Assignment 3 : structural-patterns

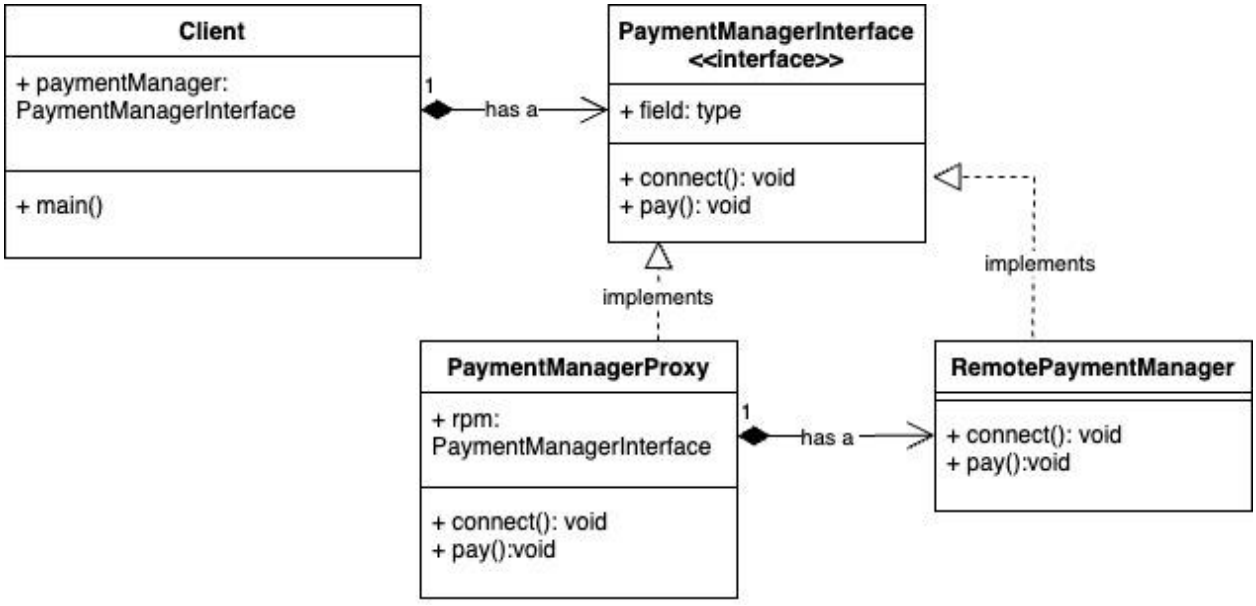
Sree Guru charan Dandyala
029394467 (sreegurucharan.dandyala01@student.csulb.edu)

Find a compelling scenario where you can apply the **Remote Proxy Design Pattern** to the food delivery system. Draw the corresponding **class diagram** and **sequence diagram**

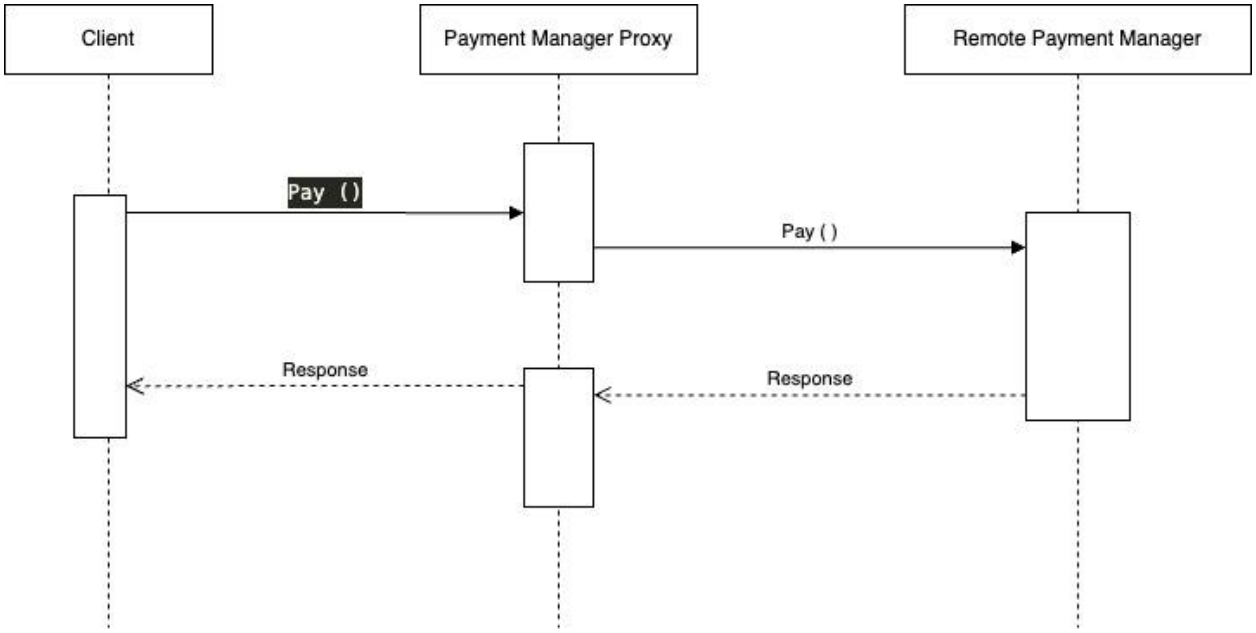
One compelling Scenario for using a Remote Proxy Design Pattern in the food delivery system is by breaking down the entire application into microservices and when an order is placed the order service should make a request to the Payment Service to process the payment microservice.

The Order Service will have a PaymentManager proxy object which is responsible for connecting to the PaymentManager Service and processing the payment. Both the payment Manager and the proxy implement a common interface and the proxy object abstracts the entire logic of connecting to the remote service, serializing and deserializing the payload and response from the client.

Class Diagram:



Sequence Diagram:



Java Implementation:

```
package DesignPatterns.remoteProxy;

interface PaymentManagerInterface {
   public void connect();

   public void pay(String sender, String reciever, double amount);
}
```

```
package DesignPatterns.remoteProxy;

public class PaymentManagerProxy implements PaymentManagerInterface {

   PaymentManagerInterface rpm;

   PaymentManagerProxy() {
```

```
        this.rpm = new RemotePaymentManager();
        this.connect();
    }

    @Override
    public void connect() {
        this.rpm.connect();
    }

    @Override
    public void pay(String sender, String reciever, double amount) {
        this.rpm.pay(sender, reciever, amount);
    }

}
```

```
package DesignPatterns.remoteProxy;

public class RemotePaymentManager implements PaymentManagerInterface
{

    @Override
    public void connect() {
        System.out.println("connecting to remote payment manager
service");
        System.out.println("_____________________");
    }

    @Override
    public void pay(String sender, String reciever, double amount) {
        System.out.println("processing payment from " + sender + " to
" + reciever + "$" + amount);
        System.out.println("_____________________");
    }
```

```
}
```

```
package DesignPatterns.remoteProxy;

public class Client {
   public static void main(String[] args) {
       PaymentManagerInterface pm = new PaymentManagerProxy();
       pm.pay("guru", "charan", 200);
       pm.pay("vijay", "kiran", 300);
   }
}
```