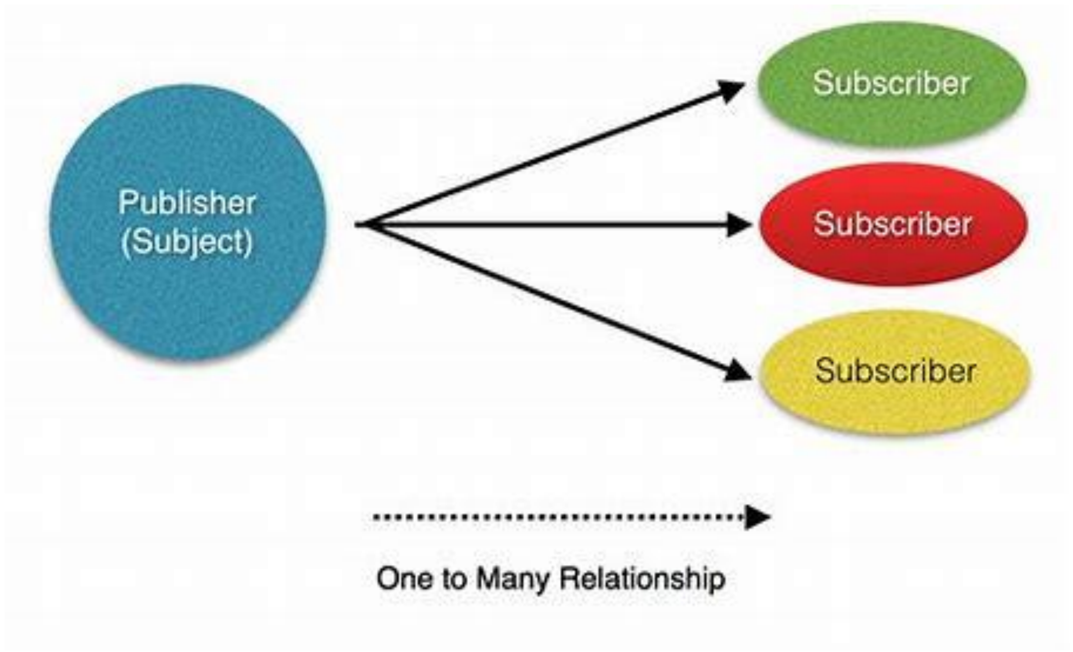
观察者模式Observer

波波老师~研发总监/资深架构师

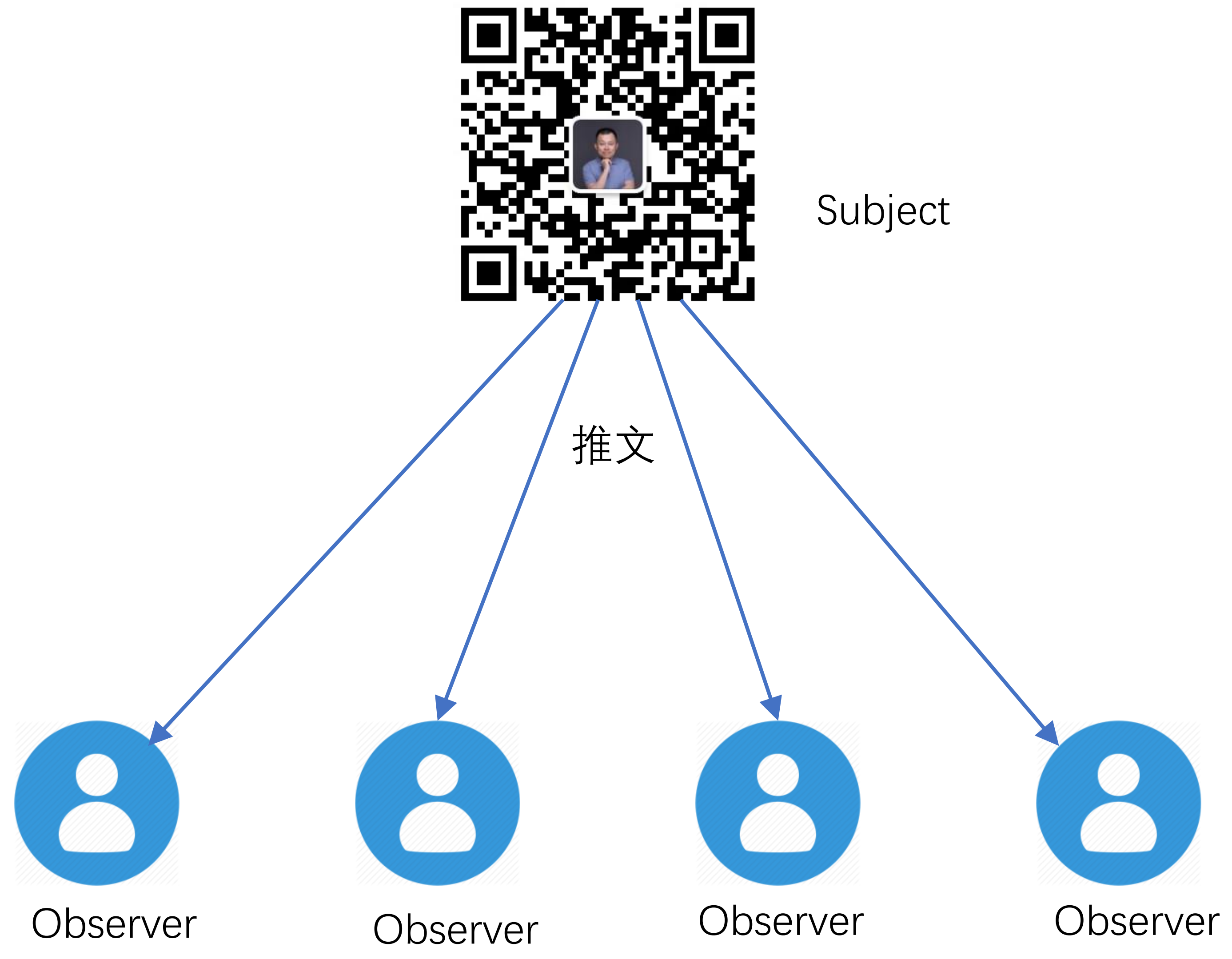
波波微课
spring2go.com

