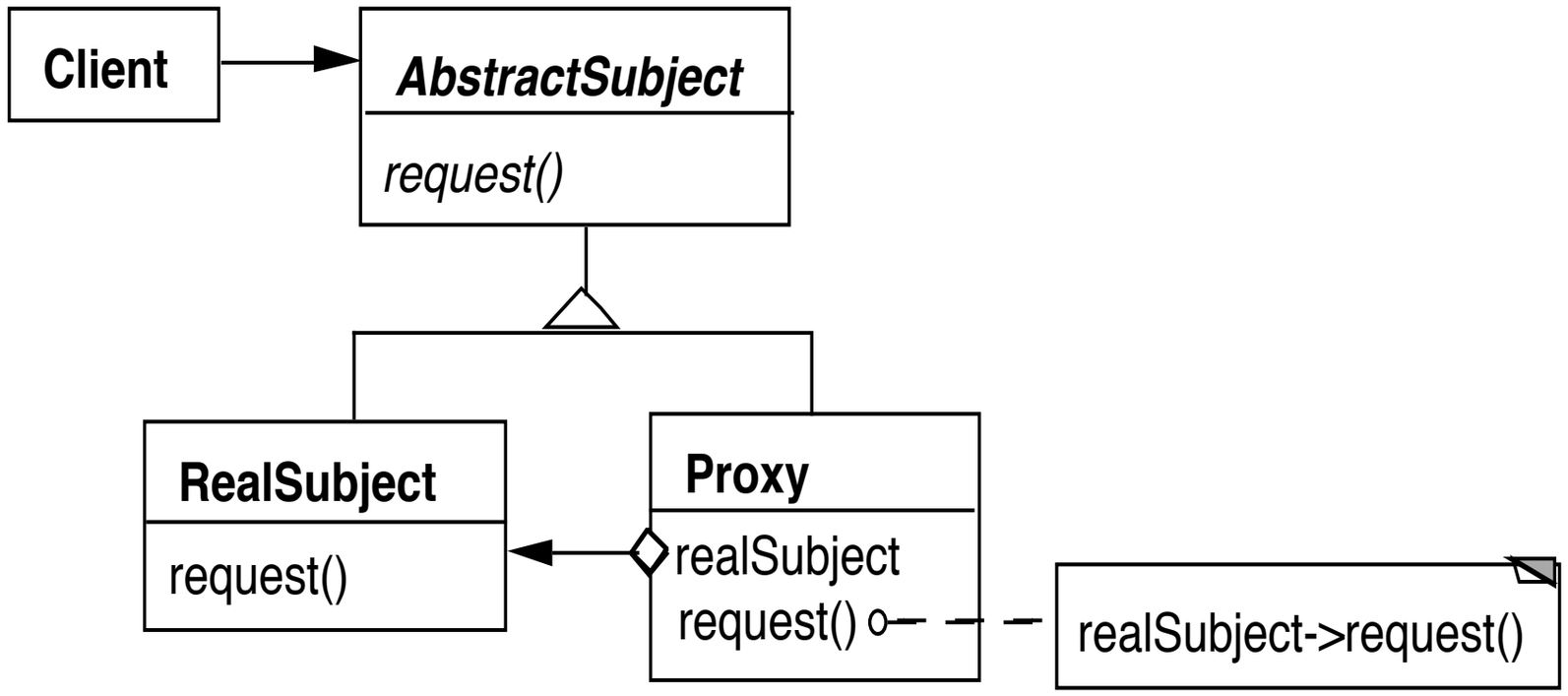
CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2013
Doc 18 Proxy
April 23, 2013

Proxy (Surrogate)

a person authorized to act on behalf of another

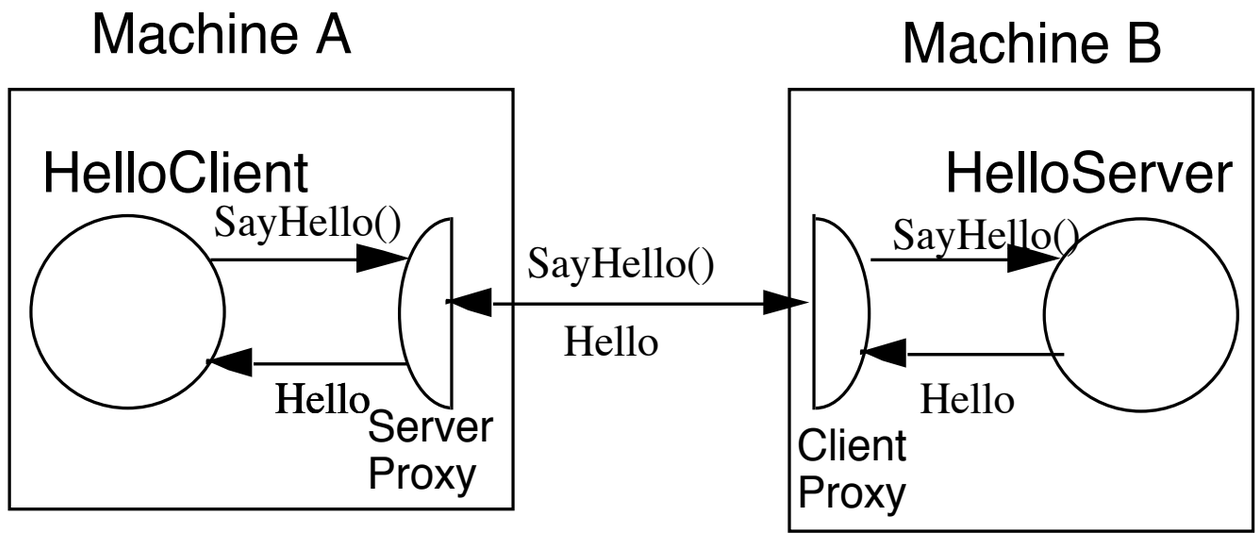



```
class Proxy {
    AbstractSubject realSubject;

public Foo service(Bar x ) {
     return realSubject(x);
}
```

Why do it?

