Proxy
Design Pattern

By Joshua Neutrom (#39), Yunlong Liu (#25), Chen Wang (#58), Dayu Wang (#59)

Introduction

Definition

Relationship

Benefits

Negatives

UML

Illustration

Code Example

Input/Output of Code

Questions

Definition

Structural design pattern

Surrogate/pointer to real object

First Request - Create object

Future requests - Directed to real object

Definition

Types of Proxies

1. Virtual - Resource intensive objects
2. Remote - Object at different location
3. Protective - Permission to access
4. Smart - Features such as counts, locks, and loading

Comparisons

Difference in interface of Proxy (same as object) from:

Adapter - different from object

Decorator - enhanced from object

Similar structure to Decorator

Benefits

Enhanced efficiency and lower cost

Decoupling clients from the location of remote server components

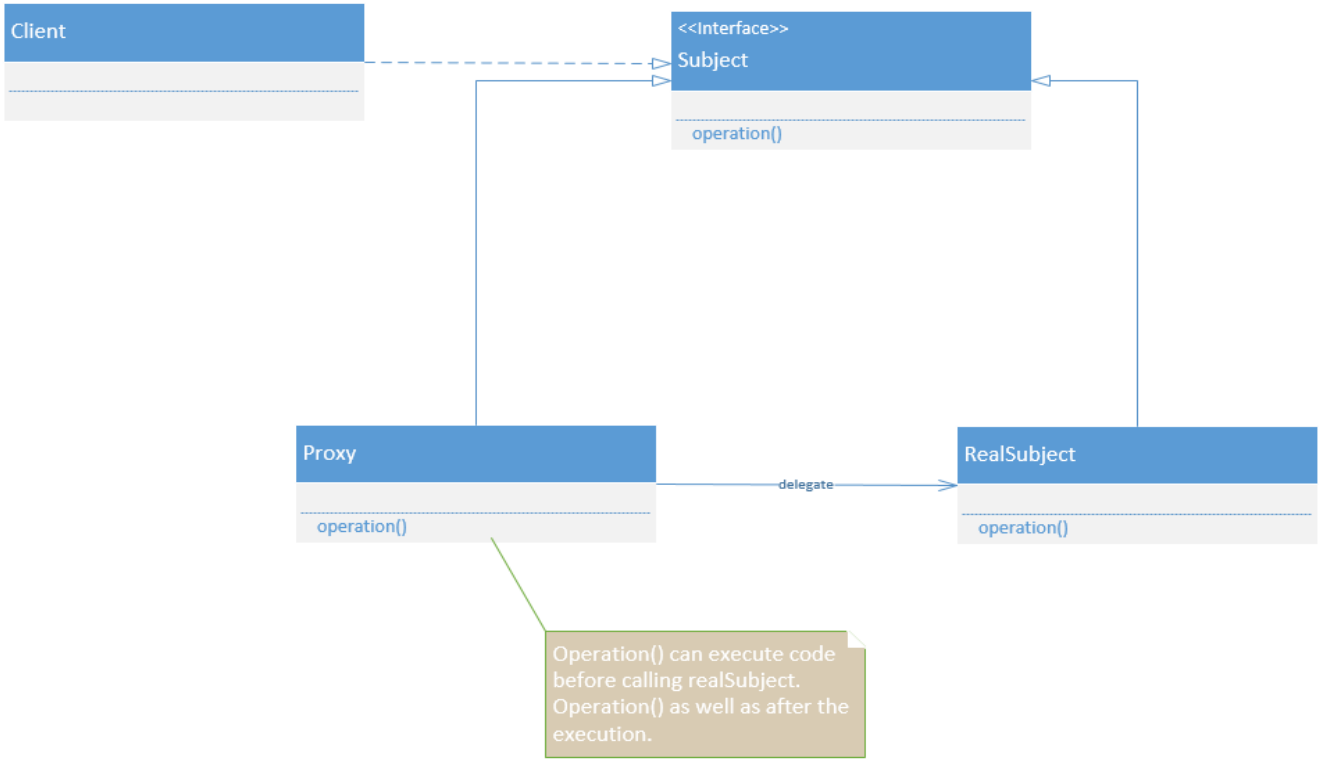
Separation of housekeeping code from functionality