定义

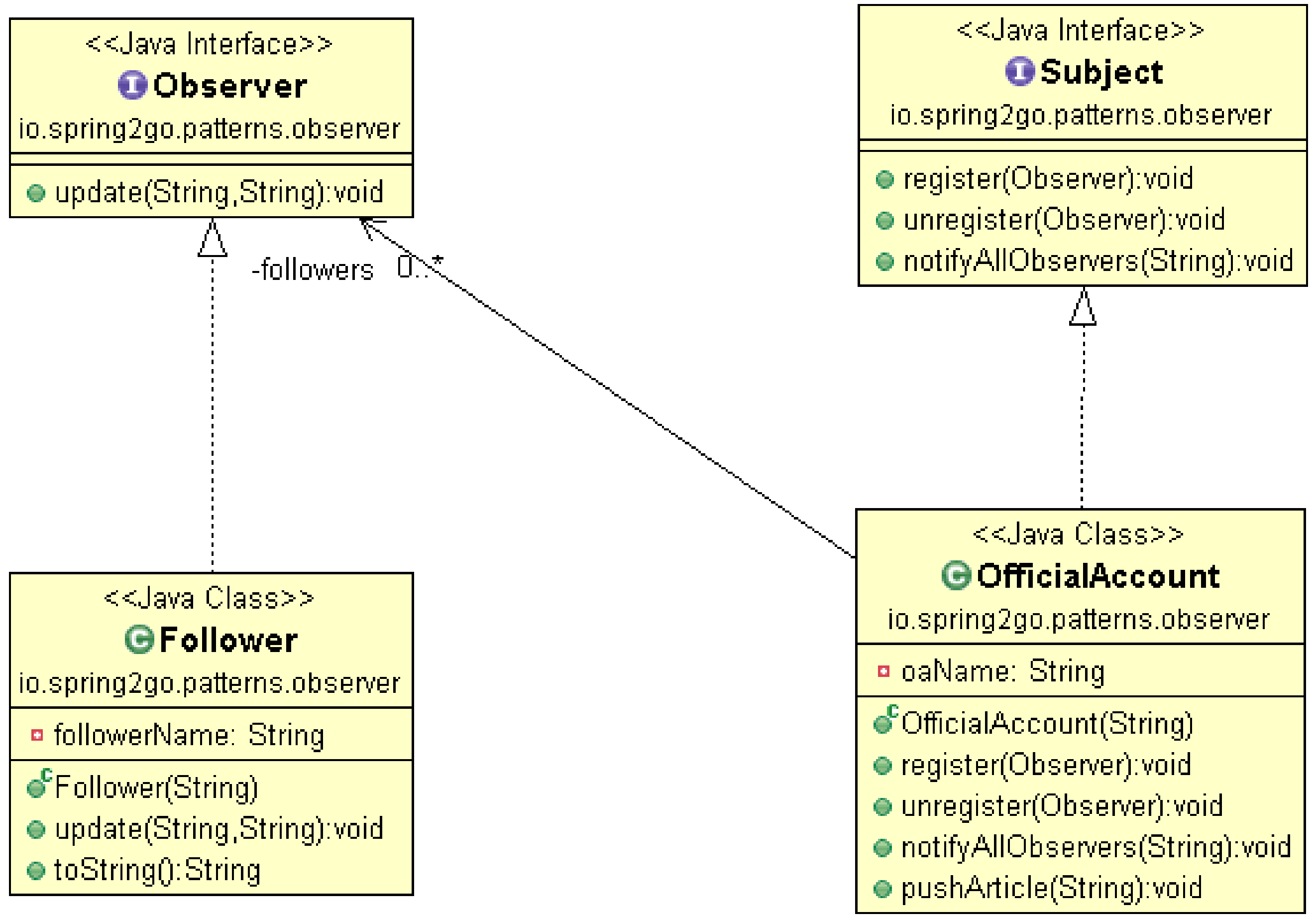
- 观察者模式定义了对象间的一种一对多依赖，当一个对象改变状态时，所有依赖的对象会收到通知并自动更新。
- 也称发布者-订阅者(Publisher-Subscriber)模式