Remote Proxy




```
String server = getHelloHostAddress( args);
Hello proxy = (Hello) Naming.lookup( server );
String message = proxy.sayHello();
System.out.println( message );
```

More General Proxy

```
class Proxy {
        AbstractSubject realSubject;

        public Foo service(Bar x ) {
                some preprocessing
                result = realSubject(x);
                some postprocessing
        }
```

Virtual Proxy

Creates/accesses expensive objects on demand

O-R Mapping Layers

Java's Synchronized List

```
ArrayList notSafe = new ArrayList();
List threadSafe = Collections.synchronizedList(notSafe);
```

```
static class SynchronizedList {
        List list;
        public Object get(int index) {
            synchronized(mutex) {return list.get(index);}
        }
}
```

Java's Unmodifiable List

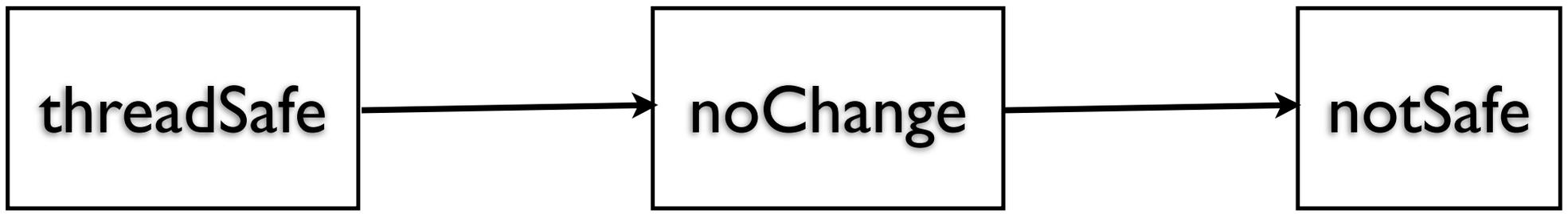
```
ArrayList notSafe = new ArrayList();
List noChange = Collections.unmodifiableList(notSafe);
```

```
static class UnmodifiableList {
        List list;
        public Object get(int index) { return list.get(index);}

        public Object set(int index, Object element) {
            throw new UnsupportedOperationException();
         }
```

Proxy or Decorator?

```
ArrayList notSafe = new ArrayList();
List noChange = Collections.unmodifiableList(notSafe);
List threadSafe = Collections.synchronizedList(noChange);
```



Proxy verses Decorator

"Decorators can have similar implementations as proxies"

Proxy controls access to an object

Decorator adds one or more responsibilities to an object

Smalltalk Proxy Trick

```
Object subclass: #Proxy
        instanceVariableNames: 'target '
        classVariableNames: ''
        poolDictionaries: ''
        category: 'Whitney-Examples'
```

Class Method

```
on: anObject
        ^super new target: anObject
```

Instance Methods

```
doesNotUnderstand: aMessage
        ^target
                perform: aMessage selector
                withArguments: aMessage arguments

target: anObject
        target := anObject
```

```
| wrapper |
wrapper := Proxy on: Transcript.
wrapper open.
wrapper show: 'Hi mom'.
```

```
| wrapper |
wrapper := Proxy on: 3.
wrapper + 5.
```

```
| wrapper |
wrapper := Proxy on: 'Hi '.
wrapper , ' mom'.
```

Java Proxy Trick

java.lang.reflect.Proxy

Can build a proxy object to interfaces

Interface and Implementation

```
public interface BarInterface {
	public String hello();

	public int foo();
}

public class Bar implements BarInterface {

	public String hello() {
		return "Hello world";
	}

	public int foo() {
		return 1;
	}
}
```

Invocation Handler

```
public class BarInvocationHandler implements InvocationHandler {

        private Bar realSubject;
        private int methodCallCount = 0;

        public BarInvocationHandler(Bar target) {
                this.realSubject = target;
        }
```

Invocation Handler

```
public Object invoke(Object proxy, Method method, Object[] args)
        throws Throwable {
    methodCallCount++;
    if (Object.class == method.getDeclaringClass()) {
        String name = method.getName();
        if ("equals".equals(name)) {
            return proxy == args[0];
        } else if ("hashCode".equals(name)) {
            return System.identityHashCode(proxy);
        } else if ("toString".equals(name)) {
            return realSubject.toString();
        } else {
            throw new IllegalStateException(String.valueOf(method));
        }
    }
    if (method.getName() == "hello")
        return method.invoke(realSubject, args) + " from proxy";
    return method.invoke(realSubject, args);
}
```

Using Proxy

```
public static void main(String[] args) {
        Bar realSubject = new Bar();
        BarInvocationHandler handler = new BarInvocationHandler(realSubject);
        BarInterface proxy = (BarInterface) Proxy.newProxyInstance(
                        Bar.class.getClassLoader(), new Class[] { BarInterface.class },
                        handler);
        System.out.println(proxy.hello());                          //Hello world from proxy
        System.out.println("Foo method: " + proxy.foo());           //Foo method: 1

        if (proxy instanceof BarInterface)
                System.out.println("is instance of BarInterface");  //is instance of BarInterface
        if (proxy instanceof Bar)
                System.out.println("is instance of Bar");
        System.out.println("Proxy class: " + proxy.getClass());     //Proxy class: class $Proxy0
        System.out.println(proxy.toString());                       // Bar@10f0f6ac
        System.out.println("Count " + handler.methodCallCount());   //Count 3

}
```