Negatives

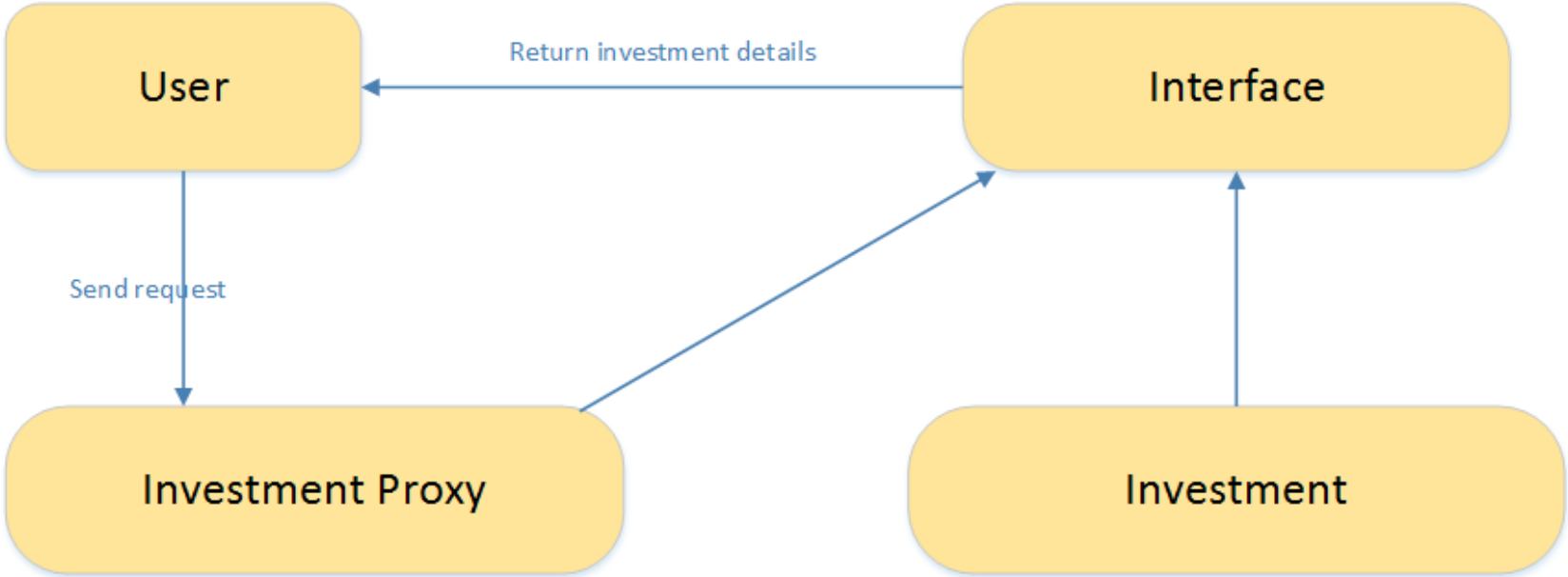
Less efficiency due to indirection

Complex implementation

UML


Client
<<Interface>>
Subject
operation()
Proxy
operation()
delegate
RealSubject
operation()
Operation() can execute code before calling realSubject. Operation() as well as after the execution.