案例~微信公众号关注

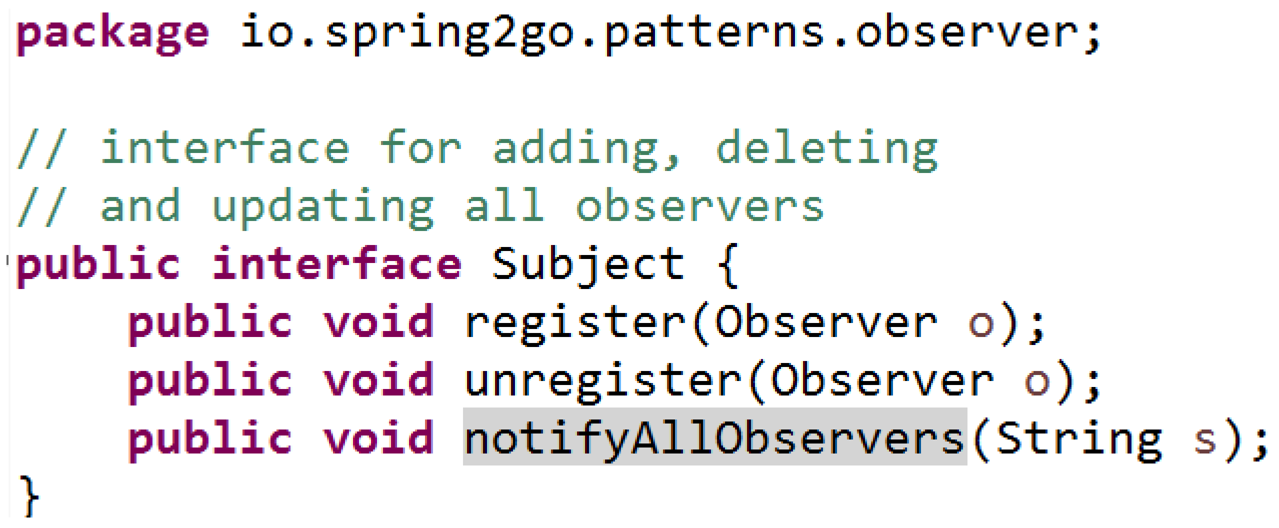


UML关系图



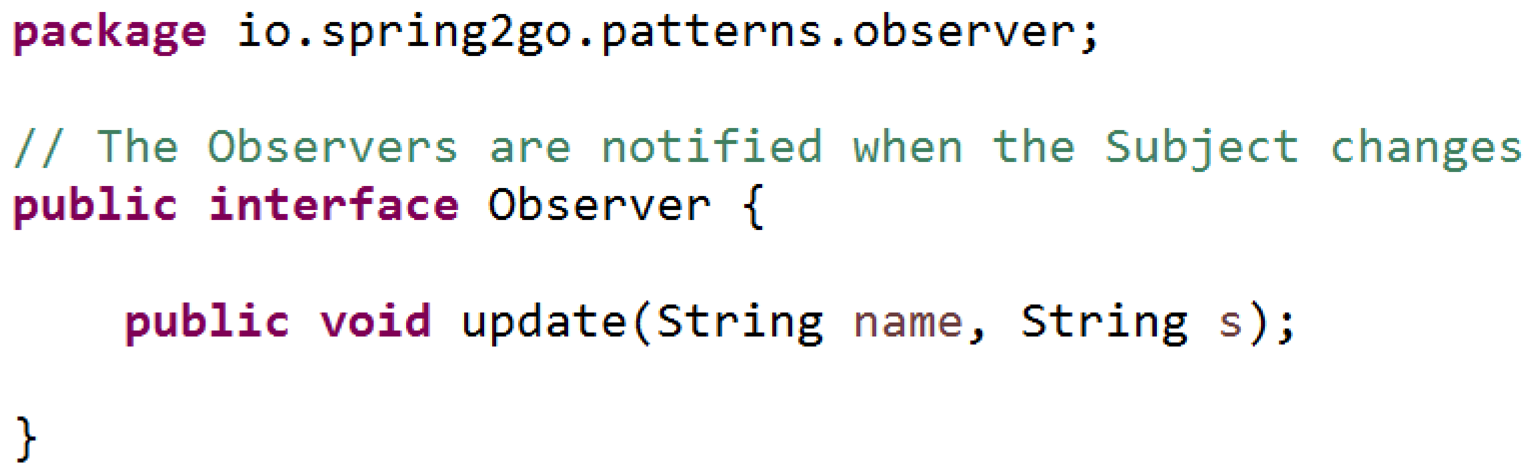
<<Java Interface>>
Observer
io.spring2go.patterns.observer
update(String,String):void
<<Java Interface>>
Subject
io.spring2go.patterns.observer
register(Observer):void
unregister(Observer):void
notifyAllObservers(String):void
-followers
0..*
<<Java Class>>
Follower
io.spring2go.patterns.observer
followerName: String
Follower(String)
update(String,String):void
toString():String
<<Java Class>>
OfficialAccount
io.spring2go.patterns.observer
oaName: String
OfficialAccount(String)
register(Observer):void
unregister(Observer):void
notifyAllObservers(String):void
pushArticle(String):void

