Design Patterns

...live and in action!

CSE 331, 10/28/11

TA: Krysta Yousoufian
Housekeeping

- Homework 3: Sunday
- Midterm: Friday
- Questions?
Common Patterns

• Recall from lecture...

• Creational
  o Create objects without calling constructor directly
  o Singleton: allow only one instance
  o Factory: hide constructors
  o Prototype: “cloneable” objects

• Structural (wrappers)
  o Interact with the “important” class through a wrapper class
  o Adapter: different interface, same functionality
  o Decorator: same interface, different functionality
  o Proxy: same interface, same functionality

• Behavioral
  o Interface for communication between objects
  o Visitor: traverse sa data structure
Singleton

• One shared instance of a class

• When useful
  o Maintaining global state; coordinating among applications
  o Often lower-level tasks (e.g. hardware interaction)

• When not useful
  o Need to store state/data specific to each use (instance fields)

• Controversial
  o Global \(\rightarrow\) hides dependencies, hard to test
  o Overused
  o Good tool to have, but only use if it’s the right tool (get a second opinion!)

• Examples: logger, window manager
Implementing Singleton

• Private constructor

• Several options (Effective Java pp. 18+)
  o One publicly accessible static instance
    • Pros: clarity – obvious that you’re using a shared copy
  o One private static instance, accessed with getInstance()
    • Pros: flexibility – could reimplement getInstance() to no longer be Singleton
      • Nice style – use for this class unless we tell you otherwise
  o Enum
    • Pros: safer (harder to break Singleton), provides serialization
    • But not how Enum is meant to be used
    • Josh Bloch recommends this, but avoid for now unless we tell you otherwise
Singleton Demo

FileSync / Logger

- Logger.java
- Client.java
- FileServer.java
- IOUtil.java
Factory

- Get new object by calling non-constructor (getInstance(), valueOf(), …)
  - May create a new object or may reuse an old one
- Advantages (*Effective Java, pg. 5*)
  - Can reuse objects
  - Can return objects of subtypes
  - More descriptive naming than constructors
- Examples
  - Boolean.valueOf() – reuse objects
  - Collections interface: static methods return private subclass
Factory Demo

GameFactory / GameRoom

- GameFactory.java
- GameRoom.java
- Game.java
Adapter

- Different interface, same functionality
- Use: translate interface to be compatible with a different object

Demo: TicTacToe / GameRoom

- TicTacToe.java
- SimpleTicTacToe.java
- Game.java
- GameRoom.java
Proxy

• Same interface – just adds a wrapper

• Uses:
  o Support concurrency – e.g. add locks to restrict access to data structures
  o Security – e.g. verify credentials
  o ...

Visitor

- Traverse a hierarchical data structure (e.g. tree)
- Do something at each step
- Nodes of data structure implement `accept(Visitor v)`
  - Calls `visit(this)` and `accept(v)` on each child
- Visitor implements `visit(Node n)`
  - Does some computation, printing, etc.
- Uses
  - “Pretty printers” for trees (e.g. compilers)
Visitor Demo

- PurchaseVisitor.java
- PurchaseNode.java
- VisitorTest.java