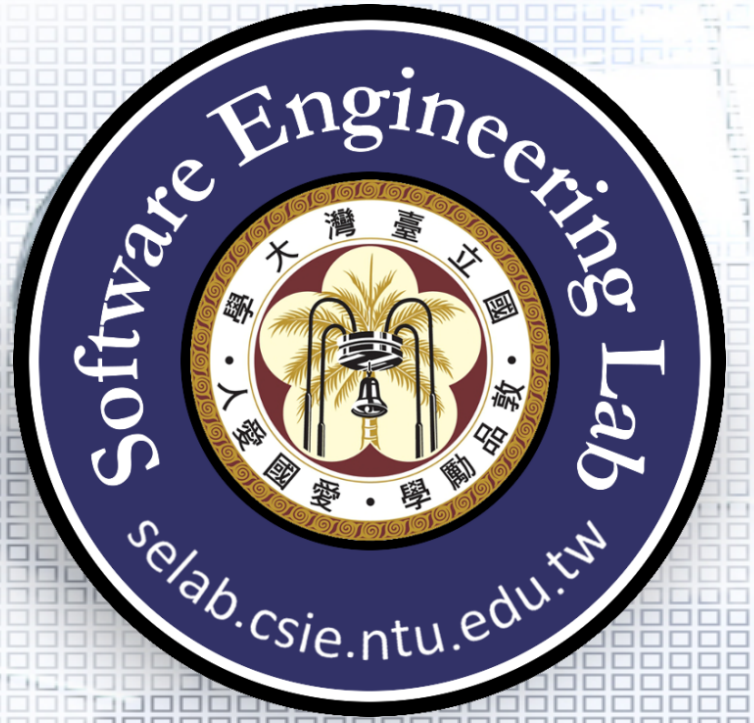
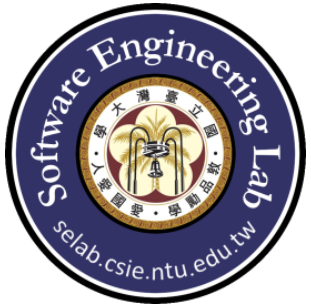


Observer Pattern

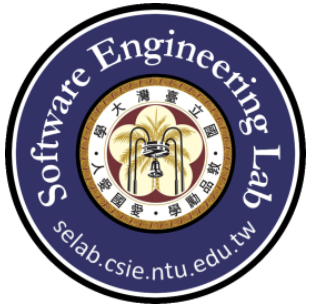
Prof. Jonathan Lee (李允中)
Department of CSIE
National Taiwan University



Design Aspect of Observer

Number of objects that depend on another object; how the dependent objects stay up to date

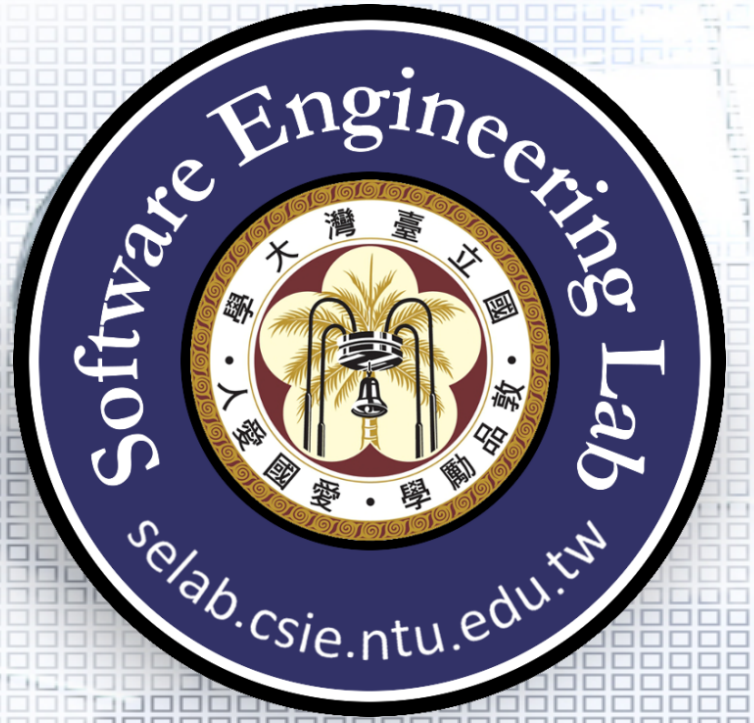
©2017 Jonathan Lee, CSIE Department, National Taiwan University.



Outline

- Spreadsheet Application Requirements Statements
- Initial Design and Its Problems
- Design Process
- Refactored Design after Design Process
- Weather Monitoring Station System: Another Example
- Recurrent Problems
- Intent
- Observer Pattern Structure
- Homework

©2017 Jonathan Lee, CSIE Department, National Taiwan University.



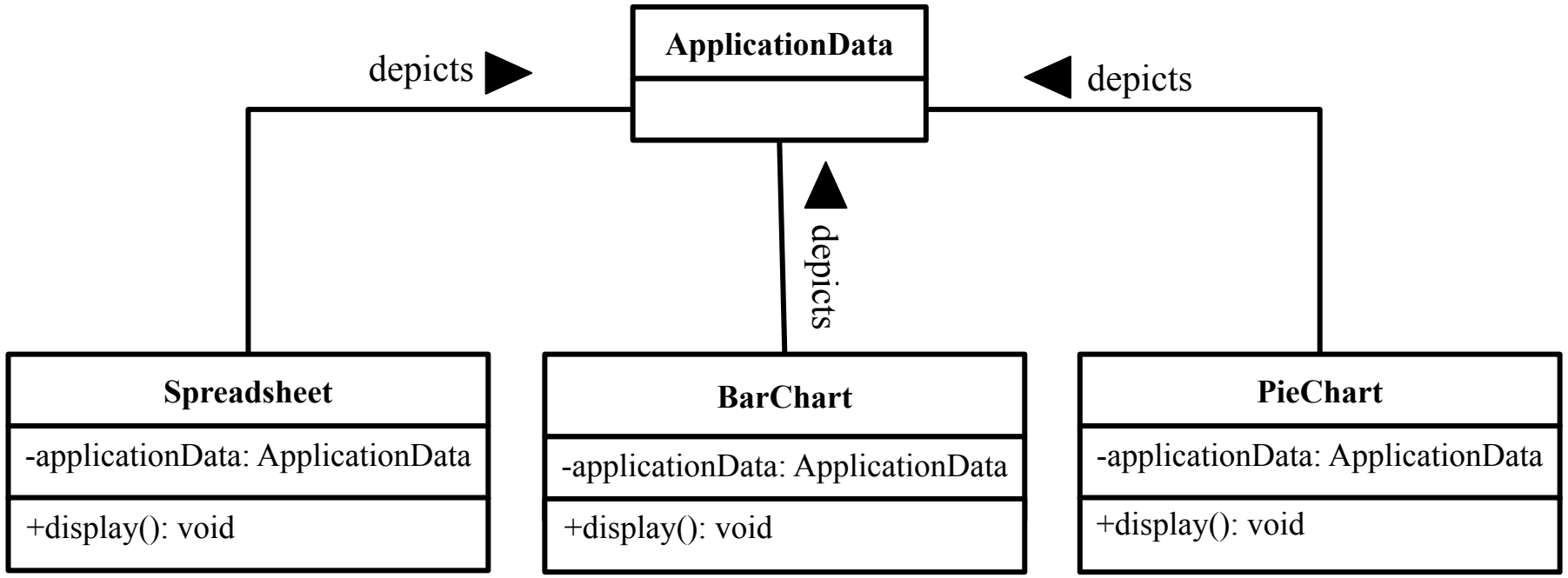
Spreadsheet Application (Observer)

Prof. Jonathan Lee (李允中)

Department of Computer Science and Information Engineering
National Taiwan University

Requirements Statements$_1$

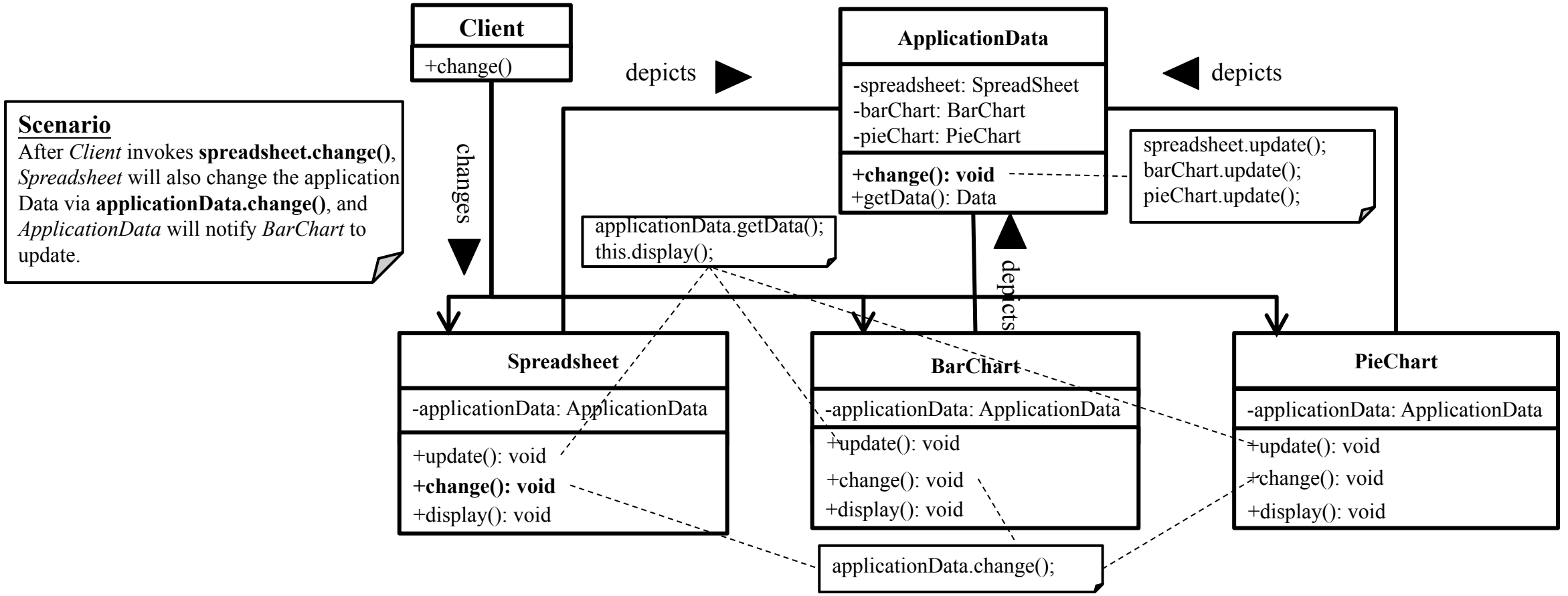
- In a spreadsheet application,
  - A spreadsheet object, bar chart object, and pie chart object can depict information in the same application data object by using different presentations.



