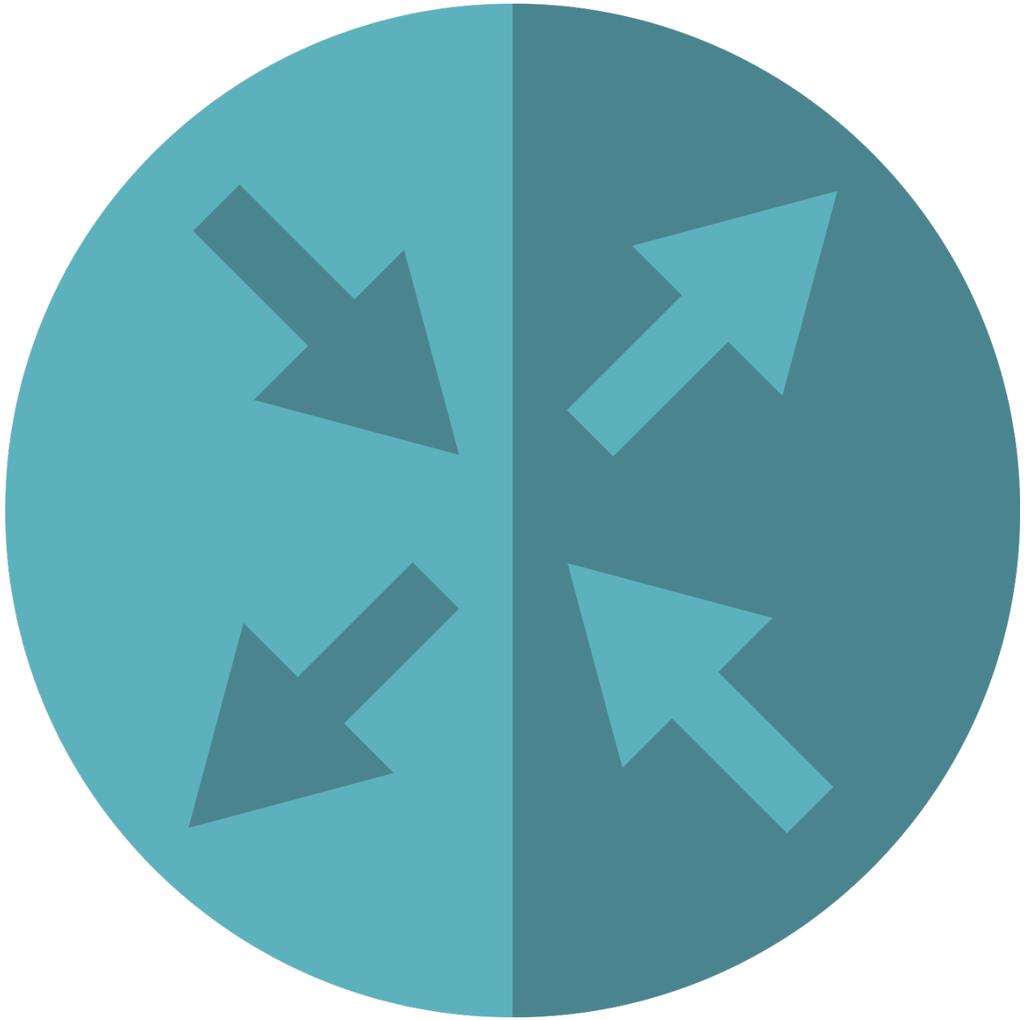
Proxy Pattern

Bryan Hansen

twitter: bh5k | http://www.linkedin.com/in/hansenbryan

Concepts

- Interface by wrapping
- Can add functionality
- Security, Simplicity, Remote, Cost
- Proxy called to access real object
- Examples:
  - java.lang.reflect.Proxy
  - java.rmi.*



