Software Systems Architecture (CS-586)

Homework #2

- Sivasenthil Namachivayan

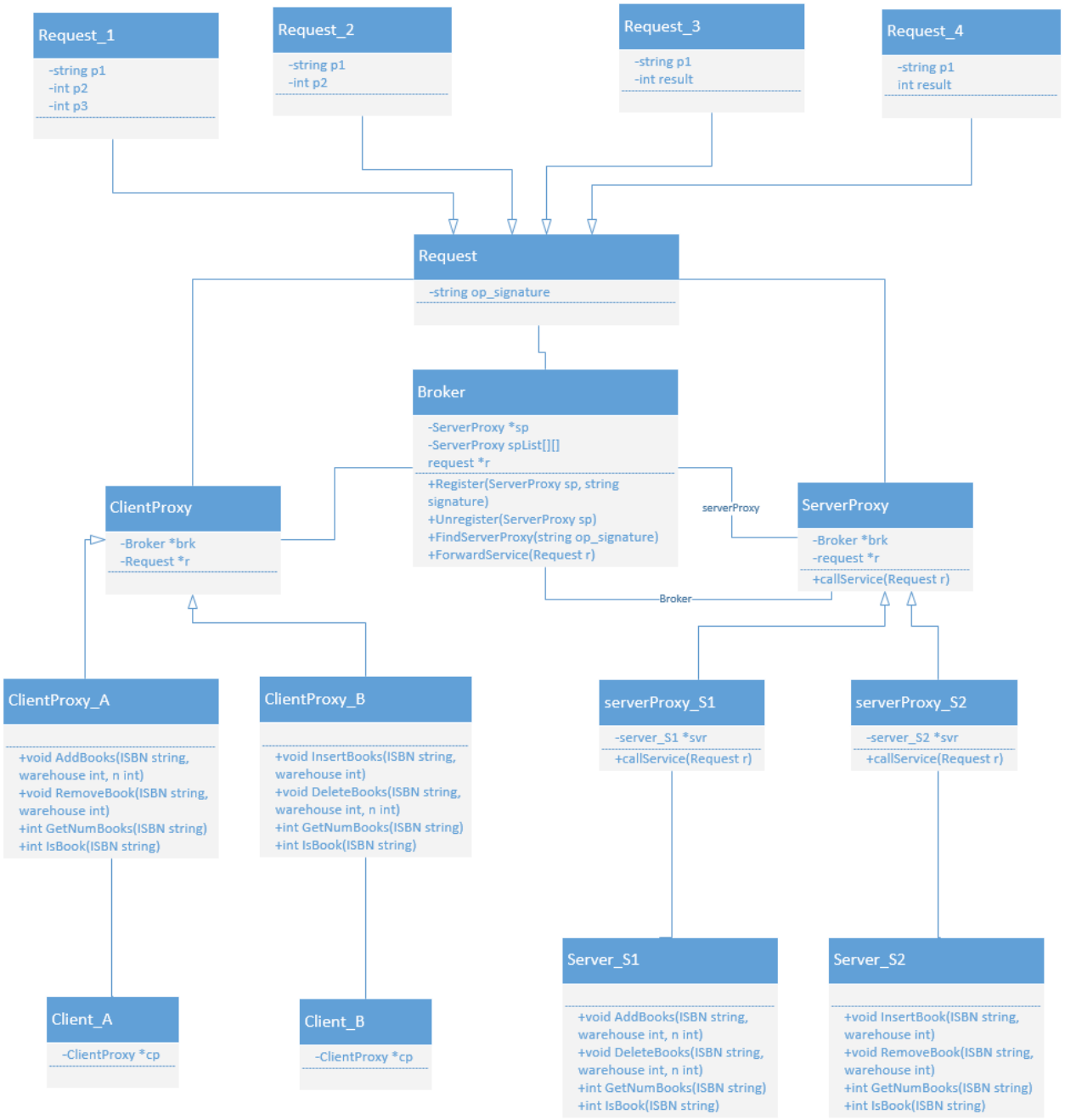
- A20391478

Table of Contents

Problem #1

Client-Broker-Server pattern

Class Diagram



Pseudo-code

```
Class Broker
{
        ServerProxy *sp
        ServerProxy spList[][]
        request *r

        //Functions
        Register(ServerProxy sp, string signature)
        {
                sp→ Serverproxy spList[] // add server to serverproxy spList
                add signature to serverproxy spList[]
        }
        unregister(serverproxy sp)
        {
                remove sp from serverproxy spList
        }
        FindServerProxy(string op_Signature)
        //For every server in the serverproxy spList
        {
                if serverproxy spList[op_signature]
                return Serverproxy
                end if
        }
        ForwardService(request r)
        {
                server proxy p1=Findserverproxy(r.op_signature)
                        if(p1!=null)
                                sp→callservice(r)
                        end if
        }

        Class clientproxy_A
        {
                void AddBooks(string ISBN,int warehouse, int n)
                r= Request_1
                r→op_signature = “void AddBooks(string,int,int)”
                r→p1=ISBN
                r→p2=warehouse
                r→p3=n
                b→ Forwardservice(r)
        }
        Void RemoveBook( ISBN, int warehouse)
        {
                r=Request_2
                r→op_signature=”Void RemoveBook(string,int)”
```