代码~Subject接口



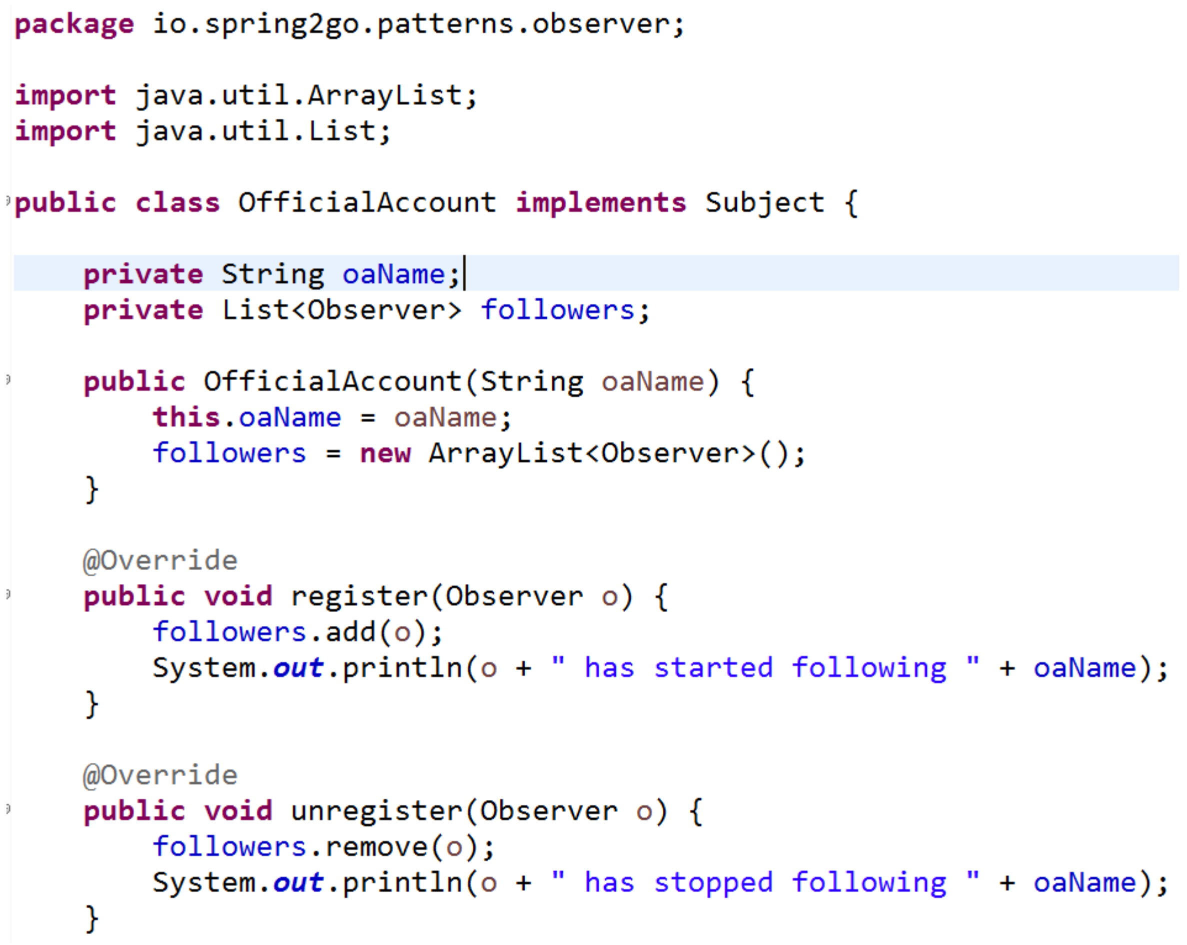
```java
package io.spring2go.patterns.observer;

// interface for adding, deleting
// and updating all observers
public interface Subject {
    public void register(Observer o);
    public void unregister(Observer o);