©2017 Jonathan Lee, CSIE Department, National Taiwan University.

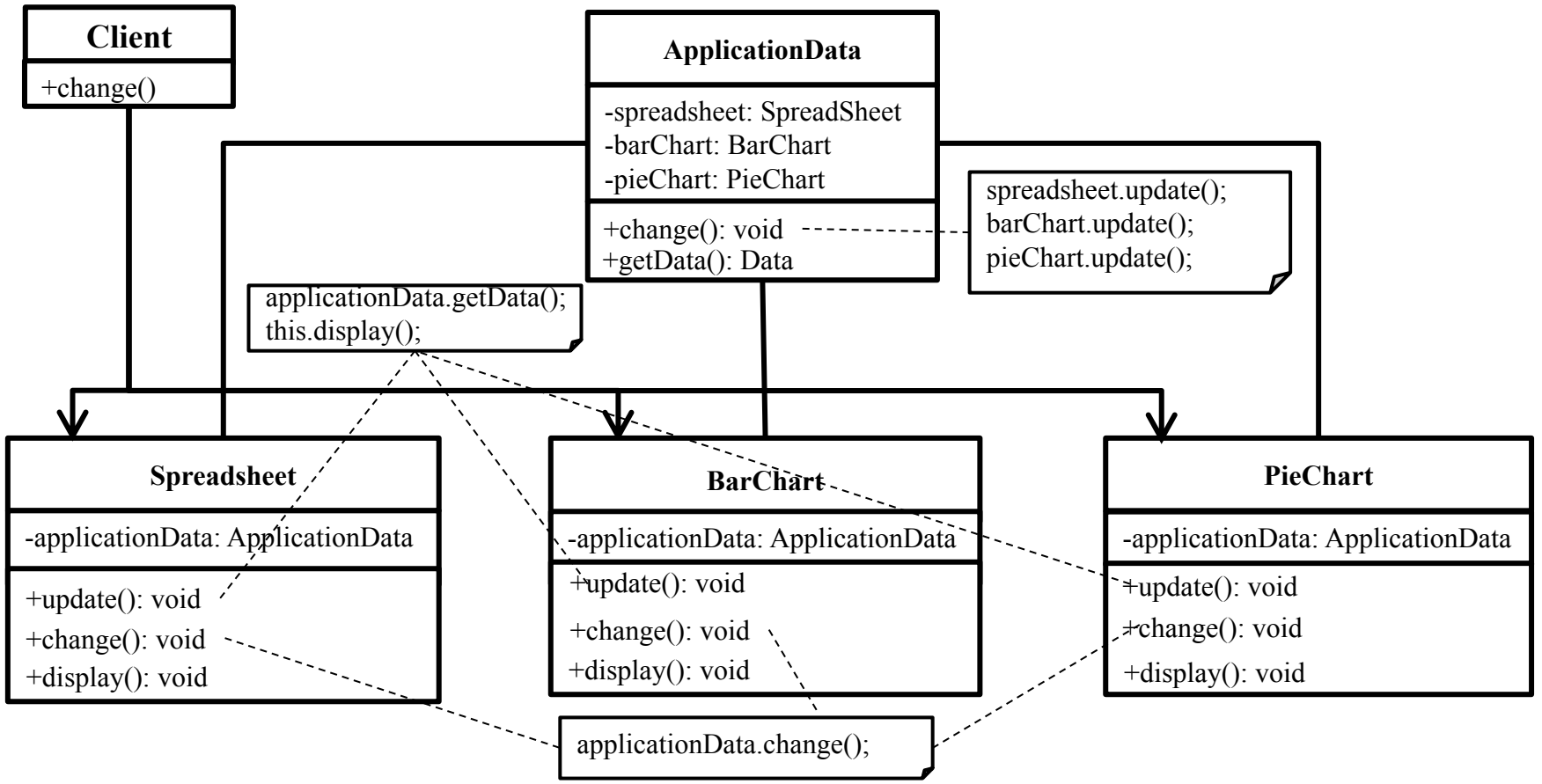
Requirements Statements$_2$

- When the user changes the information in the spreadsheet, the bar chart reflects the changes immediately, and vice versa.



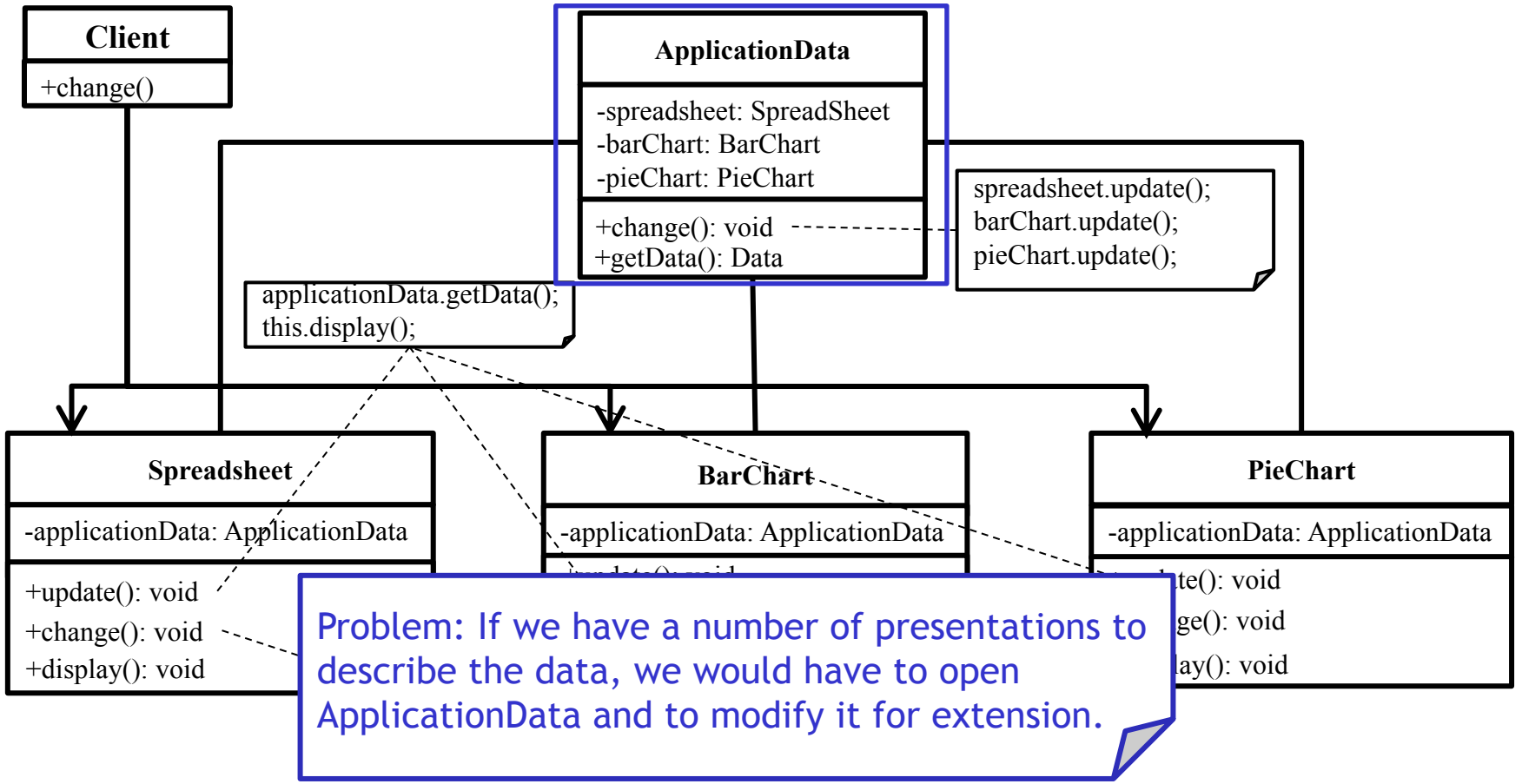
©2017 Jonathan Lee, CSIE Department, National Taiwan University.

