

CS 520

Theory and Practice of Software Engineering
Fall 2026

Object Oriented Design Patterns

1

Coming up

- Group selection for final projects due today!
 - I'll reach out to remaining malformed teams
- HW1 is due next Tuesday
- No lecture on Tuesday, September 29 due to CAREER FAIR

2

Today

- Recap: Object oriented design principles
- Design problems & potential solutions
- Design patterns:
 - What is a design pattern?
 - Categories of design patterns
 - Structural design patterns

3

Recap

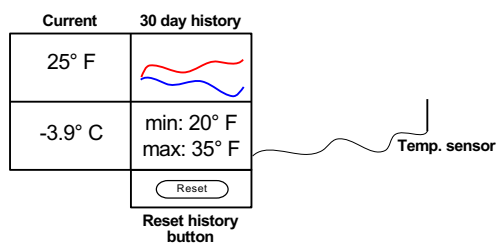
Object oriented design principles

- Information hiding (and encapsulation)
- Open/closed principle
- Liskov substitution principle
- Composition/aggregation over inheritance

4

A first design problem

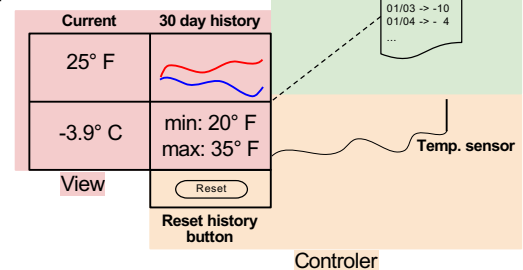
Weather station revisited



5

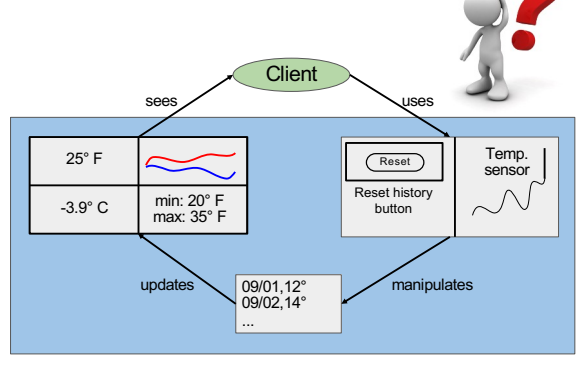
Model View Controller: example

Simple weather station



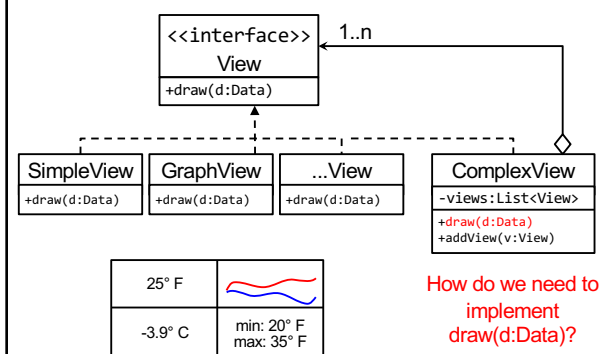
6

What's a good design for the view?



