Software Engineering Design

Spring 2017

Prof. Jonathan Lee (李允中)

Department of Computer Science and Information Engineering
National Taiwan University

CSIE 5734

- Instructor: Prof. Jonathan Lee (李允中)
- Office: CS Building Room 513
- Email: jlee@csie.ntu.edu.tw

- Class Hours: Tuesday 13:20 – 16:10, CS Building Room 110

- Office Hours: Tuesday 16:30 – 17:30 or by appointment

Grading

- Class attendance, participation, homework (every week), and web frameworks presentation (40%)

- Term project (25%)
  - Open source refactoring: Presentation, Requirements Statements, WBS, Meeting minutes, Class Diagram Design, and Coding (6/13)

- Quiz (10%)

- Final Exam (25%) (6/20)

Course Materials & Reference

- Course materials and slides at URL: https://ceiba.ntu.edu.tw/1022_csie_sed
- Git repository at URL: ssh://[學號]@140.112.90.151:10000/srv/repo/sed102/team[組別].git
- “*Design Patterns: Elements of Reusable Object-Oriented Software,*” Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Addison-Wesley Professional, 1995.
- “*Head First Design Patterns,*” Eric Freeman, Elisabeth Freeman, Kathy Sierra, and Bert Bates, O'Reilly Media, 2004.
- “*UML 2.0 in a Nutshell,*” Dan Pilone and Neil Pitman, O'Reilly Media, 2005.
- 李允中, 軟體工程, 台灣軟體工程學會, 2013.

Course Outline

(6/13) Term project presentation and demonstration (6/20) Final Exam

Design Patterns (optional)

Flyweight Pattern | Interpreter Pattern | Bridge Pattern | Prototype Pattern | Proxy Pattern | Adapter Pattern | Visitor Pattern

(3/28, 4/11, 4/18, 5/2, 5/9, 5/16, 5/23, 6/6) Design Patterns

(4/25) Quiz

Command Pattern	Composite Pattern	Decorator Pattern	Iterator Pattern
Observer Pattern	Template Pattern	Facade Pattern	Mediator Pattern
Singleton Pattern	Memento Pattern	Strategy Pattern	Builder Pattern
Factory Method Pattern	Abstract Factory Pattern	State Pattern	Chain of Responsibility Pattern

(3/21) Basic Design Concepts — Design Pattern Concepts

(3/14) Software Engineering Practices — Project Execution Plan | Project Monitoring & Control | System Requirements Document

(2/21, 3/7) Syllabus, Object-Oriented Concepts and UML

Software Design

- Starting with problem statements
- Modeling with class diagrams
- Refactoring with a design process involving the use of:
  - object-oriented concepts,
  - design principles, and
  - design patterns.

Object-Oriented Concepts

- ❑ Inheritance
- ❑ Polymorphism
- ❑ Abstraction
- ❑ Encapsulation
- ❑ Delegation
- ❑ Composition

OO Design Principles

- Inherit the most important features and delegate the rest
- Encapsulate what varies
- Favor composition over inheritance
- Program to interface, not implementation
- Strive for loosely coupled designs between objects that interact
- Classes should be open for extension but closed for modification
- Depend on abstractions. Do not depend on concrete classes
- Only interact with close classes
- ………

