1. 예제로 주어진 코드와는 다른 두 가지 방법으로 방법으로 QuackObservable의 메소드 (registerObserver(), notifyObservers())를 구현해 보시오.

* Quackable의 인터페이스를 구현하는 모든 오리 클래스에서 구현하는 방법

|  |
| --- |
| 다음과 같이 duck객체들과 거위어탭터를 수정해준다.   * 1. ArrayList추가   2. RegisterObserver에 add기능 구현   3. NotifyObservers에 이터레이터 구현   4. 이터레이터 get메소드정의 |
| <결과화면> |

* java.util.Observable 클래스를 이용하는 방법

|  |
| --- |
|  |
|  |
|  |
| <실행화면> |

1. QuackObservable의 메소드 (registerObserver(), notifyObservers())를 구현하는 방법을 비교 분석해 보세요

* Quackable의 인터페이스를 구현하는 모든 오리 클래스에서 구현하는 방법

|  |
| --- |
| 인터레이스를 각 오리클래스에서 구현하게 되면 모든 클래스마다 같은 구현 부분을 집어 넣어야 하기 때문에 프로그래밍이 힘들고 중복된 코드가 생성된다. 따라서 어떤 기능을 하는지 구체적인 코드 구현을 보는 것은 쉽지만 객체지향적인 관점에서보면 좋지 코드의 재사용성과 유지보수측면에서 좋지 못하다. |

* java.util.Observable 클래스를 이용하는 방법 :

|  |
| --- |
| 이미 정의된 자바유틸을 이용하므로 간편하게 구현이 가능하다. 그러나 Observable의 사용방법은 상속을 받아야한다. 자바는 다중상속이 안되기 때문에 다른 클래스의 상속이 필효한 상황이 발생하면 곤란해진다. |

* QuackObservable의 메소드를 Observable 이라는 한 클래스 캡슐화 해 놓은 다음 구성을 통해서 오리 클래스에 포함시킴

|  |
| --- |
| 클래스 객체의 구성하고 위임을 활용하기 때문에 코드의 기능이 분리되어 유지보수가 편리하고 코드 재사용성이 높다.  또한 상속을 활용한 것이 아니므로 필요시 상속 기능을 갖출 수 있는 장점이있다. |