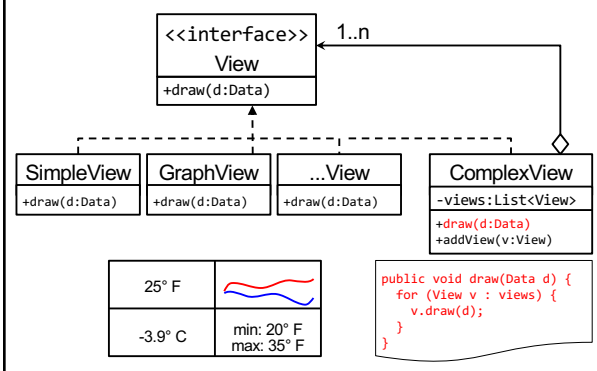
7

Weather station: view



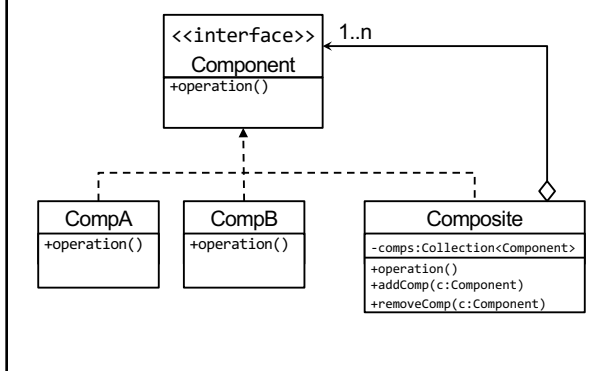
8

Weather station: view



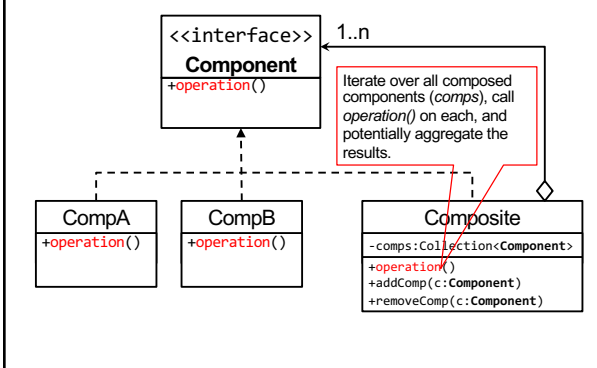
9

Design pattern: Composite



10

Design pattern: Composite



11

What is a design pattern?

- Addresses a recurring, common design problem.
- Provides a generalizable solution.
- Provides a common terminology.

12

What is a design pattern?

- Addresses a recurring, common design problem.
- Provides a generalizable solution.
- Provides a common terminology.

Pros

- Improves communication and documentation.
- "Toolbox" for novice developers.

Cons

- Risk of over-engineering.
- Potential impact on system performance.

More than just a name for common sense and best practices.

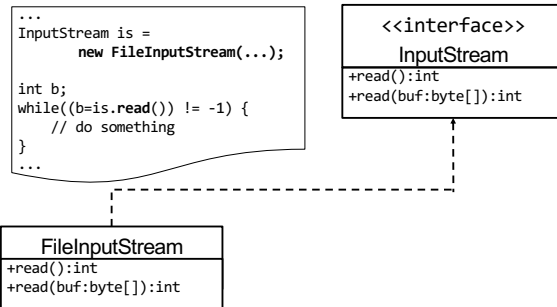
13

Design patterns: categories

1. Structural
 - Composite
 - Decorator
 - ...
2. Behavioral
 - Template method
 - Visitor
 - ...
3. Creational
 - Singleton
 - Factory (method)
 - ...