Initial Design



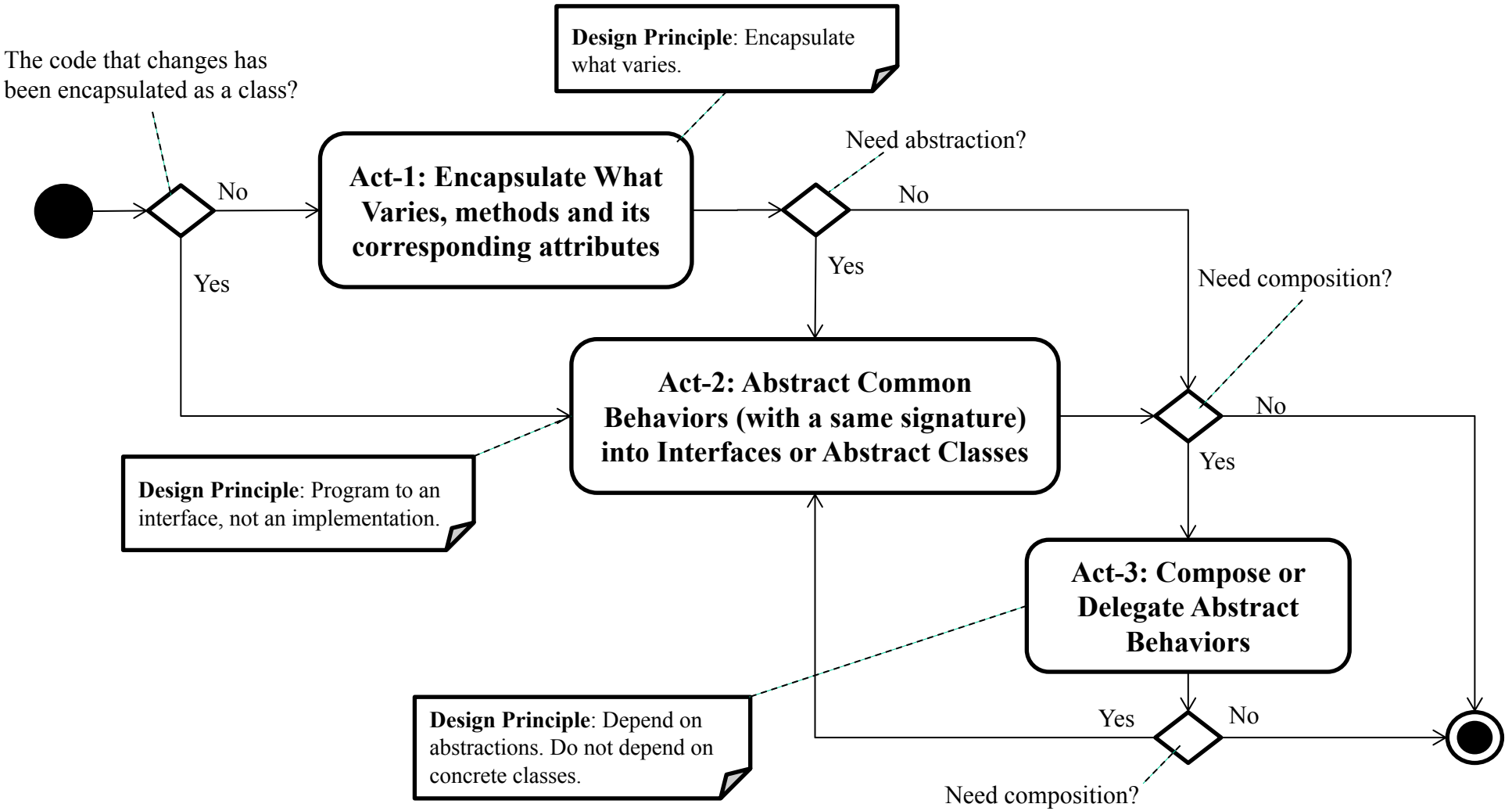
©2017 Jonathan Lee, CSIE Department, National Taiwan University.

Problems with Initial Design



©2017 Jonathan Lee, CSIE Department, National Taiwan University.

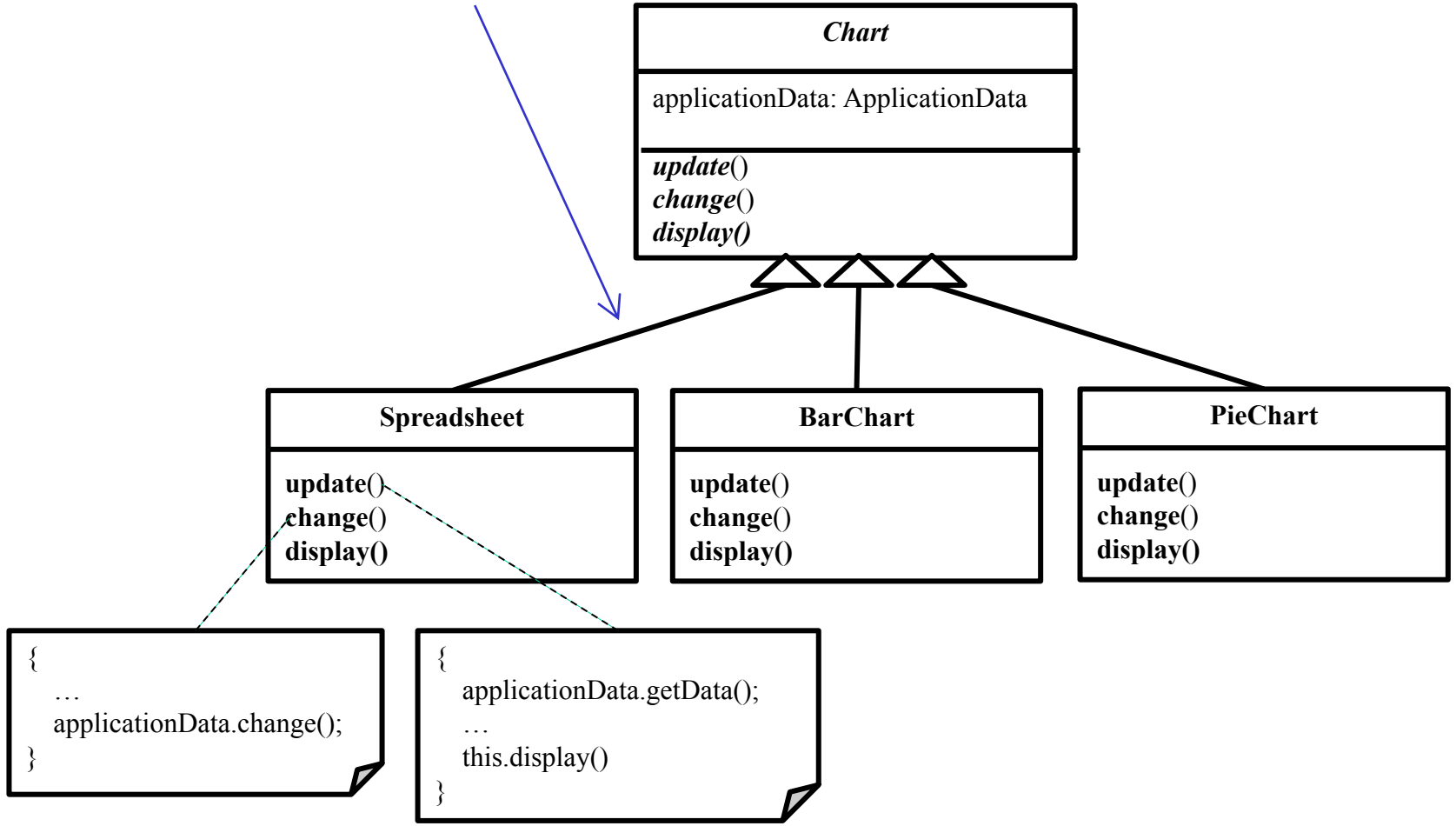
Design Process for Change



©2017 Jonathan Lee, CSIE Department, National Taiwan University.

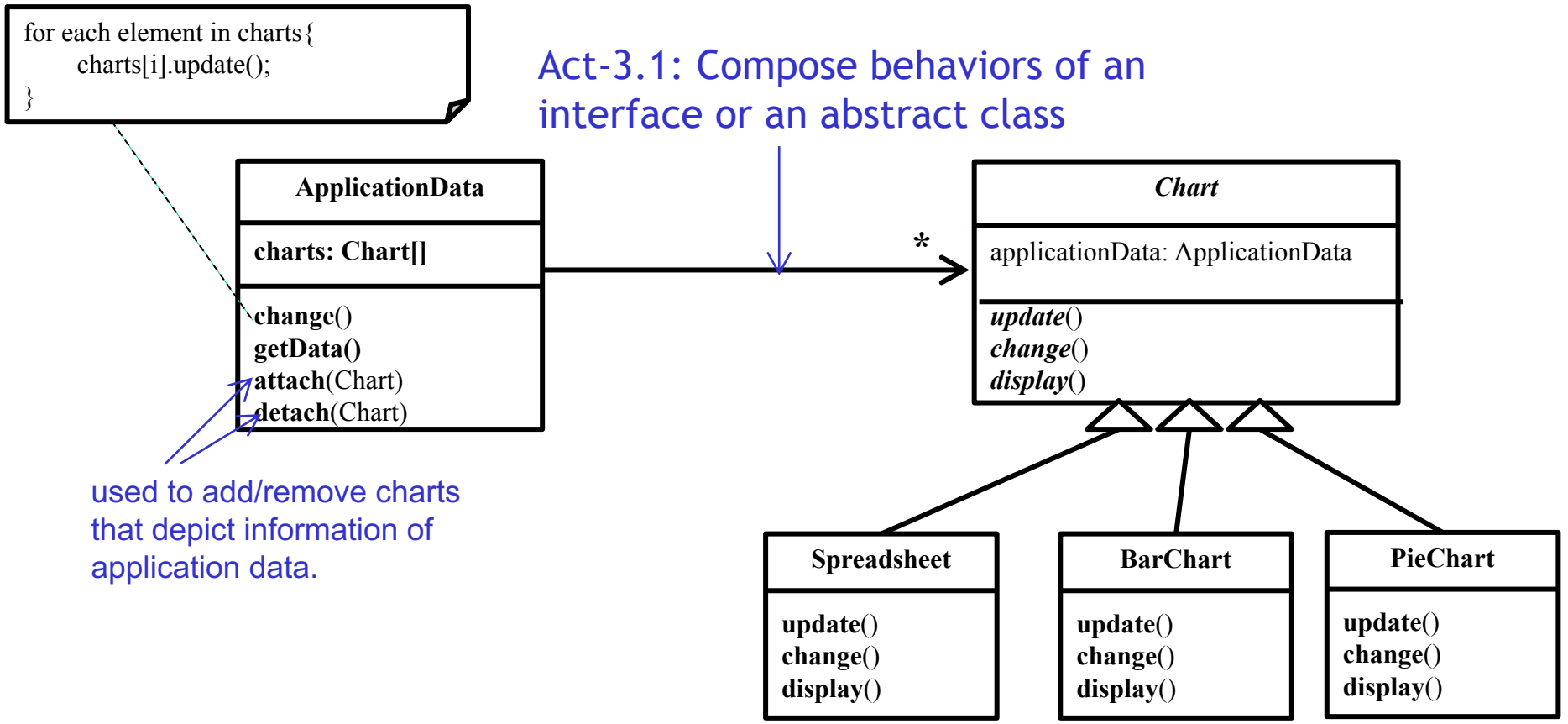
Act-2: Abstract Common Behaviors

Act-2.2: Abstract common behaviors with a same signature into abstract class through inheritance