    public void notifyAllObservers(String s);
}
```

代码~Observer接口

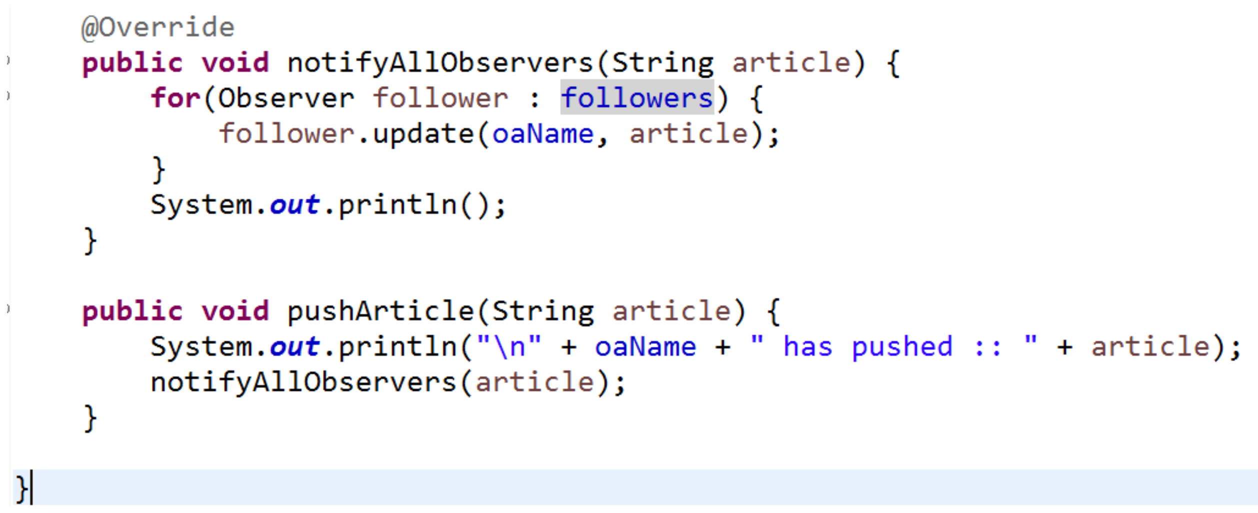


```java
package io.spring2go.patterns.observer;

// The Observers are notified when the Subject changes
public interface Observer {

    public void update(String name, String s);

}
```

