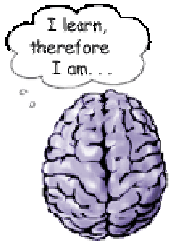
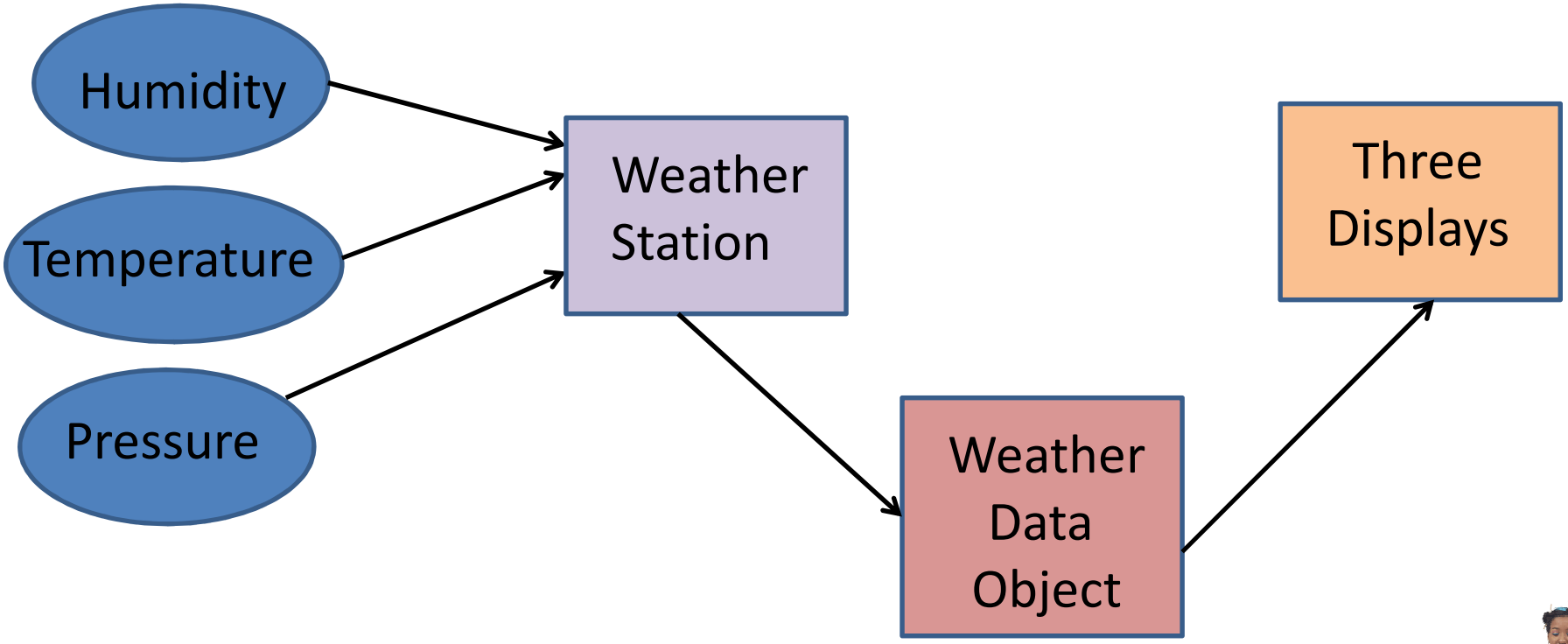
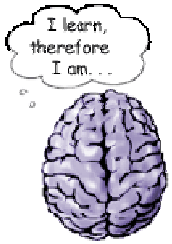
Chapter 2: The Observer Pattern



Consider the Following Application

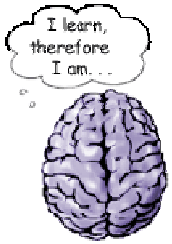
- Application specification





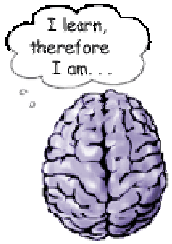
What's Given

- The WeatherData class has getter methods that obtain measurement values from temperature, humidity and pressure.
- The class has a *measurementsChanged()* method that updates the three values.
- Three displays must be implemented: current conditions, statistics and forecast display.
- System must be expandable – other display elements maybe added or removed.