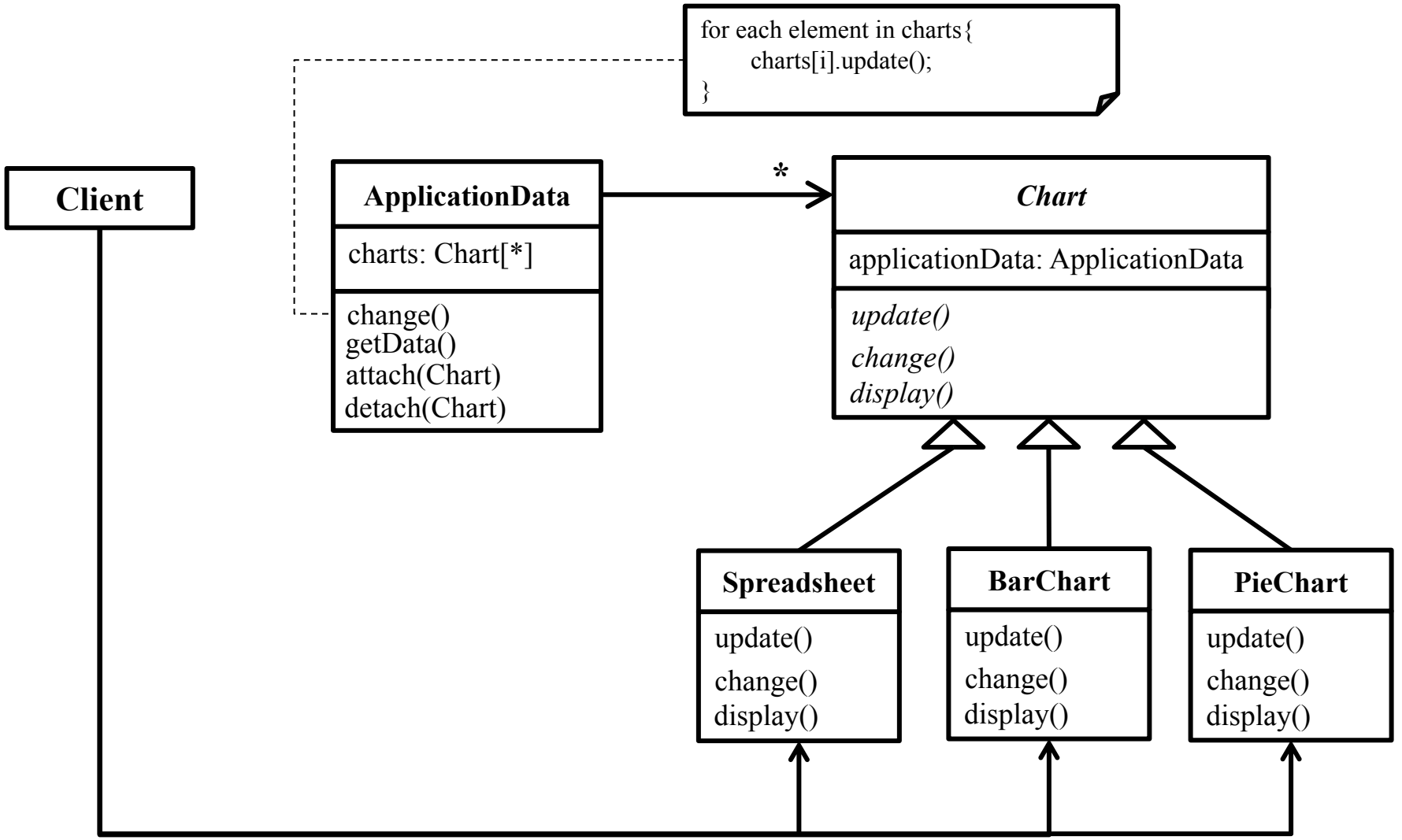
©2017 Jonathan Lee, CSIE Department, National Taiwan University.

Act-3: Compose Abstract Behaviors

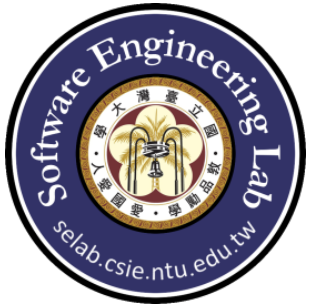


©2017 Jonathan Lee, CSIE Department, National Taiwan University.

Refactored Design



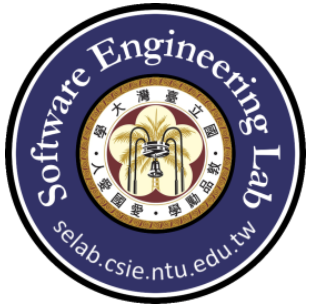
©2017 Jonathan Lee, CSIE Department, National Taiwan University.



Recurrent Problem

- The code will be modified if a new display/presentation is to be added.
  - Many graphical user interface toolkits **separate** the presentational aspects of the user interface from the underlying application data.
  - Classes defining application data and presentations can be reused independently.

©2017 Jonathan Lee, CSIE Department, National Taiwan University.

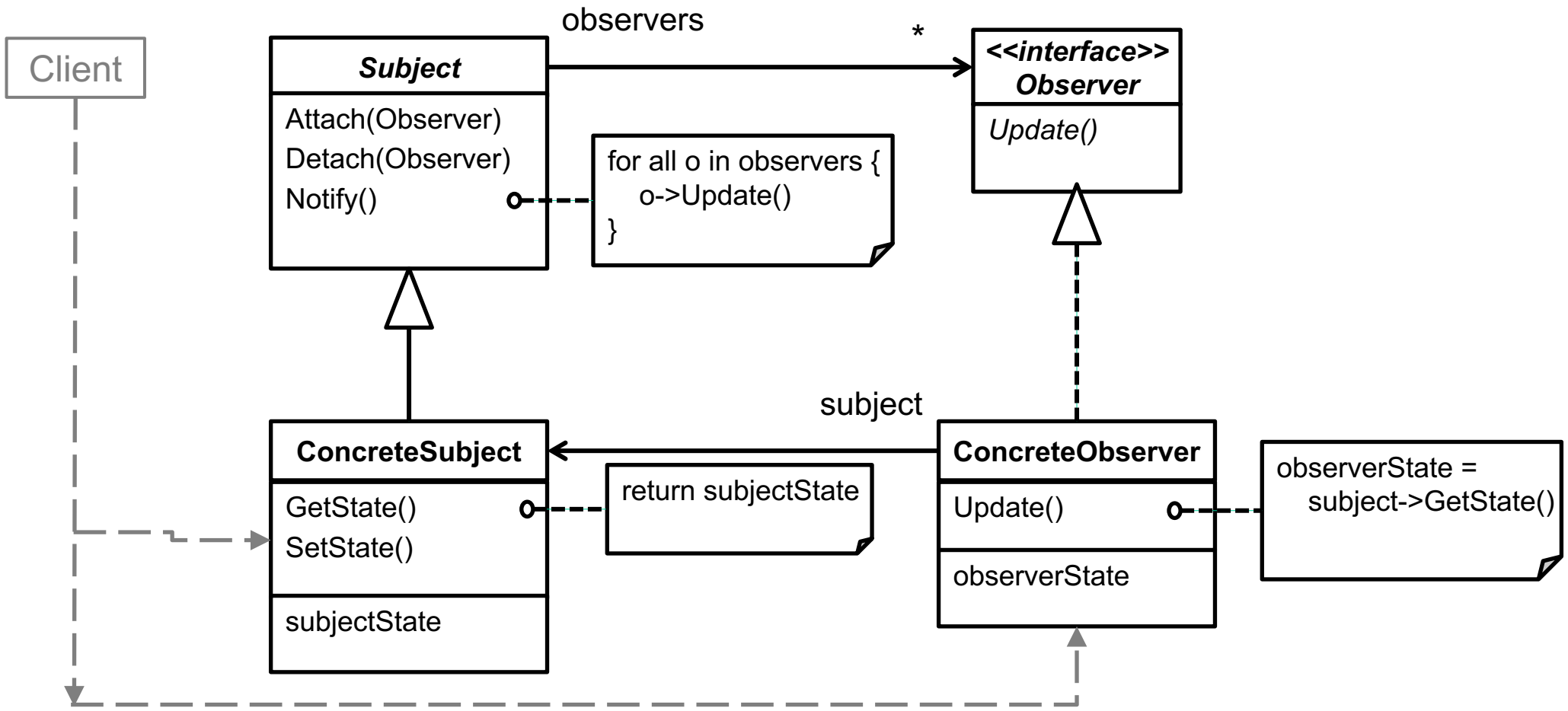


Observer Pattern

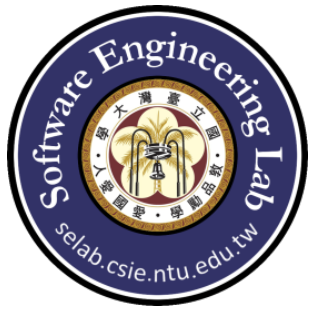
- Intent
  - Define a one-to-many dependency between objects so that when one object changes state, all its dependents are **notified and updated** automatically.

©2017 Jonathan Lee, CSIE Department, National Taiwan University.

