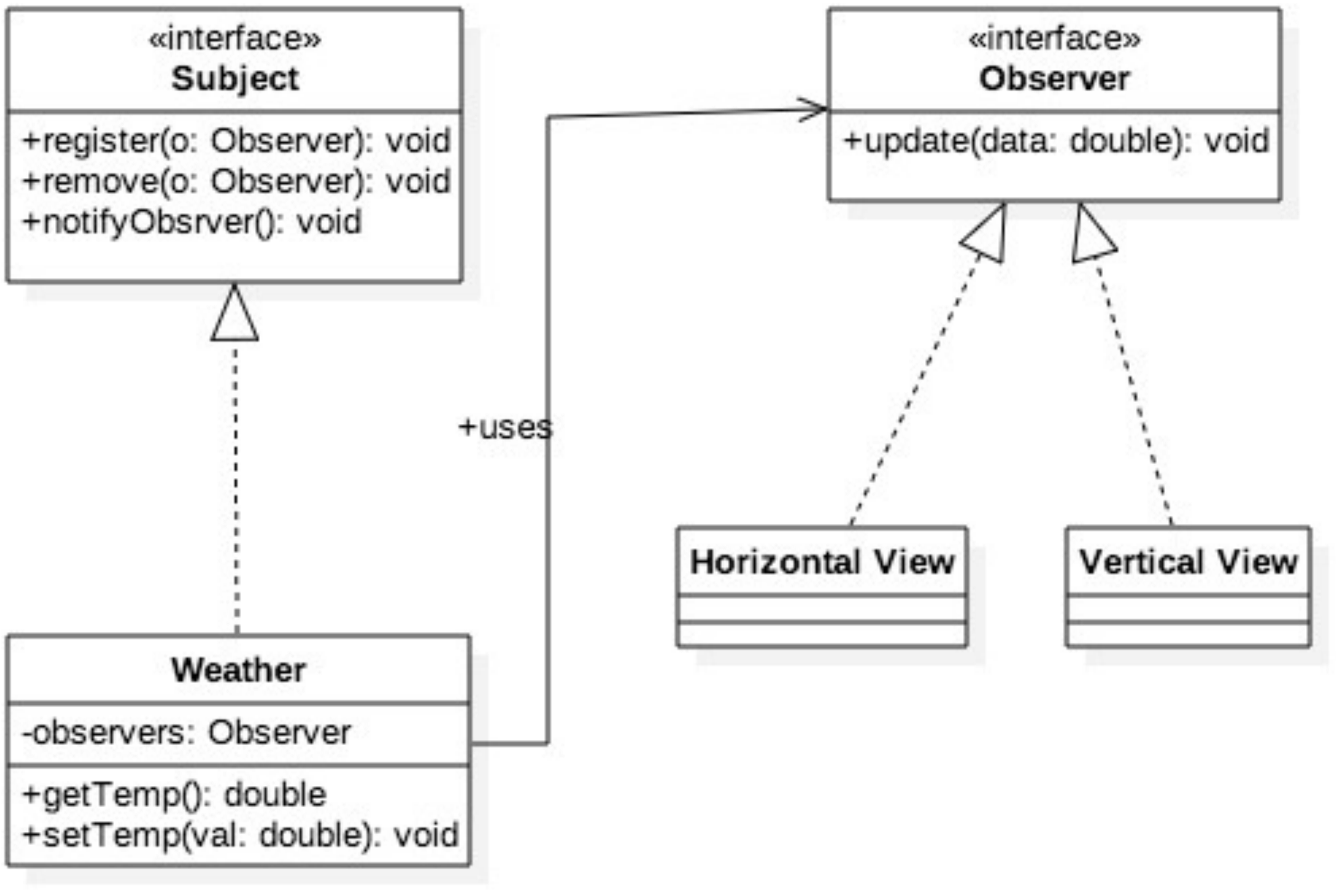
Observer Pattern Example

A Class Exercise

Let's try the following model as an example:



A Five-Step Instruction

Implementation Step 1

Create an Observer Interface:

```
public interface Observer {
    public void update(double data);
}
```

Implementation Step 2

- Create either an interface or abstract class Subject:

```
interface Subject {
    public void register(Observer o);
    public void remove(Observer o);
    public void notifyObserver();

}
```

Implementation Step 3 (cont'd)

```
class Weather implements Subject {
   private double temp;
   private ArrayList <Observer> observers;

   public Weather(double t) {
        observers = new ArrayList<Observer>();
        temp  =  t;
    }

   public void register(Observer o) {
              observers.add(o);
              o.update(temp);
   }
    public void remove(Observer o) {

    }
```

```
 public void notifyObserver() {
   for(int i = 0; i < observers.size(); i++)
   {
        Observer o = observers.get(i);
        o.update(temp);
     }
 }

 public double getTemp(){
        return temp;
  }

   public void setTemp(double t){
             temp = t;
             notifyObserver();
       }
} // END OF CLASS WEATHER
```

Implementation Step 4

- Create a set of view classes that implement observer:

```
class HorizontalDisplay implements Observer {
        double temp;
        Subject weather;

        public HorizontalDisplay(Subject w) {
                weather = w;
                weather.register(this);
        }

        @Override
        public void update(double temp) {
                this.temp = temp;
                display();
        }

        public void display(){
                // code to display horizontally
        }
}
```

```
class VerticalDisplay implements Observer, {
        double temp;
        Subject weather;

        public VerticalDisplay(Subject w) {
                weather = w;
                weather.register(this);
        }

        @Override
        public void update(double temp) {
                this.temp = temp;
                display();
        }

        public void display(){
                // code to display vertically

        }
}
```

You can assume there are more View classes that implement Observer

Implementation Step 5

- Create a client class the uses the observers:

```
public class Cient {
   public static void main(String []s) {
      Weather w = new Weather(34.5);
      HorizontalDisplay h = new HorizontalDisplay(w);
      VerticalDisplay v = new VerticalDisplay(w);
      w.setTemp(55);
      h.display();   // displays horizontally
      v.display();   // displays vertically
   }
}
```

How Easy is to Add New Observer?

```
class DiagonalDisplay implements Observer {
        double temp;
        Subject weather;

        public DiagonalDisplay(Subject w) {
                weather = w;
                weather.register(this);
        }

        @Override
        public void update(double temp) {
                this.temp = temp;
                display();
        }

        public void display(){
                // code to display temp diagonally
        }
}
```

```
public class Cient {
   public static void main(String []s) {
     Weather w = new Weather(34.5);
     HorizontalDisplay h =
                 new HorizontalDisplay(w);
     VerticalDisplay v = new VerticalDisplay(w);
     DiagonalDisplay d = new DiagonalDisplay(w);
     w.setTemp(55);
     h.display();   // displays horizontally
     v.display();   // displays vertically
     d.display();   // displays diagonally
   }
}
```