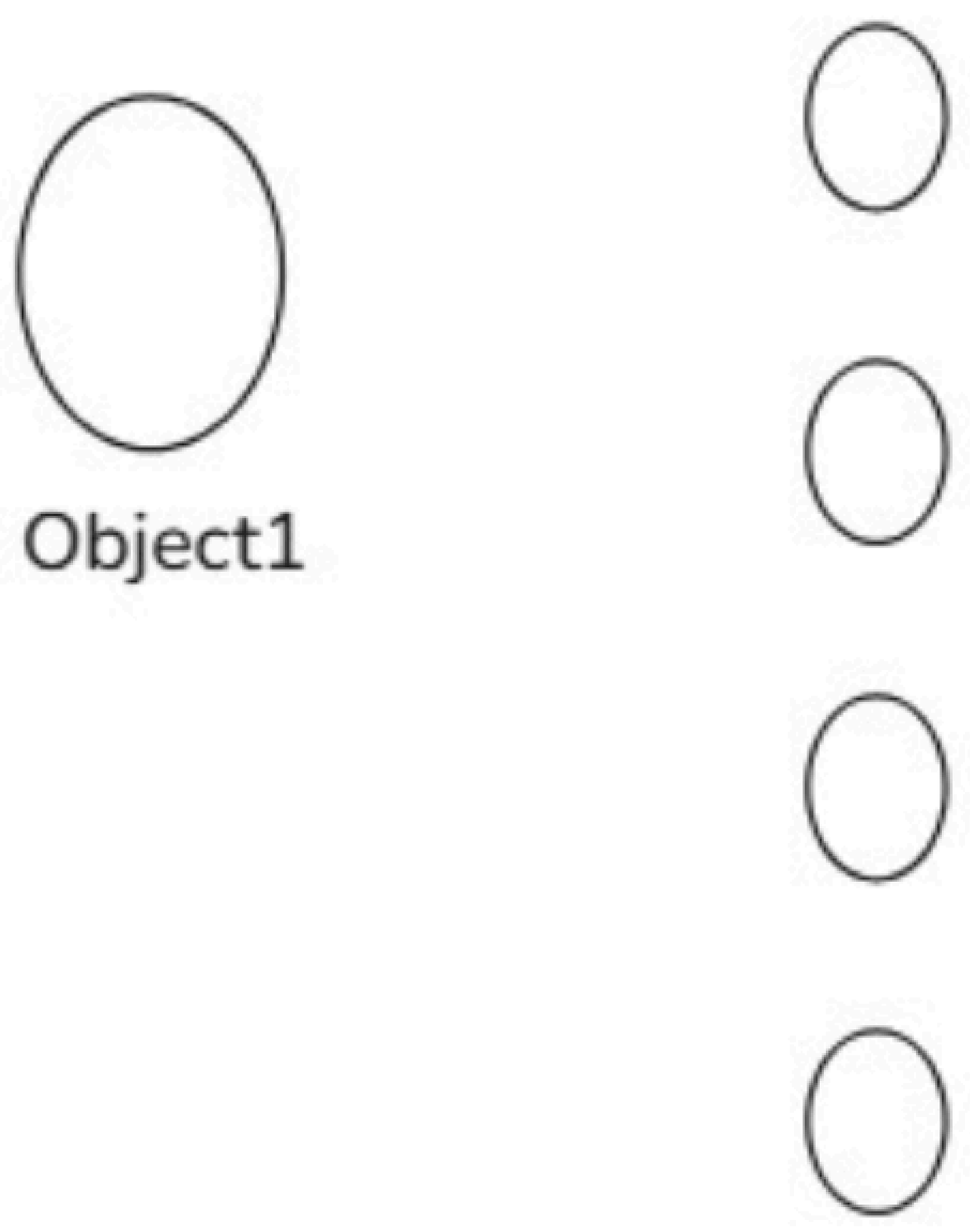