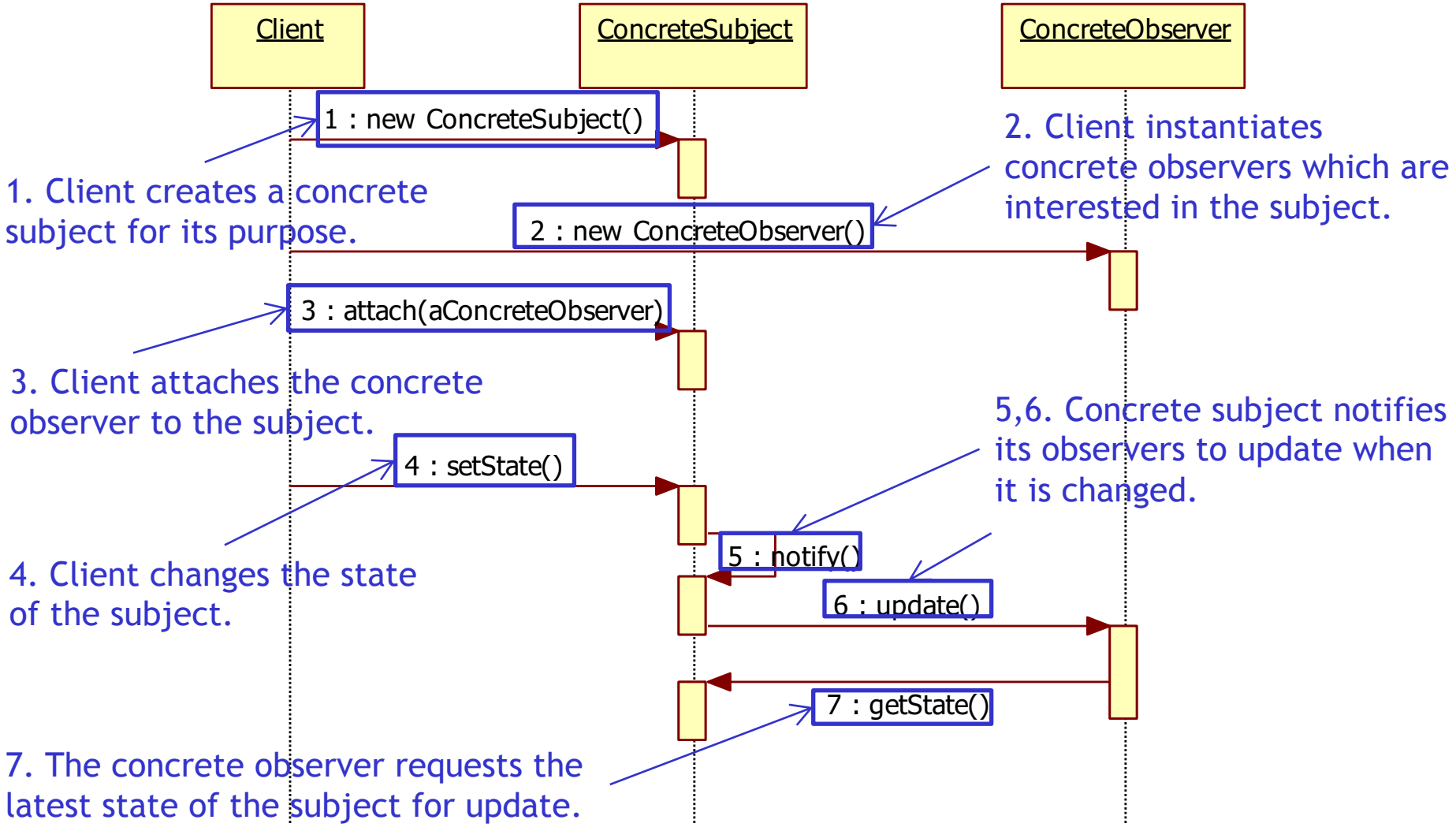
Observer Pattern Structure$_1$



©2017 Jonathan Lee, CSIE Department, National Taiwan University.



Observer Pattern Structure$_2$

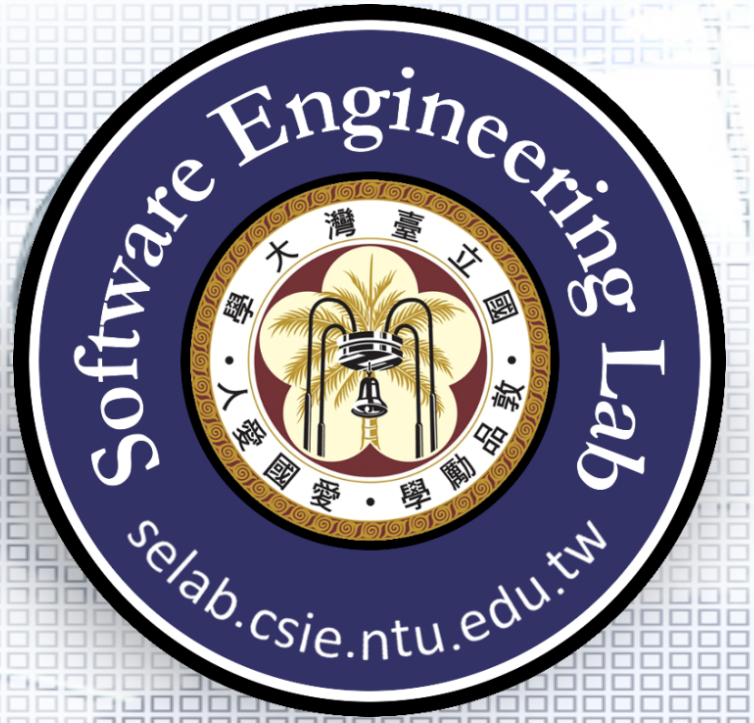


©2017 Jonathan Lee, CSIE Department, National Taiwan University.

Observer Pattern Structure$_3$

	Instantiation	Use	Termination
Client	Other class except classes in observer pattern	Other class except classes in the observer pattern	Other class except classes in the observer pattern
Subject	X	Client uses this abstract class to attach and detach ConcreteObserver	X
Concrete Subject	The client class or other class except classes in the observer pattern	ConcreteObserver or other classes use this class to set/get state of ConcreteSubject	Classes who hold the reference of ConcreteSubject
Observer	X	Subject uses this interface to notify ConcreteObserver to update the state of ConcreteSubject through polymorphism	X
Concrete Observer	The client class or other class except classes in the observer pattern	Subject notifies this class which is attached by client to update the state of ConcreteSubject through polymorphism	Classes who hold the reference of ConcreteObserver

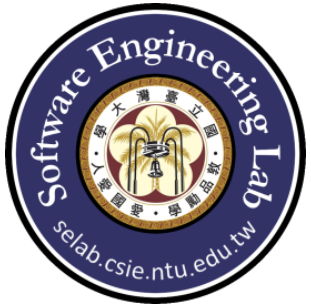
©2017 Jonathan Lee, CSIE Department, National Taiwan University.



Weather Monitoring Station System

Prof. Jonathan Lee (李允中)

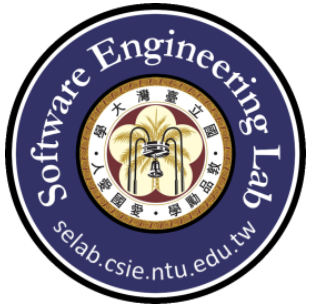
Department of Computer Science and Information Engineering
National Taiwan University



Requirements Statements

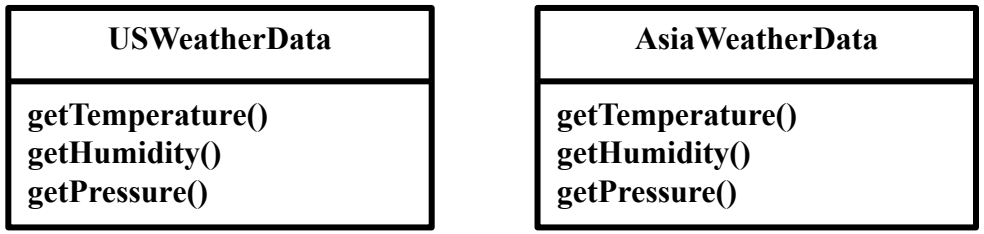
- Weather Monitoring Station System
  - The system will be based on WeatherData object, which tracks current weather conditions (temperature, humidity, and barometric pressure) in a specific area (e.g. U.S. or Asia).
  - The system initially provides three display elements: current conditions, weather statistics and a simple forecast, all updated in real time as the WeatherData object acquires the most recent measurements.
  - The system should supply an API so that other developers can write their own weather displays and plug them right in.

©2017 Jonathan Lee, CSIE Department, National Taiwan University.



