Observer

```
// Observer
public abstract class Subscriber {
	protected Publisher publisher;
	public abstract void update();
}
public class ConcreteSubscriber extends Subscriber {
	public ConcreteSubscriber(Publisher publisher) {
		this.publisher = publisher;
		this.publisher.subscribe(this);
	}
	@Override
	public void update() {
		System.out.println("[UPDATE]: " + publisher.getState());
	}
}
import java.util.ArrayList;
public class Publisher {
	ArrayList<Subscriber> subscrubers = new ArrayList<>();
	Object state = new Object();
	public Object getState() {
		return state;
	}
	public void setState(Object state) {
		this.state = state;
		notifySubscribers();
	}
	public void subscribe(Subscriber s) {
		subscrubers.add(s);
	}
	public void unsubscribe(Subscriber s) {
		subscrubers.remove(s);
	}
	public void notifySubscribers() {
		for(Subscriber s : subscrubers) {
			s.update();
		}
	}
}
```