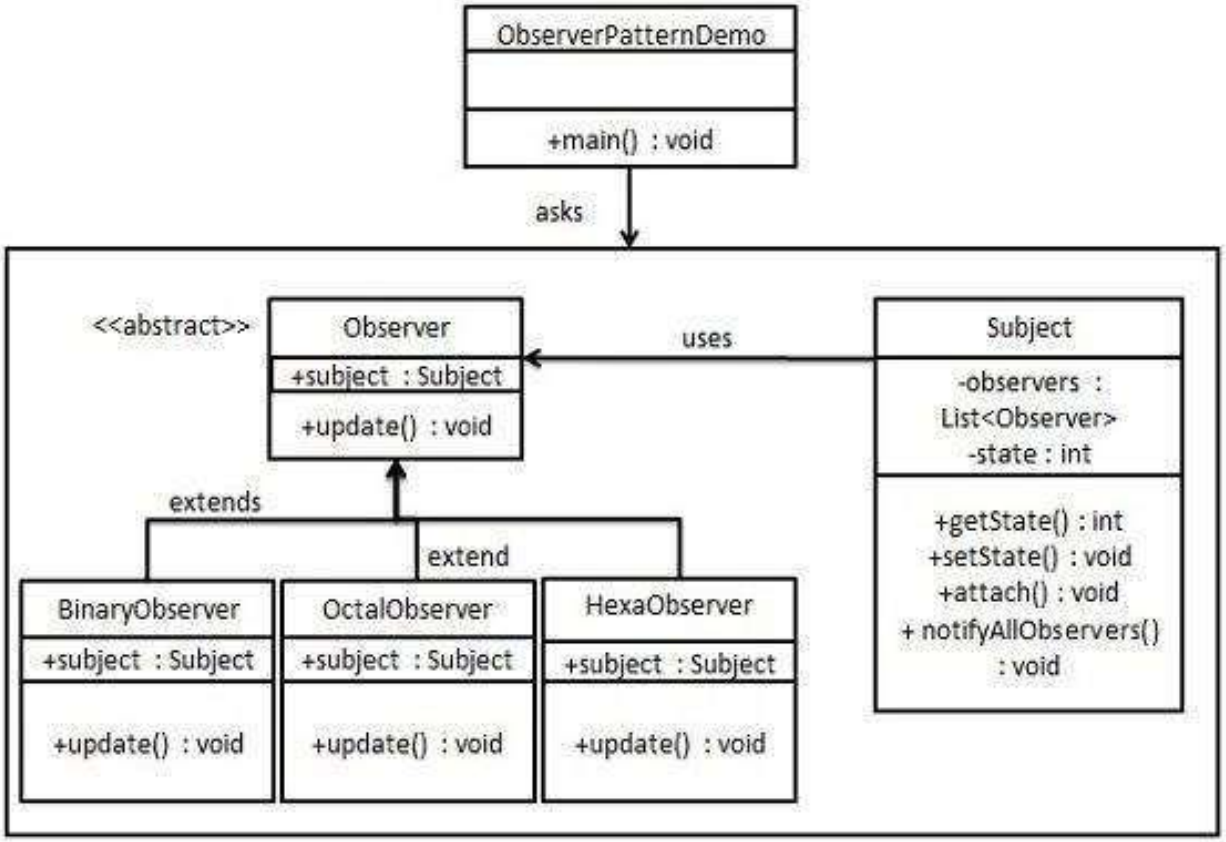
# Observer Design Pattern

# Observer Design Pattern

Observer pattern falls under behavioral pattern category.

Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its dependent objects are to be notified automatically.

# Example


ObserverPatternDemo
+main() : void
asks
<<abstract>>
Observer
+subject : Subject
+update() : void
uses
Subject
-observers : List<Observer>
-state : int
+getState() : int
+setState() : void
+attach() : void
+ notifyAllObservers() : void
extends
extend
BinaryObserver
+subject : Subject
+update() : void
OctalObserver
+subject : Subject
+update() : void
HexaObserver
+subject : Subject
+update() : void

# Implementation

- Observer pattern uses three actor classes. Subject, Observer and Client. Subject is an object having methods to attach and detach observers to a client object.
- An abstract class has been created as Observer and a concrete class Subject that is extending class Observer.
- ObserverPatternDemo, our demo class, will use Subject and concrete class object to show observer pattern in action.
- Let us consider another example…

# Example

**Subscriber.java**

```
package com.observer;
public class Subscriber {
private String name;
private Channel channel = new Channel();
public Subscriber(String name) {
this.name = name;
}
public void update() {
System.out.println("Hey "+ name+ " Video Uploaded");
}
public void subscribeChannel(Channel ch) {
channel = ch;
}
}
```

## Channel.java

```
package com.observer;

import java.util.ArrayList;
import java.util.List;

public class Channel {
	List <Subscriber> subs = new ArrayList<>();
	private String title;

	public void subscribe(Subscriber sub) {
		subs.add(sub);
	}

	public void unSubscribe(Subscriber sub) {
		subs.remove(sub);
	}

	public void notifySubscribers() {
		for(Subscriber sub:subs)
		{
			sub.update();
		}
	}

	public void upload(String title)
	{
		this.title = title;
		notifySubscribers();
	}
}
```

**Youtube.java**

```
package com.observer;
public class Youtube {
public static void main(String[] args) {
// TODO Auto-generated method stub
Channel design_pattern = new Channel();
Subscriber s1 = new Subscriber(“mno");
Subscriber s2 = new Subscriber("abc");
Subscriber s3 = new Subscriber("pqr");
Subscriber s4 = new Subscriber("xyz");
design_pattern.subscribe(s1);
design_pattern.subscribe(s2);
design_pattern.subscribe(s3);
design_pattern.subscribe(s4);
s1.subscribeChannel(design_pattern);
s2.subscribeChannel(design_pattern);
s3.subscribeChannel(design_pattern);
s4.subscribeChannel(design_pattern);
design_pattern.upload("Learn C++");
}
}
```

**Output**

```
Hey mno Video Uploaded
Hey abc Video Uploaded
Hey pqr Video Uploaded
Hey xyz Video Uploaded
```