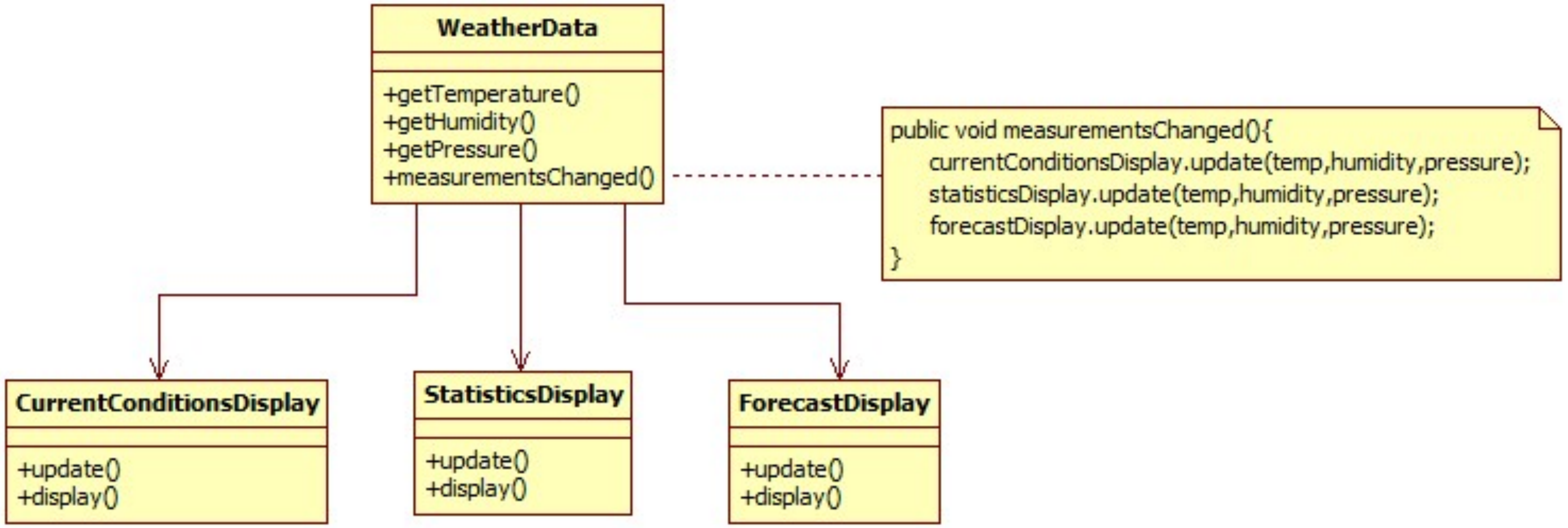
Design Patterns (GoF)

- Creational: Involve object creation.
  - Factory Method, Abstract Factory, Builder, Prototype, and Singleton.
- Structural: Compose classes or objects into larger structures.
  - Adapter, Bridge, Composite, Decorator, Façade, Flyweight, and Proxy.
- Behavioral: Concern with how classes and objects interact and distribute responsibility.
  - Interpreter, Template Method, Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, and Visitor.

Software Modeling

- Problem statement
  - To build our next generation Internet-base Weather Monitoring Station! The weather station will be based on the WeatherData object, which tracks current weather conditions (temperature, humidity and barometric pressure).
  - We'd like for you to create an application that initially provides three display elements: current conditions, weather statistics and a simple forecast, all updated in real time as the WeatherData object acquire the most recent measurements.
  - Further, this is an expandable weather station. We want to release an API so that other developers can write their own weather displays and plug them right in.

