CS220
Software Maintenance
Definition

• Software maintenance
  • The process of changing a software system after it has been delivered
Reasons behind maintenance

- Why delivered software needs change?
Reasons behind maintenance

• Why delivered software needs change?
  • Fault repair
  • Platform adaptation
  • System enhancement (adding functionality features)
Maintenance is important

• 60%--80% of overall IT cost
  • Software is too expensive to discard after one version
A big picture

• Initial development
• Software evolution
• Software servicing
• Phase-out phase
A smaller picture --- evolution process
Software reengineering

• Redocumenting
• Structure/architecture refactoring
• Programming language translation
• Data reengineering
When to stop supporting a software
When to stop supporting a software

• Business value
• Maintenance expense
Design patterns
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch
What are design patterns

• Solutions to specific problems in OO software design
• 23 patterns in 3 categories
  • Creational
  • Structural
    • Composite
    • ...
  • Behavioral
    • Observer
    • Interpreter
    • ...
Why are we studying them?
Observer

• One to many relationship
  • The many need to know changes in “one” immediately

• Example
  • Points & Shapes
  • Map & location-based services
  • A game character & other game components
  • ...
Example

• If a person/subject changes its status, how to let all his “subscriber” knows?
  • What to do when there is only one subscriber?
  • What to do when there are multiple subscribers of different types?
  • What if new subscribers are added?
  • How to make the code easy to maintain and extend?
Class diagram

```
for each view in views v.update()
```

```
model.getState();
```

```
Subject
+attach(in Observer)
+setState()
+getState()
```

```
Observer
+update()
```

```
ViewOne
+update()
```

```
ViewTwo
+update()
```
Example (location, location-related service)

• “location” would be the Subject in previous slide
• “observer” would be the superclass of all the sub-classes that try to update themselves based on the location information
The benefit of observer pattern

• When new types of observers are added, the prototype and implementation of the subject class doesn’t need any changes.