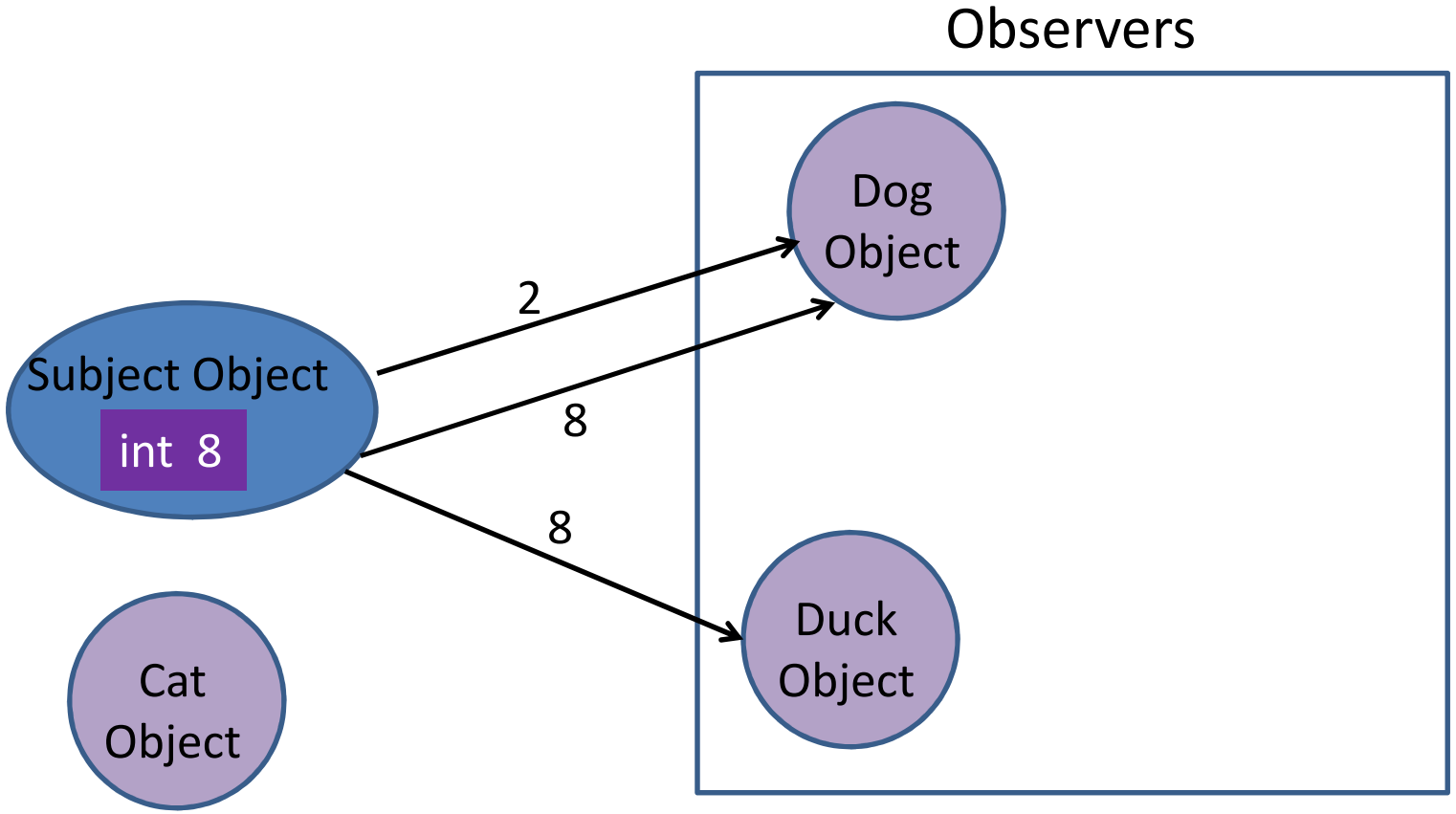
A First Attempt

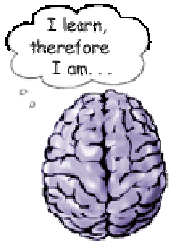
```
public class WeatherData{
   //instance variables
   public void measurementChanged() {
       float temp = getTemperature();
       float humidity = getHumidity();
       float pressure = getPressure();
       currentConditionsDisplay.update(temp,humidity,
                                       pressure)
       statisticsDisplay.update(temp,humidity,pressure)
       forecastDisplay.update(temp,humidity,pressure)
   }
   //other WeatherData methods here
}
```



The Observer Pattern "Observed"

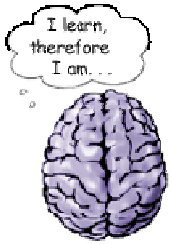
Subject Broadcasts ;



