CS 520/620
Advanced Software Engineering
Spring 2016

February 09, 2016
The big picture

Software design and architecture

Software engineering research and paper presentations

Class project

Midterm
The big picture

Software design and architecture

- Modeling and abstraction
- Software architecture and design
- UML (class diagrams)
- OO principles
- Design patterns
The big picture

Software design and architecture

- Modeling and abstraction
- Software architecture and design
- UML (class diagrams)
- OO principles
- Design patterns

We will wrap up this section this week.
The big picture

Software design and architecture (until 02/11)

Software engineering research and paper presentations

- Introduction and discussion of paper suggestions 02/18
- Paper selection 02/25
- Paper presentations and tooling sessions 03/01--03/31
The big picture

Software design and architecture (until 02/11)

Software engineering research and paper presentations (until 03/31)

Class project
- Project plan 02/23
- Project presentations end of April
The big picture

Software design and architecture (until 02/11)

Software engineering research and paper presentations (until 03/31)

Class project (until 04/26)

Midterm

- Preparation 04/05
- Exam (in class) 04/07
The big picture

Software design and architecture (until 02/11)

Software engineering research and paper presentations (until 03/31)

Class project (until 04/26)

Midterm (04/07)
Logistics

Two assignments

- MVC implementation
  - Individual submissions due 02/18/2016, 11:55pm

- Project plan
  - Group submissions due 02/23/2016, 11:55pm
Project plan

The project plan should answer the following questions:

- What problem are you addressing?
- Why is this problem interesting and non-trivial?
- How will you solve the problem? (timeline and tasks for each group member)
- How will you evaluate your solution? (how will you show that you solved the problem)
Recap

- Decorator pattern revisited
- Behavioral patterns
  - Template method
  - Strategy
Template method pattern

AbstractStats

+ median(a:double[]):double
+ sort(a:double[])

MyStats

+ sort(a:double[])

The template method (median) implements the algorithm but leaves the sorting of the array open.

The concrete subclass only needs to implement the actual sorting.
Strategy pattern

```plaintext
<<interface>>
Stats
+ median(a:double[]):double

MyStats
- sortStrategy:Sorter
+ median(a:double[]):double

<<interface>>
Sorter
+ sort(array:double[])

HeapSort

QuickSort
...
```

MyStats delegates the sorting of the array to a “strategy”, which can be configured and changed at runtime.
Today

● A thorough code review
  ○ How to recognize a bad design?
  ○ How to come up with a better one?

● Two more behavioral patterns
  ○ Iterator
  ○ Observer

● A concrete example for the MVC paradigm
Simple stats: a first attempt

Numbers: 6 Median: 2.0 Mean: 3.0

1,2,2,2,1,10,
Simple stats: a first attempt

Problems:

- Code duplication
- No information hiding and encapsulation
- No decoupling (separation of concerns)
- Violates the open/closed principle
- Poor to no testability
Simple stats: use the MVC paradigm?

Client

View

controller

model

Client sees

uses

updates

manipulates
Simple stats: use the MVC paradigm?

The MVC paradigm is not a single design pattern. It incorporates several design patterns!
Observer and iterator

Observer pattern
● Models a “one to many” dependency.
● Decoupling of state and action:
  Notify registered observer(s) about state change.

Iterator pattern
● Provides (sequential) access to a data structure.
● Does not reveal implementation details.