Pigeon is a kind of Bird, and B777 is a kind of Airplane. Bird and Airplane are abstract classes. Pigeon and B777 are both Flyers. Every Flyer has public void methods takeOff() and land(). Some Birds are not Flyers (e.g., Penguin).

A Pigeon is in one of two states: Relaxed or Startled. It has public void methods setRelaxed() and setStartled().

When startled, the Pigeon’s reaction is to takeOff and fly away. Pigeons observe one other, and startled easily, so that if one suddenly takes off, others that are watching are also startled and will take off as well. But flying is relaxing to Pigeons, so, after a short refreshing flight, they begin to settle down and land. It’s a pigeon thing (not all Birds so closely couple their flocking behaviors).

Airplanes can’t reliably see one another, so they are observed by an ATC (for “Air Traffic Controller”). Also, Airplanes observe the ATC, which tells them when to takeOff and land.

A Boy is a Visitor to Flyers. If he visits a Pigeon, the latter will be startled (and take off, see above). If he visits an Airplane, he will be invited into the cockpit\(^1\) and given a little pair of wings (by Airplane’s public Gift getGift() method).

To help with the fact that a given class can extend only one other class, you are given a special version of Observable (which implements ObservableI), and Observer designed for use with the Adaptor Pattern. In addition to a default constructor, Observable permits specifying the parent ObservableI that owns the delegate Observable:

```java
    public Observable(ObservableI o) { // special constructor
        public Observable() { this(this); } // default constructor
```

The Observer interface is also slightly modified, with the first argument to public void update being an ObservableI, not an Observable.

Design (don’t fully implement) a system for the above that uses the State, Observer, Adaptor and Visitor patterns. Draw UML class diagrams (there are about 13 boxes to draw, so don’t spend time on the details of listing all methods unless you have time at the end, but do show which classes own instances of what other classes, etc.) Supplement with written descriptions, fragments of code. Describe special design features. Show your understanding of the visitor pattern aspect by completely writing the public class Boy.

---

\(^1\) The cockpit, what is it? It’s the little room in the front of the plane where the pilots sit, but that’s not important right now.