Design

Interface based

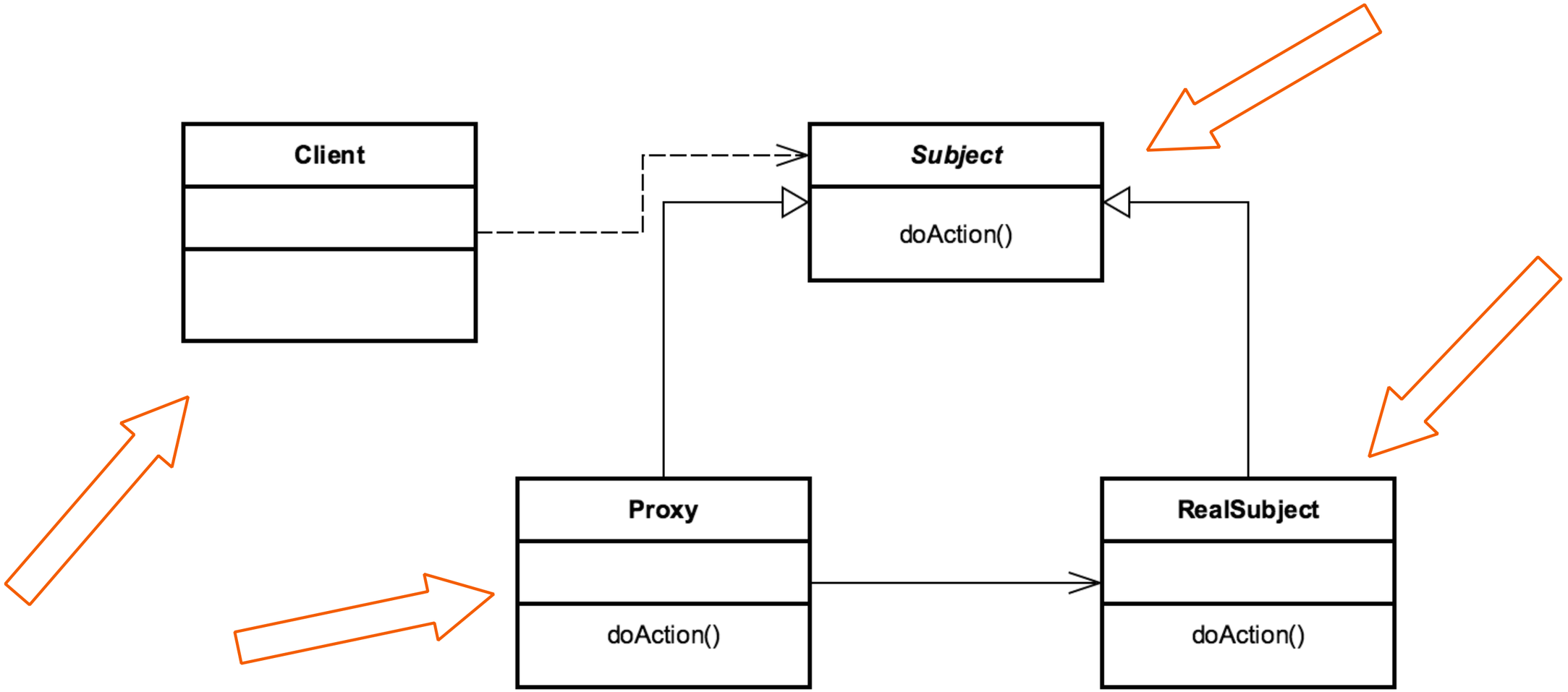
Interface and Implementation Class

java.lang.reflect.InvocationHandler

java.lang.reflect.Proxy

Client, Interface, InvocationHandler, Proxy, Implementation

UML


Client
Subject
doAction()
Proxy
doAction()
RealSubject
doAction()

Everyday Example - Proxy

```
public Object invoke(Object proxy, Method m, Object[] args) throws
  Throwable {
     Object result;
     try {
         result = m.invoke(obj, args);
     } catch (InvocationTargetException e) {
        throw e.getTargetException();
     } catch (Exception e) {
        throw new RuntimeException("unexpected invocation exception: "
             + e.getMessage());
     }
     return result;
}
```

Exercise Proxy

Twitter API Download

Twitter Service Impl

Restrict POST call

Pitfalls

- Only one proxy
- Another Abstraction
- Similar to other patterns

Contrast

Proxy

- Can add functionality, but not its main purpose
- Can only have one
- Compile time

Decorator

- Dynamically add functionality
- Chained
- Decorator points to its own type
- Runtime

Proxy Summary

- Great utilities built into Java API
- Only one instance
- Used by DIJ/IOC Frameworks