Requirements Statements$_1$

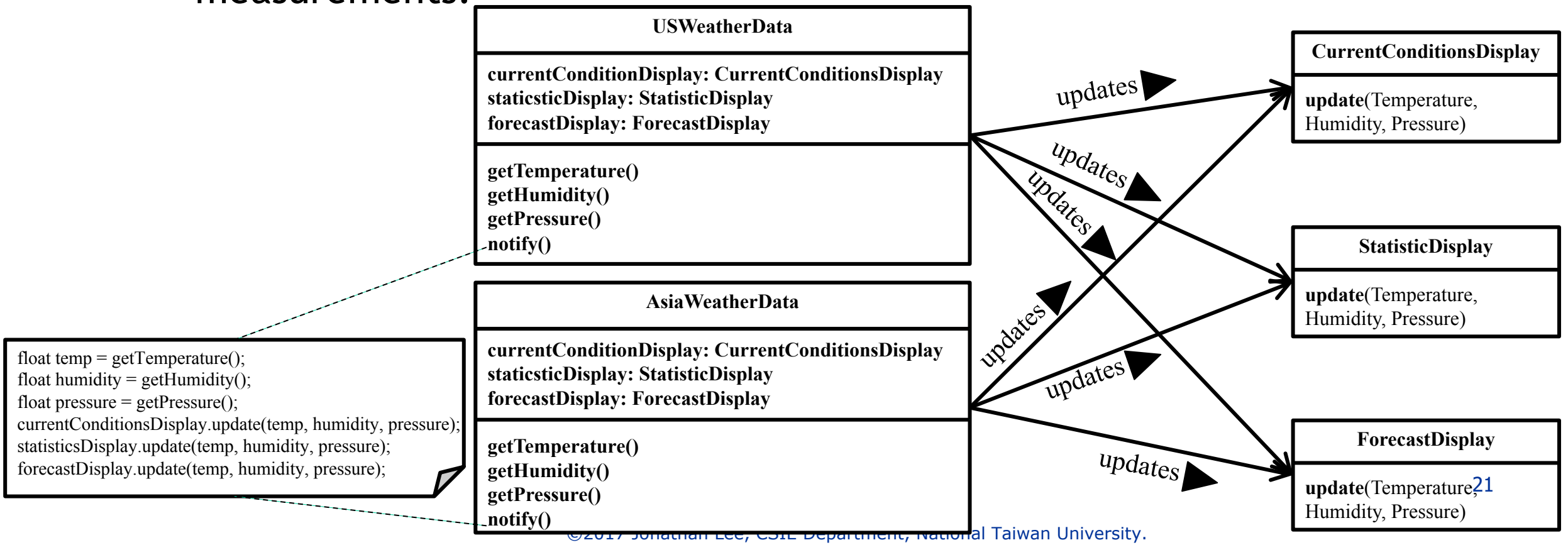
- The system will be based on WeatherData object to track current weather conditions, including temperature, humidity, and barometric pressure, in a designated area (e.g. U.S. or Asia).



©2017 Jonathan Lee, CSIE Department, National Taiwan University.

Requirements Statements$_2$

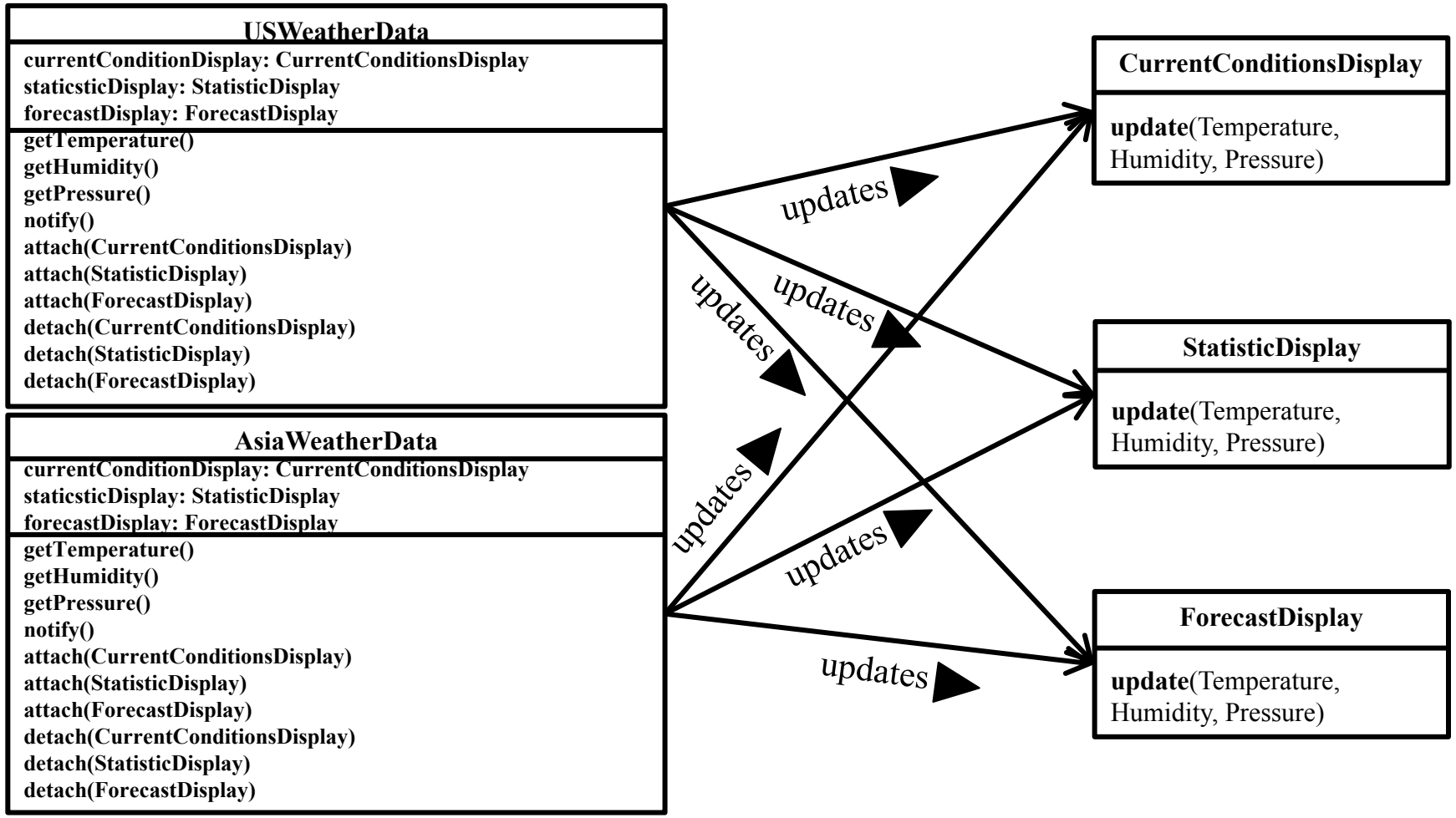
- The system initially provides three display elements: current conditions, weather statistics, and a simple forecast, all updated in real time as the WeatherData object acquires the most recent measurements.



©2017 Jonathan Lee, CSIE Department, National Taiwan University.

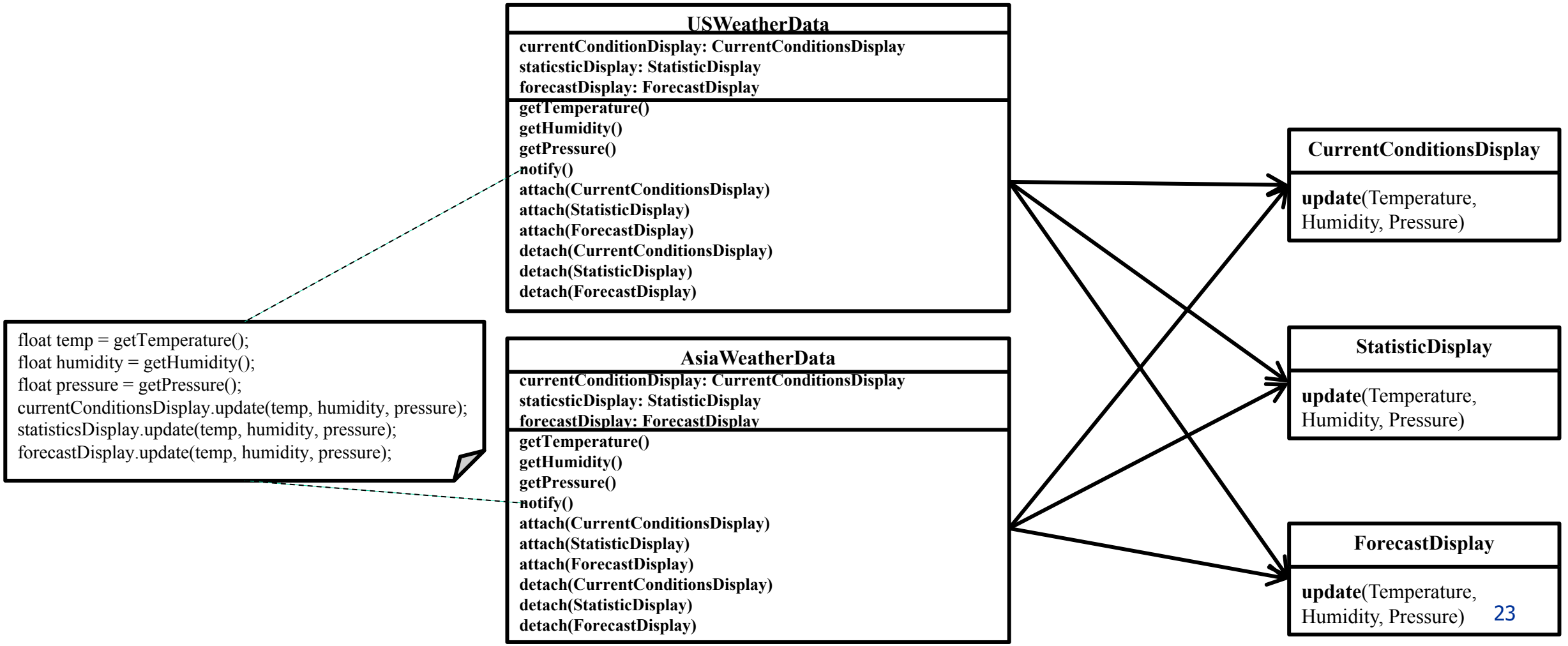
Requirements Statements$_3$

- The system supplies methods to plug the three displays in or out.



©2017 Jonathan Lee, CSIE Department, National Taiwan University.

Initial Design - Class Diagram



©2017 Jonathan Lee, CSIE Department, National Taiwan University.

Problems with Initial Design

Problem: Code will be modified if a new display is added.