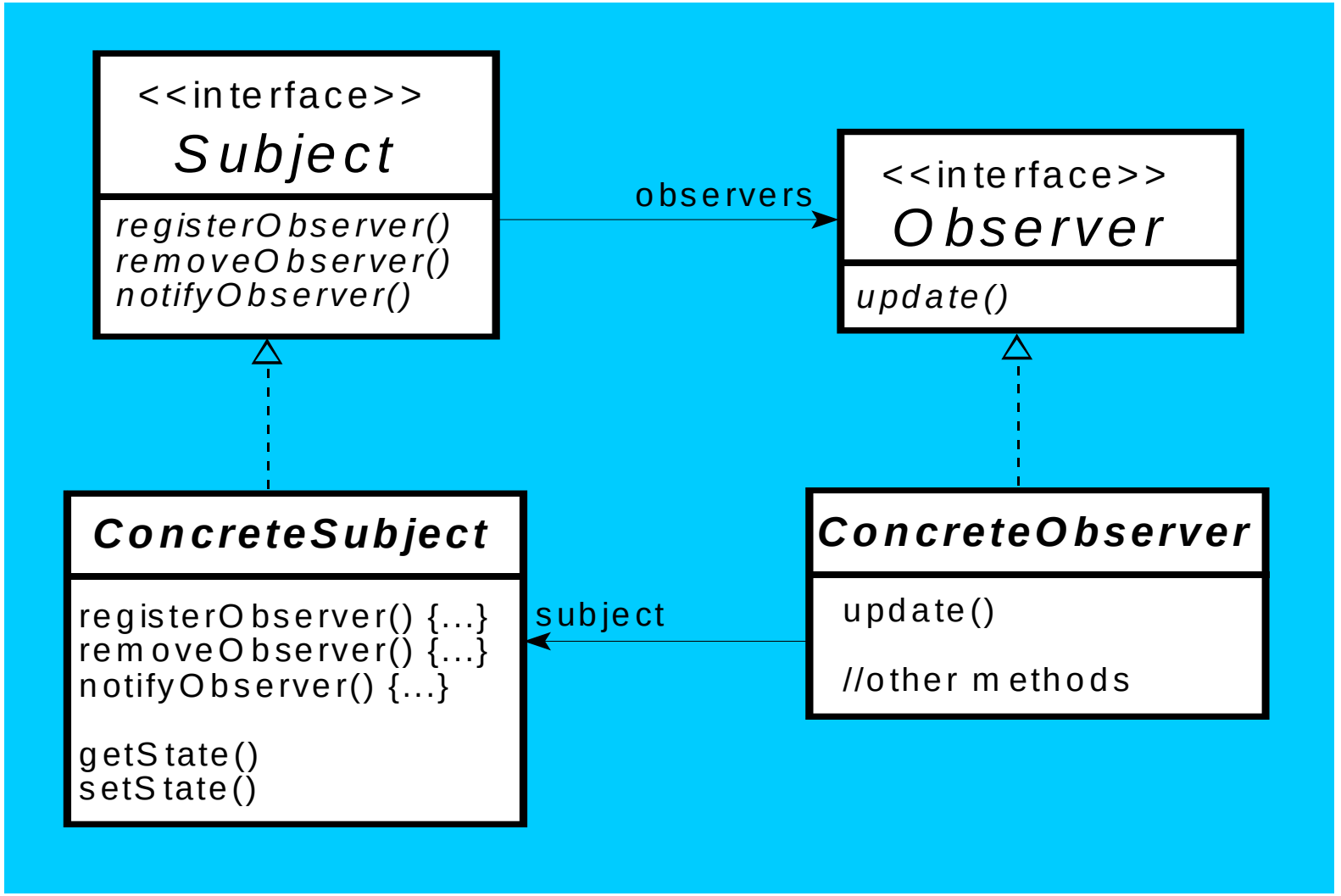


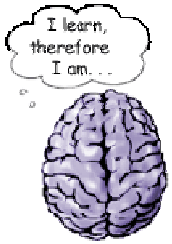
Definition

- The observer pattern implements a one-to-many relationship between a set of objects.
- A single object changes state and updates the objects (*dependants*) that are affected by the change.
- The object that changes state is called the *subject* and the other objects are the *observers*.



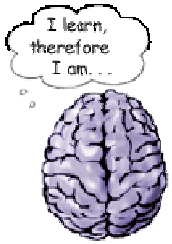
The Class Diagram





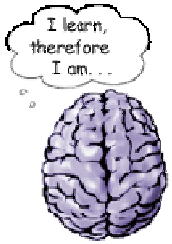
Loose Coupling

- Subjects and observers are loosely coupled.
- The subject only knows the observer interface and not its implementation.
- Observers can be added and removed at any time.
- In adding new observers the subject does not need to be modified.
- Subjects and observers can be reused independently.
- Changes to the subject or observer will not affect the other.



Design Principle

- Strive for loosely coupled designs between objects that interact.
- Loosely coupled designs allow us to build flexible object-oriented systems.
- These systems can handle change because they minimize the interdependency between objects.



Weather Station Class Diagram