Illustration



Sample Code

*InvestmentViewer.java | InvestmentProxy.java | User.java | GetInvestmentDetails.java | *Investment.java

```java
package team1.cs5551.fs16.proxy_design.study;

import org.joda.time.DateTime;

public class InvestmentProxy implements GetInvestmentDetails{

    private Investment realInvestment;

    public InvestmentProxy(Investment realInv) {
        realInvestment = realInv;
    }

    @Override
    public String GetInvestor() {
        return realInvestment.GetInvestor();
    }

    @Override
    public String GetStock() {
        return realInvestment.GetStock();
    }

    @Override
    public DateTime GetPurchaseDate() {
        return realInvestment.GetPurchaseDate();
    }

    @Override
    public double GetPurchasePrice() {
        return realInvestment.GetPurchasePrice();
    }
}
```

Sample Code

*InvestmentViewer.java | InvestmentProxy.java | User.java | GetInvestmentDetails.java | *Investment.java

```java
package team1.cs5551.fs16.proxy_design.study;

import org.joda.time.DateTime;

public class Investment implements GetInvestmentDetails {
    private User investor;
    private DateTime purchaseDate;
    private String stockSymbol; // All uppercase letters
    private double purchaseUnitPrice; // Price per share (in USD)

    // Proxy does not allow others to see the total investment amount by hiding the stockShares.
    private int stockShares;

    public Investment(User theUser, DateTime date, String symbol, double unitPrice) {
        investor = theUser;
        purchaseDate = date;
        stockSymbol = symbol;
        purchaseUnitPrice = unitPrice;
    }

    public Investment(Investment other) {
        this.investor = other.investor;
        this.purchaseDate = other.purchaseDate;
        this.stockSymbol = other.stockSymbol;
        this.purchaseUnitPrice = other.purchaseUnitPrice;
        this.stockShares = other.stockShares;
    }

    public User GetFullUser() { return investor; }
    public int getShares() { return stockShares; }

    @Override
    public String GetInvestor() {
        return investor.getUserName();
    }

    @Override
    public String GetStock() {
        return stockSymbol;
    }

    @Override
    public DateTime GetPurchaseDate() {
        return purchaseDate;
    }
```

Sample Code

*InvestmentViewer.java | InvestmentProxy.java | User.java | GetInvestmentDetails.java

```java
package team1.cs5551.fs16.proxy_design.study;

import org.joda.time.DateTime;

public interface GetInvestmentDetails {
    public String GetInvestor();
    public String GetStock();
    public DateTime GetPurchaseDate();
    public double GetPurchasePrice();
}
```

Sample Code

*InvestmentViewer.java | InvestmentProxy.java | User.java | GetInvestmentDetails.java | Investment.java

```java
package team1.cs5551.fs16.proxy_design.study;

public class User {
    private String userName;

    // The password and email are not allowed to be viewed by others by the proxy.
    private String password;
    private String emailAddress;

    public User(String id, String pwd, String email) {
        userName = id;
        password = pwd;
        emailAddress = email;
    }

    public String getUserName() { return userName; }
    public String getPassword() { return password; }
    public void changePassword(String newPwd) { password = newPwd; }
    public String getEmail() { return emailAddress; }
    public void changeEmail(String newEmail) { emailAddress = newEmail; }
}
```

Input

```java
package team1.cs5551.fs16.proxy_design.study;

import org.joda.time.DateTime;

public class InvestmentViewer {
    public static void main(String[] args) {
        // Input
        User a_user = new User("Team 1", "CS5551", "an_email@example.com");
        Investment an_investment = new Investment(a_user, DateTime.parse("2012-04-15"), "GOOG", 22.66);

        GetInvestmentDetails realInvestment = new Investment(an_investment);
        GetInvestmentDetails proxy = new InvestmentProxy(an_investment);

        // Output
        System.out.println("Investment Details:");
        System.out.println("User: " + proxy.GetInvestor());
        System.out.println("Stock: " + proxy.GetStock());
        System.out.println("Purchase Date: " + DateTimeFormat.forPattern("MM/dd/yy").print(proxy.GetPurchaseDate()));
        System.out.println("Purchased Price per Share: $" + proxy.GetPurchasePrice());
    }
}
```

Output

Markers Properties Servers Data Source Explorer Snippets Console

```
<terminated> InvestmentViewer [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\
Investment Details:
User: Team 1
Stock: GOOG
Purchase Date: 04/15/12
Purchased Price per Share: $22.66
```

Conclusion

Definition

Benefits

Negatives

Illustration

UML

Code Example

Input/Output of Code

Questions

Questions/Compliments