USWeatherData

currentConditionDisplay: CurrentConditionsDisplay
staticsticDisplay: StatisticDisplay
forecastDisplay: ForecastDisplay

getTemperature()
getHumidity()
getPressure()
notify()
attach(CurrentConditionsDisplay)
attach(StatisticDisplay)
attach(ForecastDisplay)
detach(CurrentConditionsDisplay)
detach(StatisticDisplay)
detach(ForecastDisplay)

```
float temp = getTemperature();
float humidity = getHumidity();
float pressure = getPressure();
currentConditionsDisplay.update(temp, humidity, pressure);
statisticsDisplay.update(temp, humidity, pressure);
forecastDisplay.update(temp, humidity, pressure);
```

AsiaWeatherData

currentConditionDisplay: CurrentConditionsDisplay
staticsticDisplay: StatisticDisplay
forecastDisplay: ForecastDisplay

getTemperature()
getHumidity()
getPressure()
notify()
attach(CurrentConditionsDisplay)
attach(StatisticDisplay)
attach(ForecastDisplay)
detach(CurrentConditionsDisplay)
detach(StatisticDisplay)
detach(ForecastDisplay)

CurrentConditionsDisplay

update(Temperature, Humidity, Pressure)

StatisticDisplay

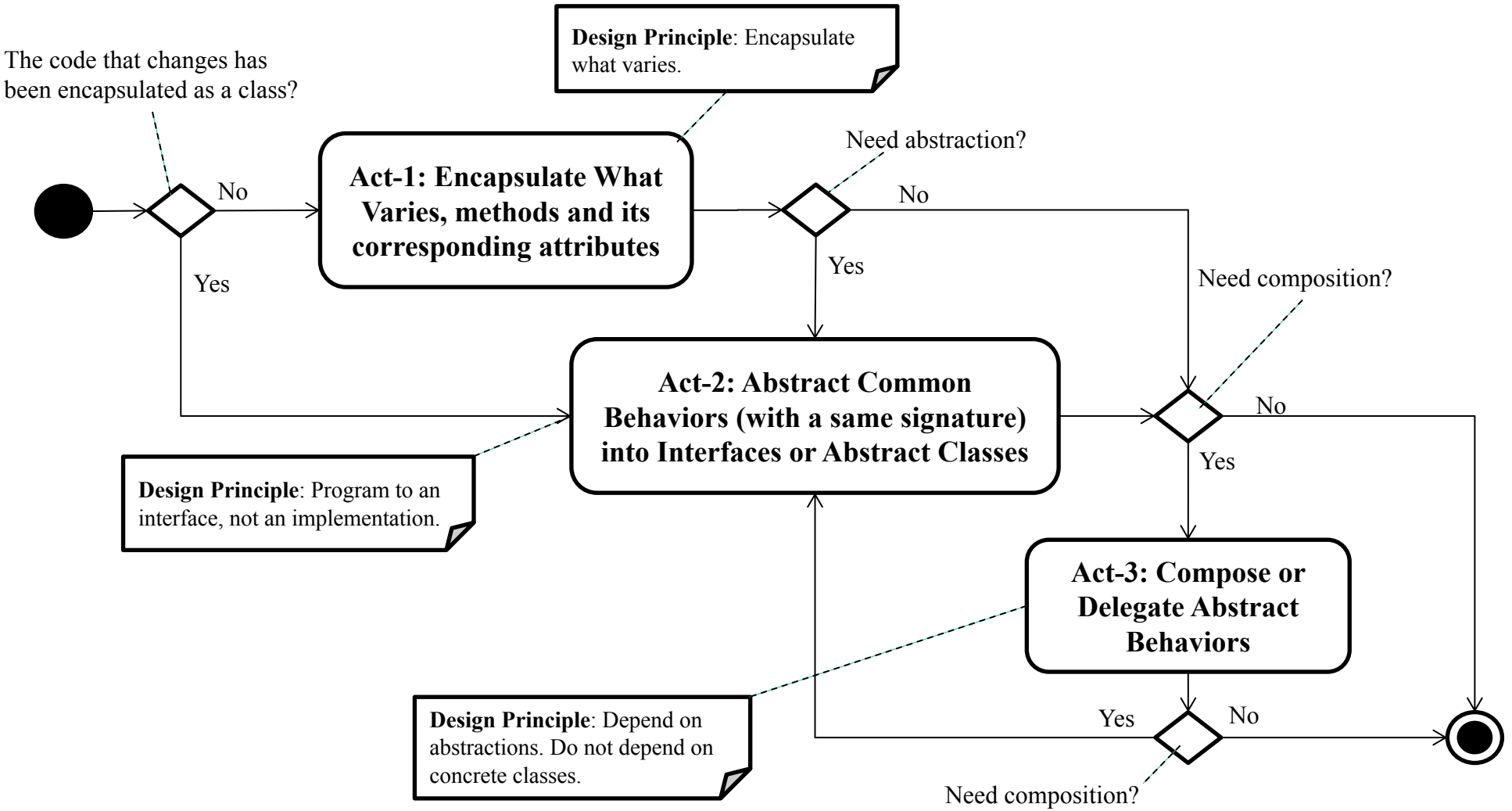
update(Temperature, Humidity, Pressure)

ForecastDisplay

update(Temperature, Humidity, Pressure)

©2017 Jonathan Lee, CSIE Department, National Taiwan University.

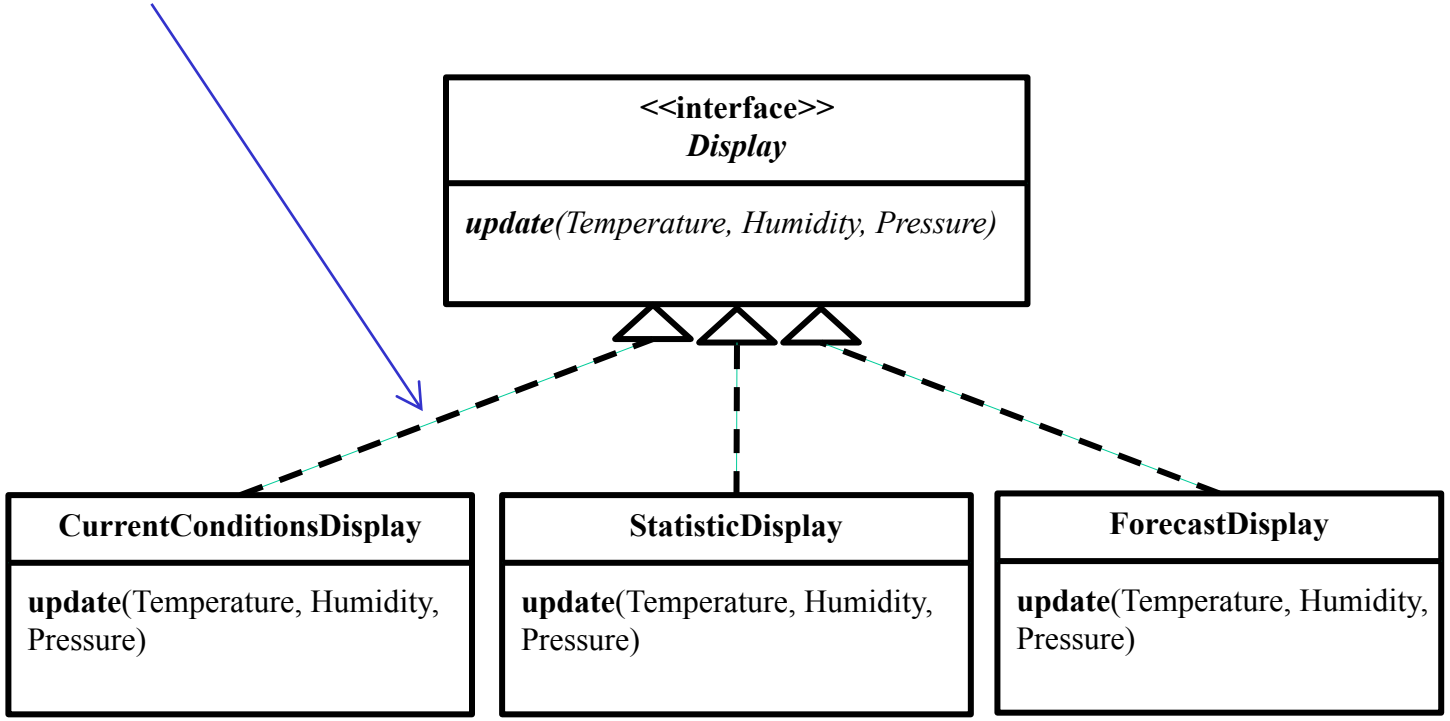
Design Process for Change



©2017 Jonathan Lee, CSIE Department, National Taiwan University.

Act-2: Abstract Common Behaviors

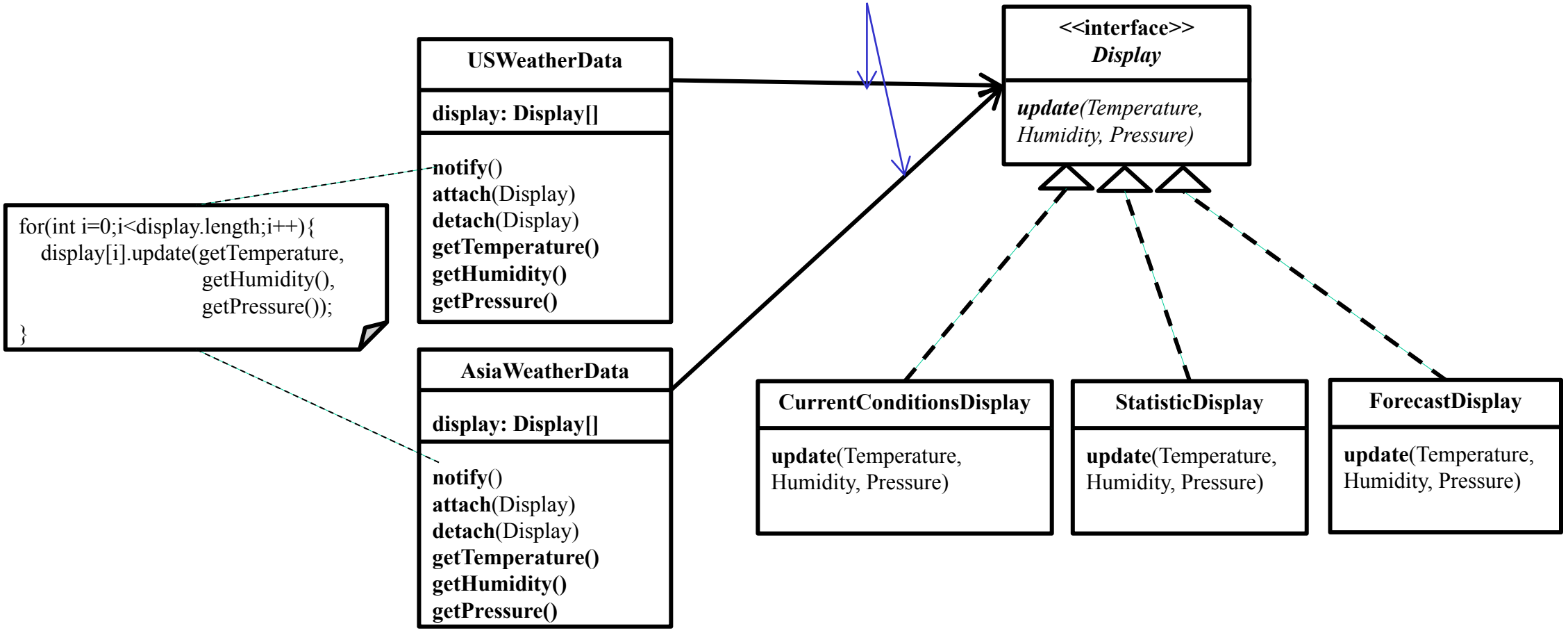
Act-2.1: Abstract common behaviors with a same signature into interface through polymorphism