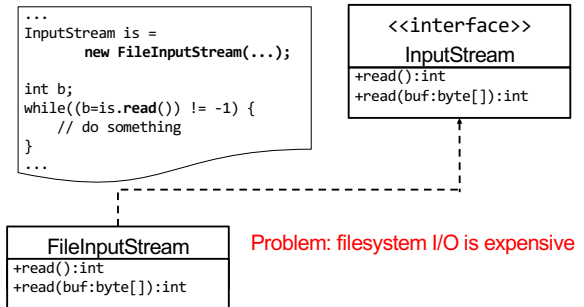
14

Another design problem: I/O streams



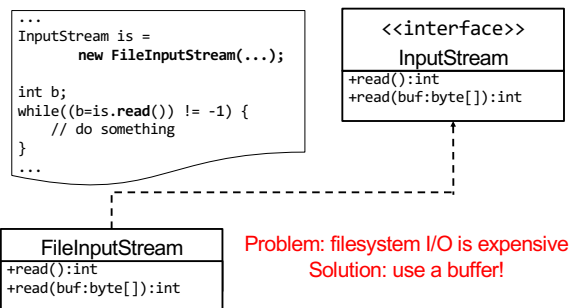
15

Another design problem: I/O streams



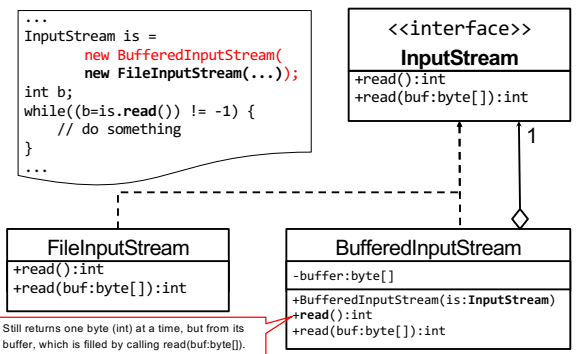
16

Another design problem: I/O streams



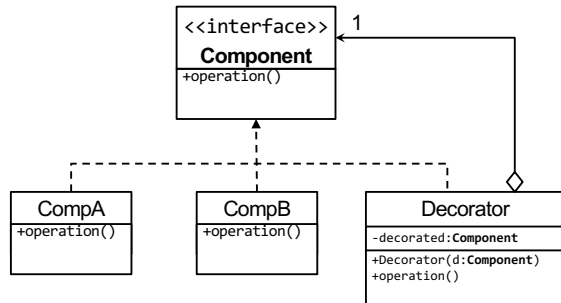
17

Another design problem: I/O streams



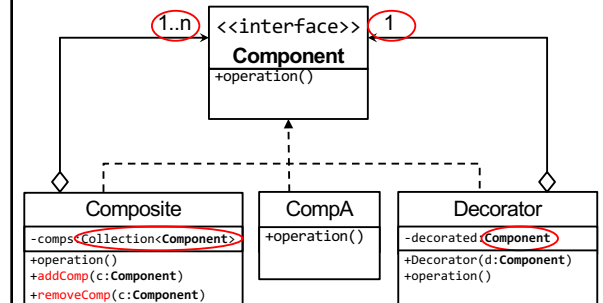
18

Design pattern: Decorator



19

Composite vs. Decorator



20

Find the median in an array of doubles

Examples:

- median([1, 2, 3, 4, 5]) = ???
- median([1, 2, 3, 4]) = ???



21

Find the median in an array of doubles

Examples:

- median([1, 2, 3, 4, 5]) = 3
- median([1, 2, 3, 4]) = 2.5

**Algorithm****Input:** array of length n **Output:** median

22

Find the median in an array of doubles

Examples:

- median([1, 2, 3, 4, 5]) = 3
- median([1, 2, 3, 4]) = 2.5

Algorithm**Input:** array of length n **Output:** median

1. Sort array
2. if n is odd return $((n+1)/2)$ th element
otherwise return arithmetic mean of $(n/2)$ th element and $((n/2)+1)$ th element

23

Median computation: naive solution

```

public static void main(String ... args) {
    System.out.println(median(1,2,3,4,5));
}

public static double median(double ... numbers) {
    int n = numbers.length;
    boolean swapped = true;
    while(swapped) {
        swapped = false;
        for (int i = 1; i < n; ++i) {
            if (numbers[i-1] > numbers[i]) {
                // swap
                swapped = true;
            }
        }
    }
    if (n % 2 == 0) {
        return (numbers[(n/2) - 1] + numbers[n/2]) / 2;
    } else {
        return numbers[n/2];
    }
}

```

What's wrong with this design?
How can we improve it?



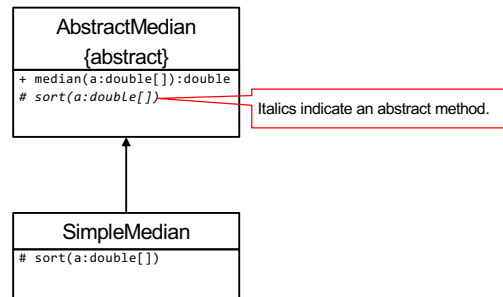
24

Ways to improve

- 1: Monolithic version, static context.
- 2: Extracted sorting method, non-static context.
- 3: Proper package structure and visibility, extracted main method.
- 4: Proper testing infrastructure and build system.

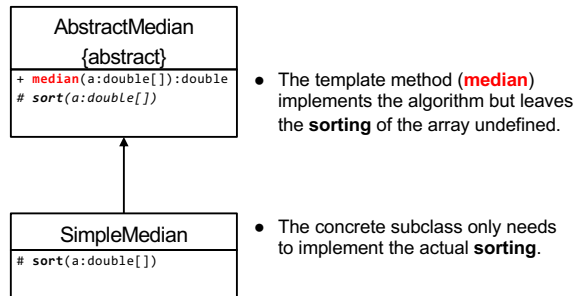
25

One possible solution: **template method pattern**



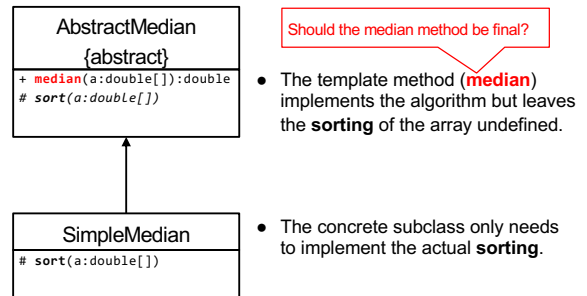
26

One possible solution: template method pattern



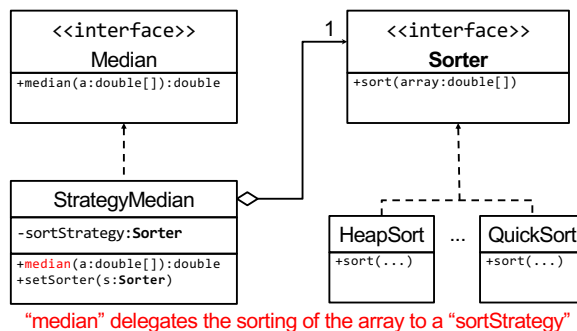
27

One possible solution: template method pattern



28

Another solution: **strategy pattern**



29

Template method pattern vs. strategy pattern

Two solutions to the same problem



What are the differences, pros, and cons?

