Problem: Don't want to load in all the product info just to show a list to choose from
Proxy (General)

subject
+operation()

actualsubject
+operation()

proxysubject
+subject: ActualSubject
+operation()

return subject.operation();
Proxy Types

- **Virtual**  
  Save space by not putting everything in memory

- **Remote**  
  Reduce the "load" on an object by load balancing between proxies

- **Protection**  
  Keep sensitive data out of memory
Problem: the proxy caches information which might change – how do we keep the proxy cache up to date?
- The proxy keeps checking with the subject to see whether anything has changed (i.e. the proxy *polls* the subject)
Observer II

```
for (cache c : cachelist)
    c.update_data();
```

```
cachelist.add(c);
```
Observer (General)

**Subject**

- #state
- #observers
- +attach(Observer)
- +detach(Observer)
- +get州State()
- +notify()

**Observer**

- #state
- #subject
- +update()

```java
for (Observer o : observers)
    o.update();
state = subject.getState();
observers.add(observer);
```
Problem: Different versions of the web interface for different browsers - how do we adapt the code to support this?
public class WebButton {
    private int mInterfaceType;

    public void render() {
        if (mInterfaceType == 0) {
            // Output HTML button
        } else if (mInterfaceType == 1) {
            // Output Flash button
        } else ...
    }
}
Abstract Factory II

```
 InterfaceElement
     +render()

 HTMLElement

 HTMLTextBox  HTMLButton  ...

 FlashElement

 FlashTextBox  FlashButton  ...
```
Abstract Factory III

InterfaceElement

+render()
Factory (General)  WRONG
Abstract Factory (General)
Design Patterns...

- Are guidelines, not rules!
- You should adapt them to your problem
- Many are used in modern libraries
- You should implement a few and be able to discuss them
- What about *performance*..?