©2017 Jonathan Lee, CSIE Department, National Taiwan University.

Act-3: Compose Abstract Behaviors

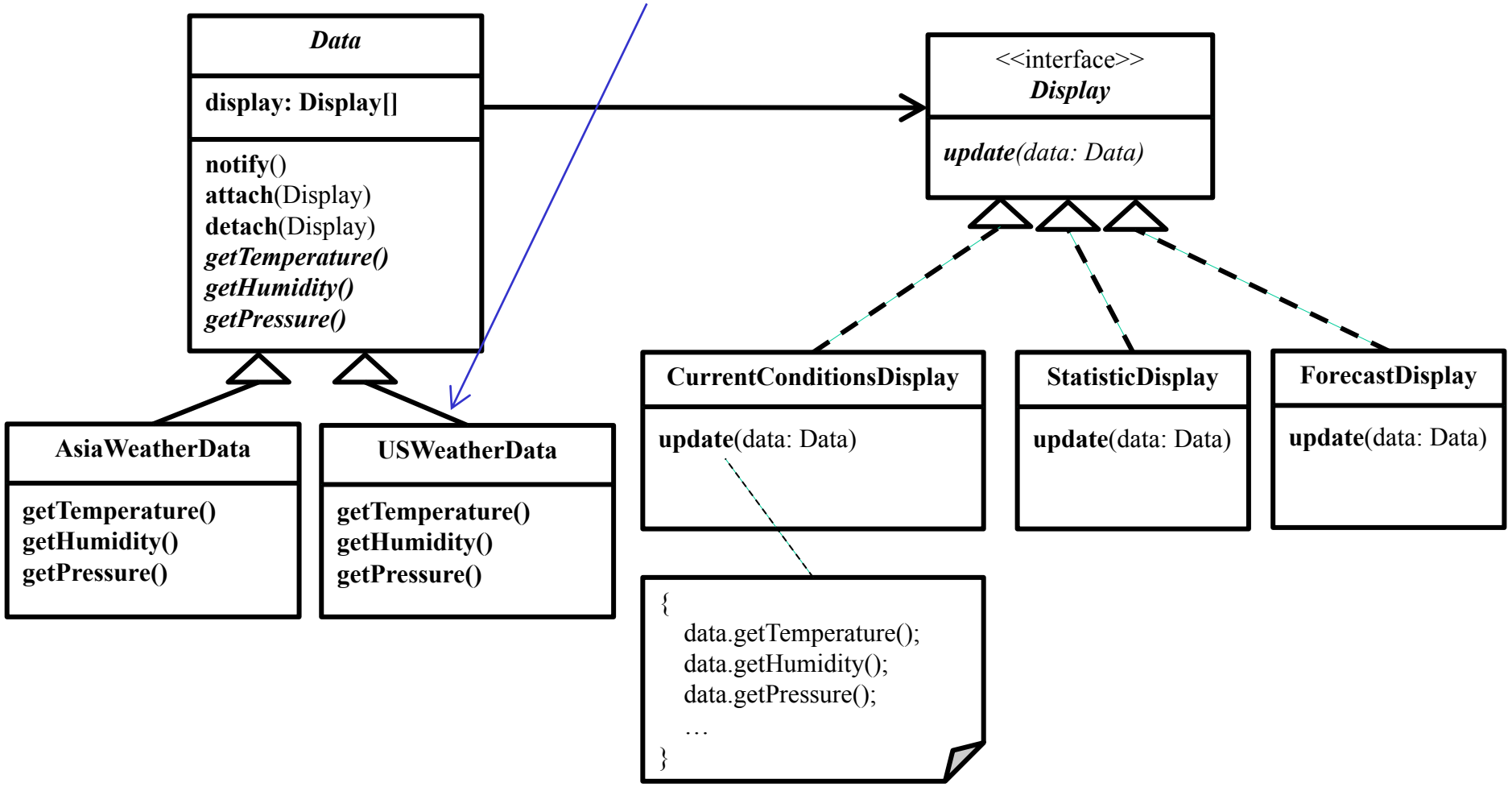
Act-3.1: Compose behaviors of an interface or an abstract class



©2017 Jonathan Lee, CSIE Department, National Taiwan University.

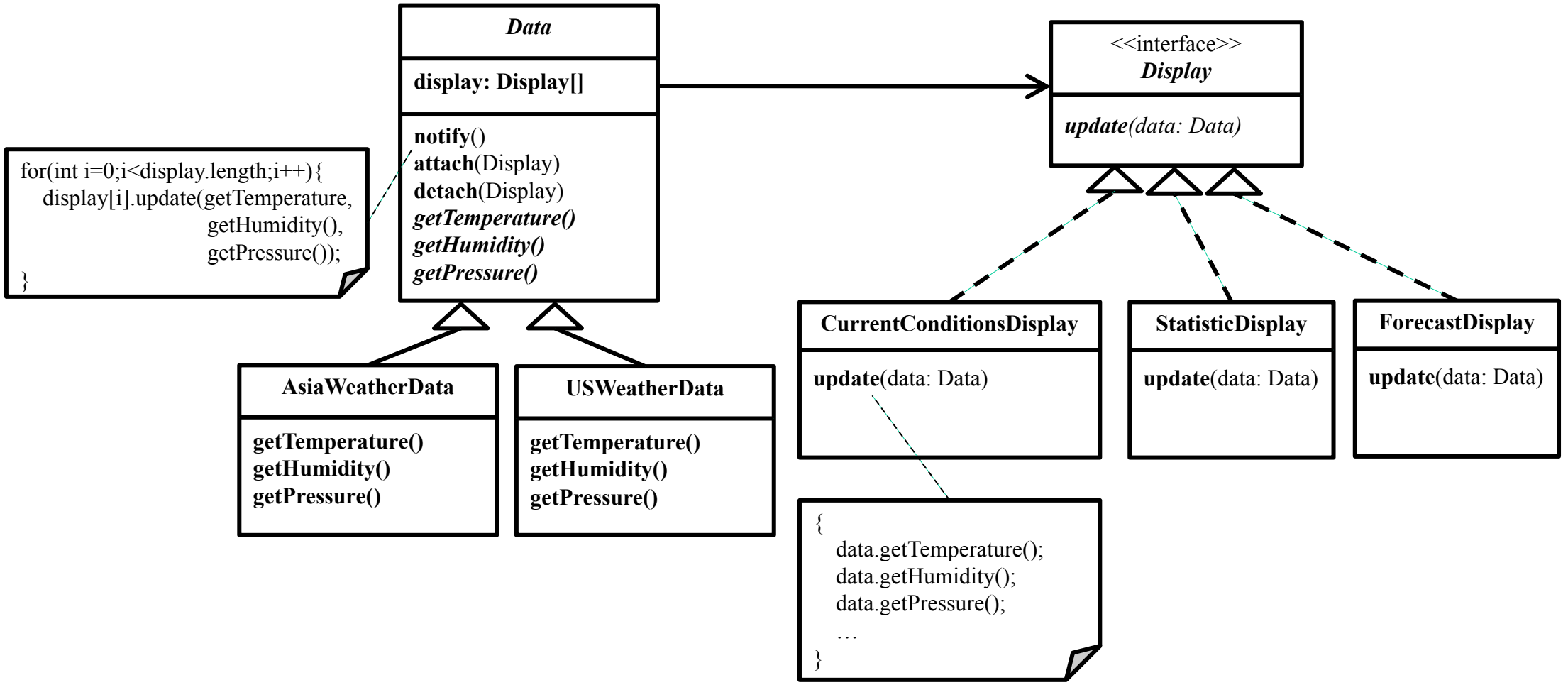
Act-2: Abstract Common Behaviors

Act-2.2: Abstract common behaviors with a same signature into abstract class through inheritance

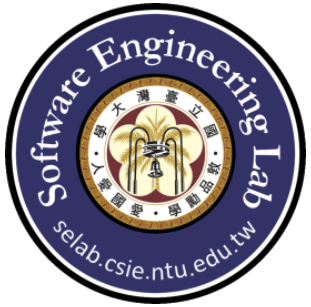


©2017 Jonathan Lee, CSIE Department, National Taiwan University.

Refactored Design after Design Process



©2017 Jonathan Lee, CSIE Department, National Taiwan University.



Homework 1: Requirements Statements

- In FileViewer,
  - We have a TextView object that displays text in a window.
  - TextView has no scroll bars by default, because we might not always need them.
  - We can also add a thick black border around the TextView.
  - It is highly likely that we will support various file formats for display in the future.

©2017 Jonathan Lee, CSIE Department, National Taiwan University.