代码~Subject实现



```java
package io.spring2go.patterns.observer;

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

public class OfficialAccount implements Subject {

    private String oaName;
    private List<Observer> followers;

    public OfficialAccount(String oaName) {
        this.oaName = oaName;
        followers = new ArrayList<Observer>();
    }

    @Override
    public void register(Observer o) {
        followers.add(o);
        System.out.println(o + " has started following " + oaName);
    }

    @Override
    public void unregister(Observer o) {
        followers.remove(o);
        System.out.println(o + " has stopped following " + oaName);
    }
```



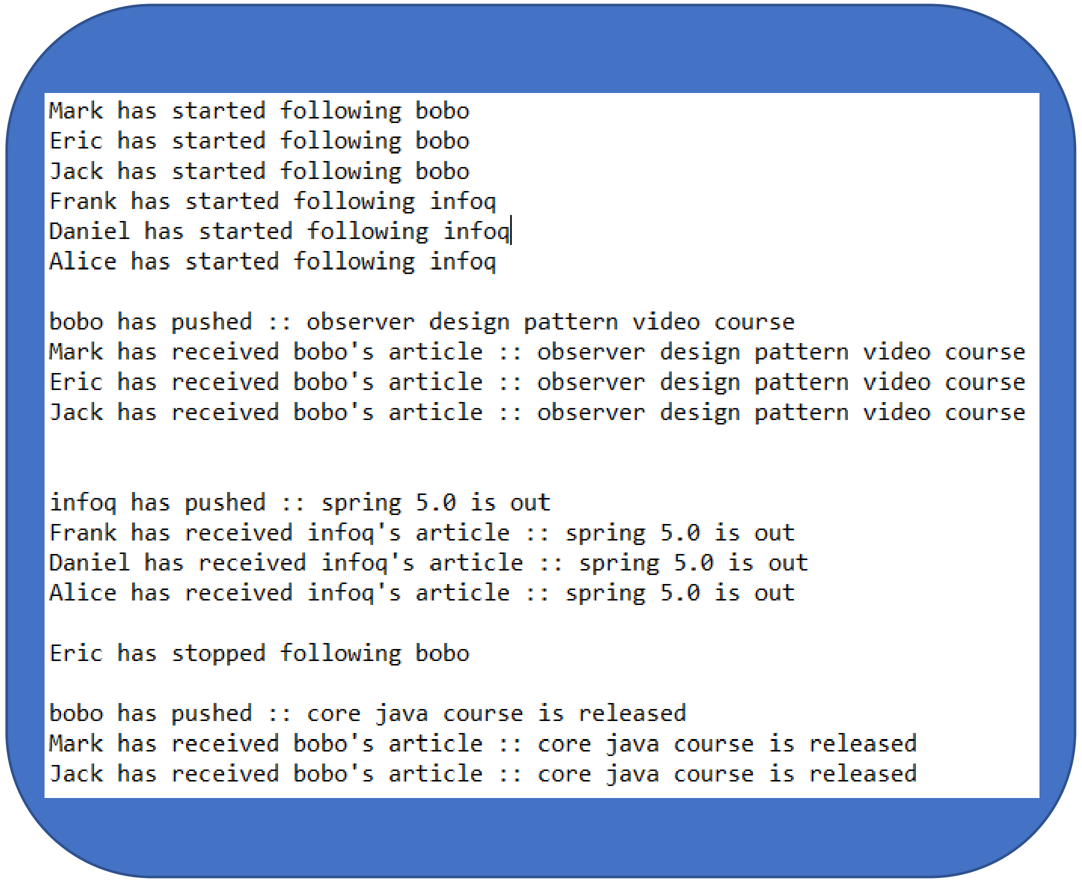
```java
    @Override
    public void notifyAllObservers(String article) {
        for(Observer follower : followers) {
            follower.update(oaName, article);
        }
        System.out.println();
    }

    public void pushArticle(String article) {
        System.out.println("\n" + oaName + " has pushed :: " + article);
        notifyAllObservers(article);
    }

}
```