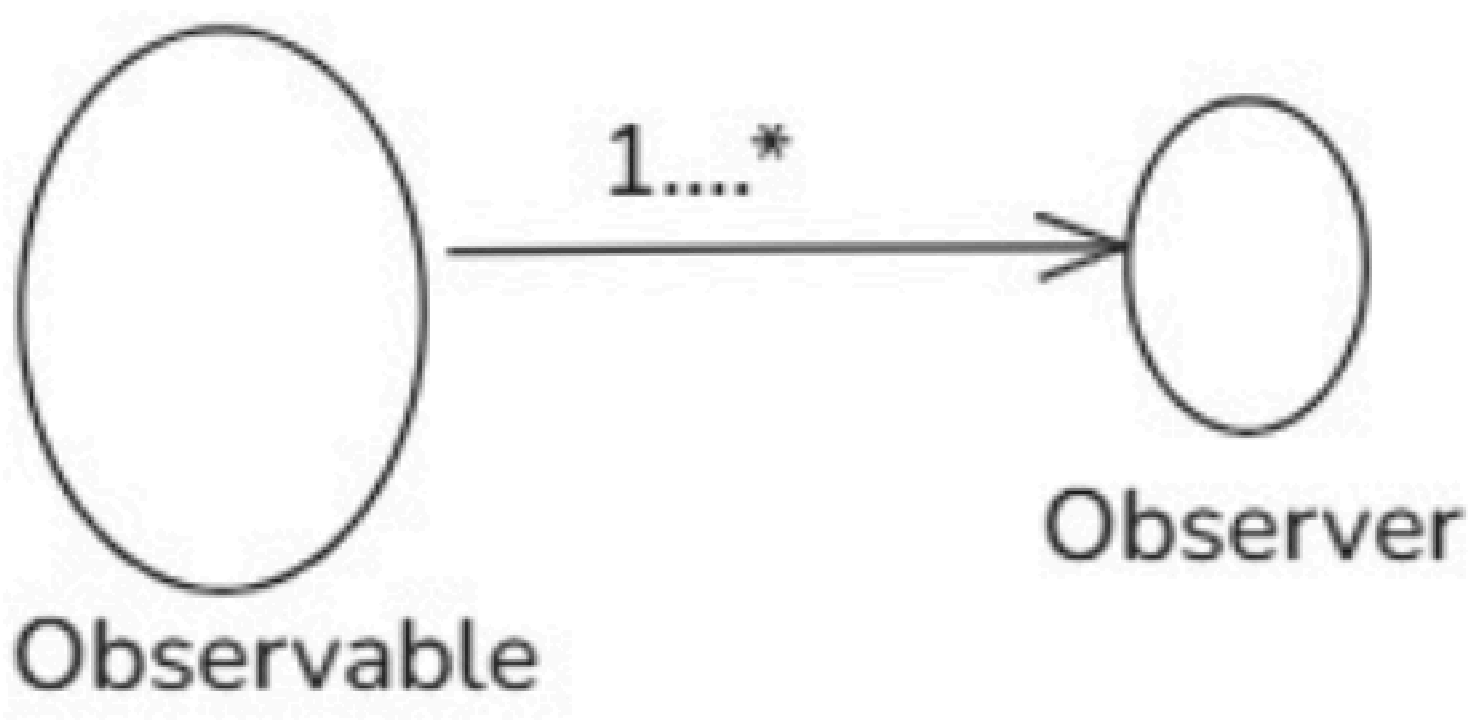
Observer Design Pattern