Weather Monitoring Station – Class Diagram

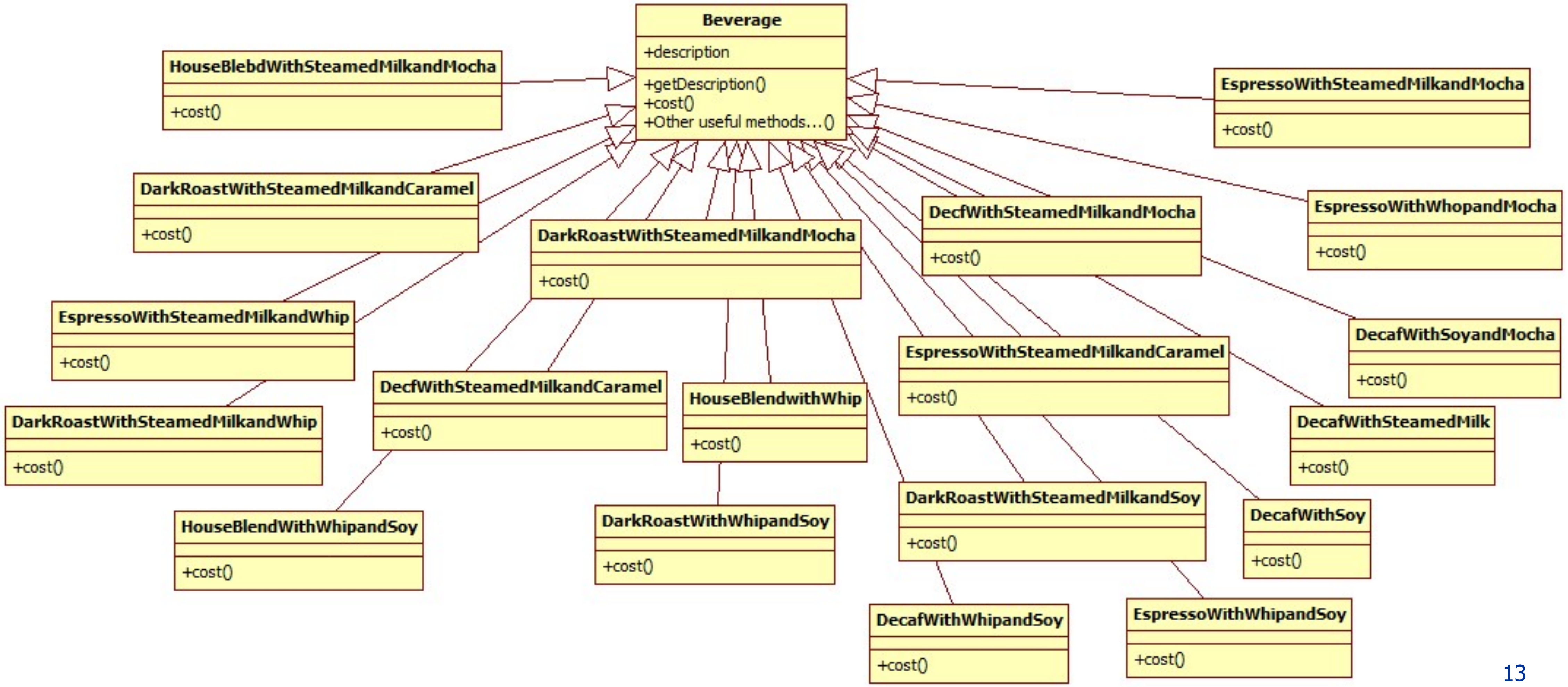


```
public void measurementsChanged(){
    currentConditionsDisplay.update(temp,humidity,pressure);
    statisticsDisplay.update(temp,humidity,pressure);
    forecastDisplay.update(temp,humidity,pressure);
}
```

Software Modeling

- Problem statement
  - NTUCoffee wants to update their ordering systems to match their beverage offerings.
  - There are four kinds of coffees: HouseBlend, DarkRoast, Decaf, and Espresso.
  - In addition to your coffee, you can also ask for several condiments like steamed milk, soy, and mocha, and have it all topped off with whipped milk.
  - NTUCoffee charges a bit for each of these, so they really need to get them built into their order systems.

NTUCoffee Order Systems – Class Diagram


Beverage
+description
+getDescription()
+cost()
+Other useful methods...()
HouseBlebdWithSteamedMilkandMocha
+cost()
EspressoWithSteamedMilkandMocha
+cost()
DarkRoastWithSteamedMilkandCaramel
+cost()
DarkRoastWithSteamedMilkandMocha
+cost()
DecfWithSteamedMilkandMocha
+cost()
EspressoWithWhopandMocha
+cost()
EspressoWithSteamedMilkandWhip
+cost()
DecafWithSoyandMocha
+cost()
DecfWithSteamedMilkandCaramel
+cost()
HouseBlendwithWhip
+cost()
EspressoWithSteamedMilkandCaramel
+cost()
DarkRoastWithSteamedMilkandWhip
+cost()
DecafWithSteamedMilk
+cost()
DarkRoastWithSteamedMilkandSoy
+cost()
DecafWithSoy
+cost()
HouseBlendWithWhipandSoy
+cost()
DarkRoastWithWhipandSoy
+cost()
DecafWithWhipandSoy
+cost()
EspressoWithWhipandSoy
+cost()