代码~Observer实现

```java
package io.spring2go.patterns.observer;

public class Follower implements Observer {

    private String followerName;

    public Follower(String followerName) {
        this.followerName = followerName;
    }

    @Override
    public void update(String oaName, String article) {
        System.out.println(followerName + " has received "
                + oaName + "'s article :: " + article );
    }

    @Override
    public String toString() {
        return followerName;
    }

}
```

代码~客户端

```java
public static void main(String[] args) {
    OfficialAccount bobo = new OfficialAccount("bobo");
    OfficialAccount infoq = new OfficialAccount("infoq");

    Follower mark = new Follower("Mark");
    Follower eric = new Follower("Eric");
    Follower jack = new Follower("Jack");
    Follower frank = new Follower("Frank");
    Follower daniel = new Follower("Daniel");
    Follower alice = new Follower("Alice");

    bobo.register(mark);
    bobo.register(eric);
    bobo.register(jack);

    infoq.register(frank);
    infoq.register(daniel);
    infoq.register(alice);

    bobo.pushArticle("observer design pattern video course");
    infoq.pushArticle("spring 5.0 is out");

    bobo.unregister(eric);

    bobo.pushArticle("core java course is released");
}
```



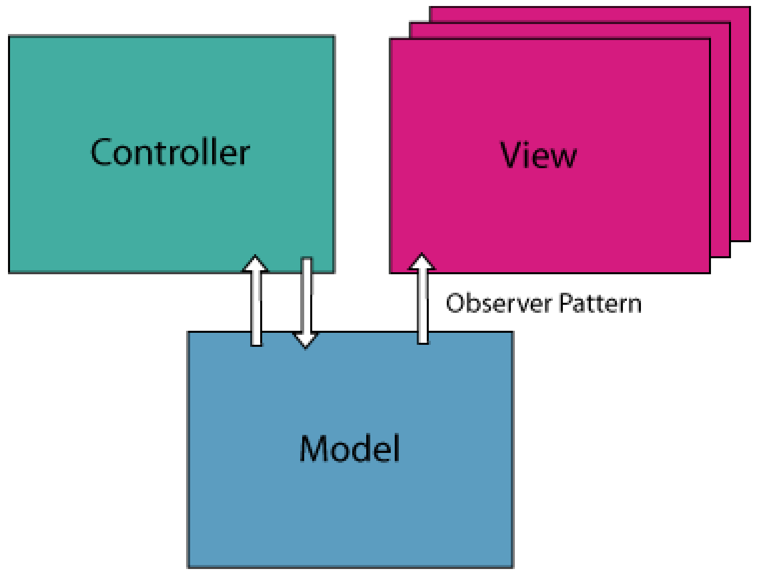
```
Mark has started following bobo
Eric has started following bobo
Jack has started following bobo
Frank has started following infoq
Daniel has started following infoq
Alice has started following infoq

bobo has pushed :: observer design pattern video course
Mark has received bobo's article :: observer design pattern video course
Eric has received bobo's article :: observer design pattern video course
Jack has received bobo's article :: observer design pattern video course


infoq has pushed :: spring 5.0 is out
Frank has received infoq's article :: spring 5.0 is out
Daniel has received infoq's article :: spring 5.0 is out
Alice has received infoq's article :: spring 5.0 is out

Eric has stopped following bobo

bobo has pushed :: core java course is released
Mark has received bobo's article :: core java course is released
Jack has received bobo's article :: core java course is released
```

优势

- Subject和Observer之间解耦
- 支持广播

应用

- java.util.Observer/java.util.Observable(实际用得很少)
- Swing中java.util.EventListener
- Spring中ApplicationListener/ApplicationEvent
- JMS
- MVC framework



参考

- Observer Design Pattern
  - http://codepumpkin.com/observer-design-pattern/

代码

- https://github.com/spring2go/core-spring-patterns

波波微课
spring2go.com

波波微课

spring2go.com