Whenever a new video is uploaded on youtube you get notification.
How do you get a notification of a subscribed youtube channel?
How subscribed yt account and creator account is communicated?



Polling Technique

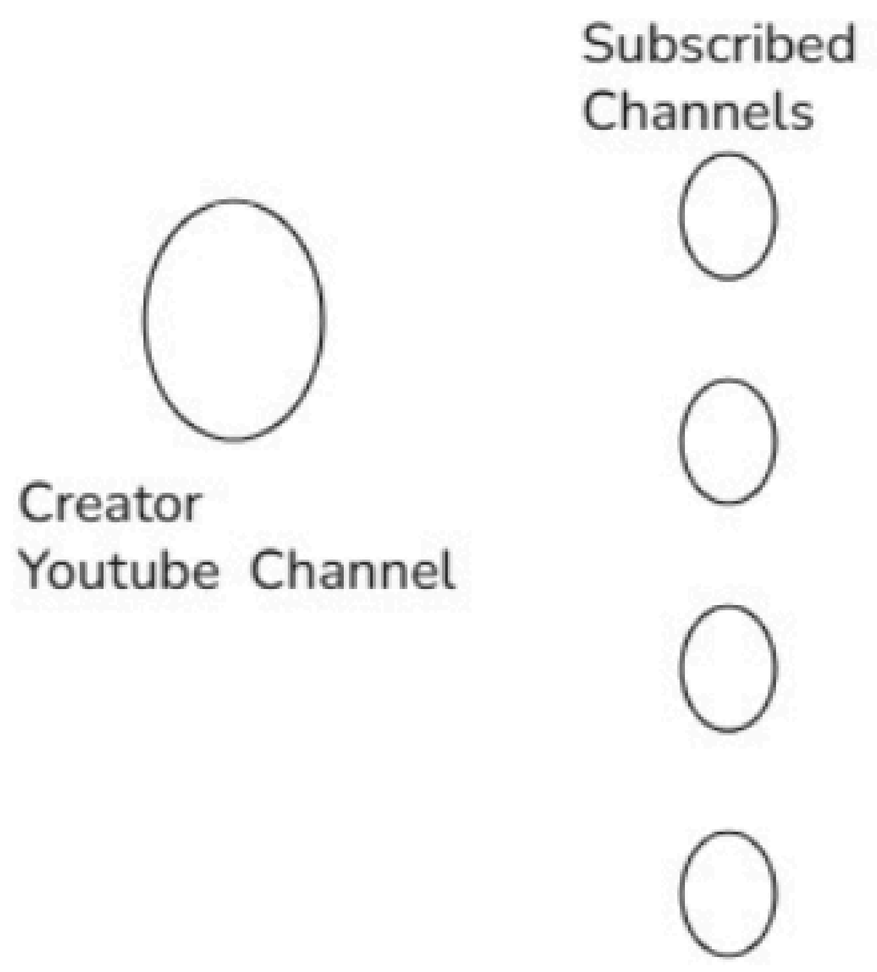


- Observer asks to Observable that is your state changed in frequency of every second.

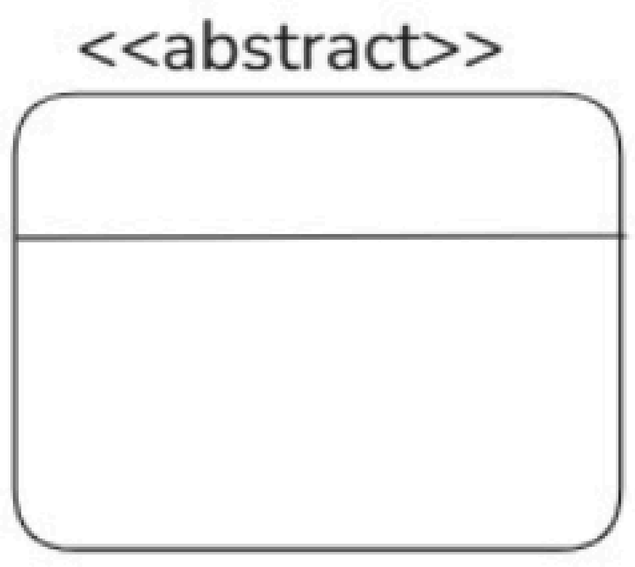
- Multiple times Observer sends request to Observable that have your value changed.It replies no, once value changes in Observable replies yes, then we read that value.
  Asking Frequently to Observable by Observer is very time-consuming process.
  Determing Frequency of Observer.This Concept is very bad.

Pushing Technique

- Whenever there is change in state in Observable.Its Observable responsibilty to inform Observers whenever there is a change in Observable.
- Observable sends request to observers whenever there is change in state then Observer can read the notification.



- maintains a list internally of subscribed channels by Creator Youtube Channel.
- When Creator Youtube Channel state changes internally then Subscribed Channel gets update from Creator Youtube Channel that state changed(new video uploaded).