30

Template method pattern vs. strategy pattern

Two solutions to the same problem

Template method

- Behavior selected at compile time.
- Template method is usually final.

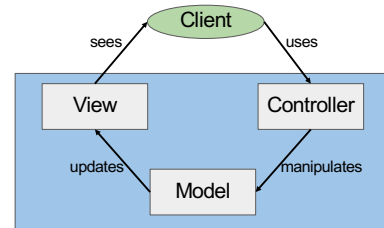
Strategy

- Behavior selected at runtime.
- Composition/aggregation over inheritance.

31

Model-View-Controller revisited

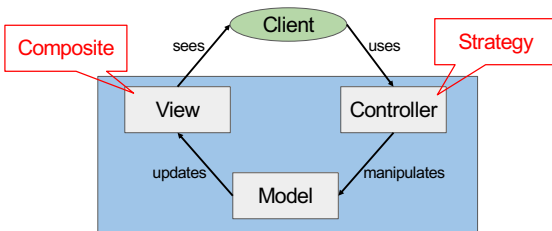
Design patterns in a MVC architecture



32

Model-View-Controller revisited

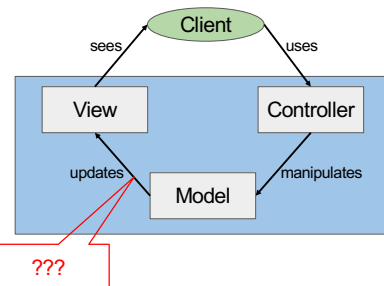
Design patterns in a MVC architecture



33

Model-View-Controller revisited

Design patterns in a MVC architecture



34

Observer pattern

Observer pattern

From Wikipedia, the free encyclopedia

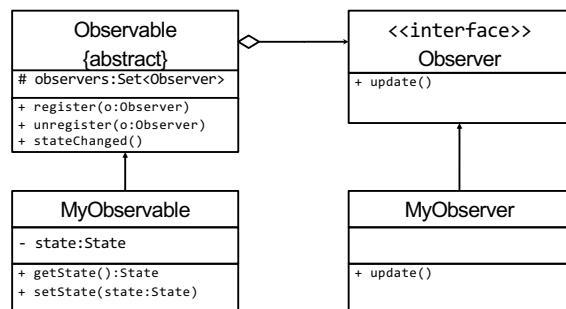
The **observer pattern** is a *software design pattern* in which an object, called the **subject**, maintains a list of its dependents, called **observers**, and notifies them automatically of any state changes, usually by calling one of their *methods*.

• Problem solved:

- A one-to-many dependency between objects should be defined without making the objects tightly coupled.
- When one object changes state, an open-ended number of dependent objects are updated automatically.
- One object can notify an open-ended number of other objects.

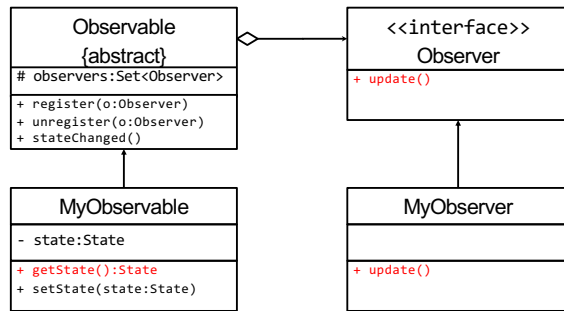
35

Observer pattern



36

Observer pattern

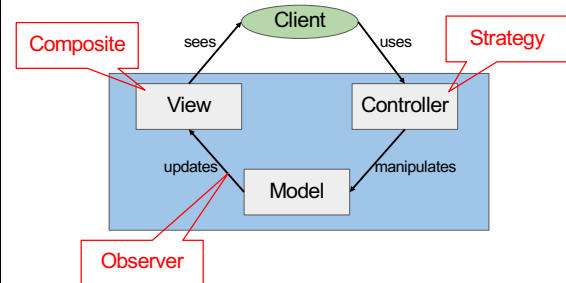


Variation: pass incremental changes or the state to update method.

37

Model-View-Controller revisited

Design patterns in a MVC architecture



38

Reminder: Coming up

- Group selection for final projects due today!
 - I'll reach out to remaining malformed teams
- HW1 is due next Tuesday
- No lecture on Tuesday, September 29 due to CAREER FAIR

39