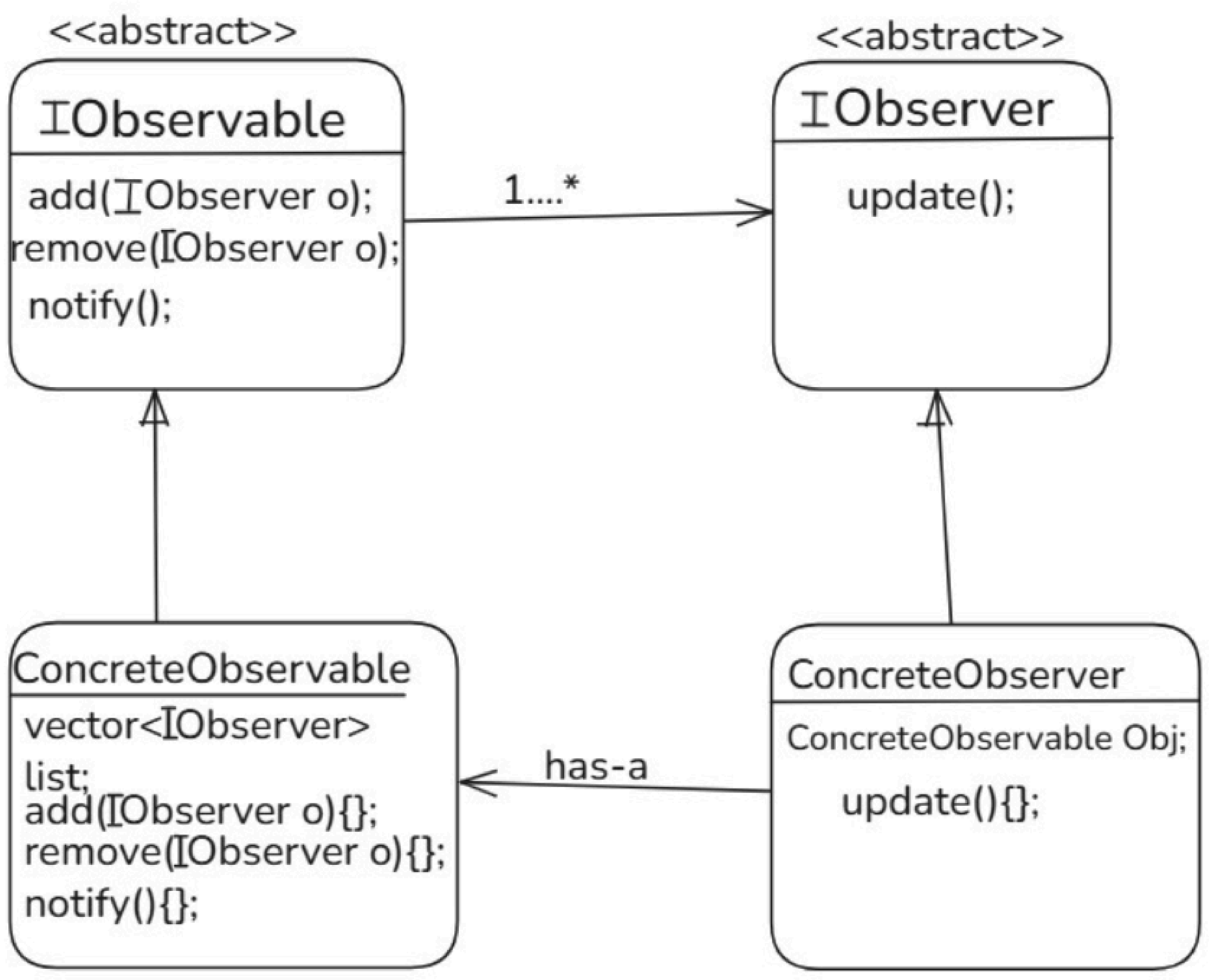


ꞮChannel

pure abstract class where all methods are virtual functions, the concrete class which overrides the abstract Class.

- Abstract in C++ and Interface in other Languages contract/interact with Concrete Classes.
- Pure Abstract Class, in which there will be all virtual functions i.e called as Interface we denote it as Ɪ

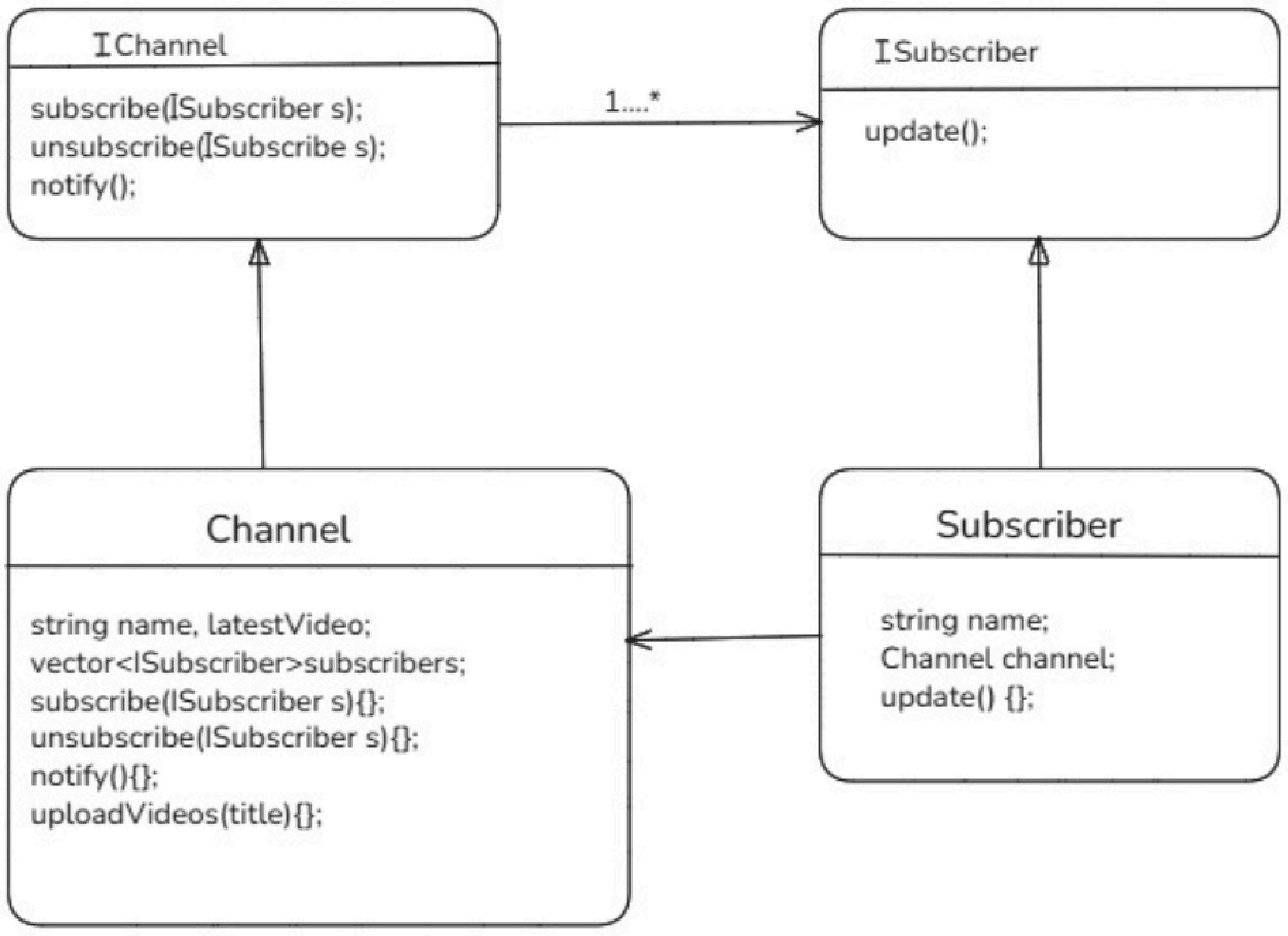
UML Design



Observer Design Pattern Definition:

Defines a one-to-many relationship b/w objects so that when one object changes state, all of its dependents are notified, and updated automatically.

Youtube Example


IChannel
subscribe(ISubscriber s);
unsubscribe(ISubscribe s);
notify();
1....*
ISubscriber
update();
Channel
string name, latestVideo;
vector<ISubscriber>subscribers;
subscribe(ISubscriber s){};
unsubscribe(ISubscriber s){};
notify(){};
uploadVideos(title){};
Subscriber
string name;
Channel channel;
update() {};

* Observer breaks SRP

- Concrete Channel doing 2 tasks i)subscribe & unsubscribe ii)upload video.
  Its handling both observer design logic (subscribe & unsubscribe) and business logic(upload video,get video).

- To maintain SRP, we don't make IChannel as AbstractClass and we write the Observer Design Logic(Subscribe & Unsubscribe Methods Code) & in Channel only business logic we write UploadVideo Method.

- Business Logic can be changed but Observer Design Logic Functionality won't change.

- The code which changes keep them seperate i.e Businesss Logic
  from no-change code i.e Observer Design Functionality.

Real World Examples

->Notification Service
In Facebook a new video posted, it notifies to their followers.

->Event handling(Programming)