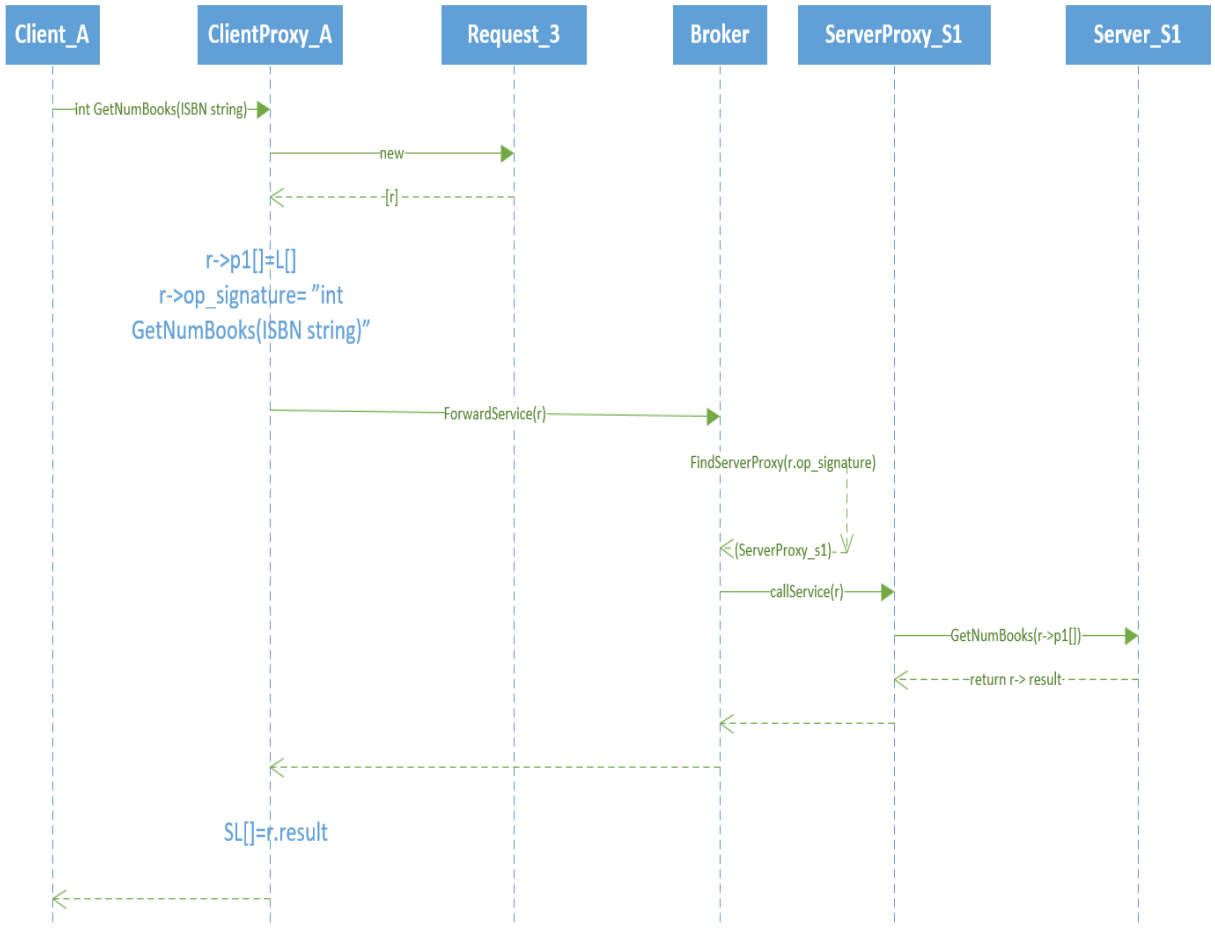
```
                r→p1=ISBN
                r→p2=warehouse
                b→ForwardService(r)
        }
        int GetNumBooks(string ISBN)
        {
                r=Request_3
                r→op_signature=”int GetNumBooks(string)”
                r→p1=ISBN
                b→ForwardService(r)
                return r→ result
        }

        int IsBook(string ISBN)
        {
                r=Request_4
                r→ op_signature= “int IsBook(string)”
                r→ p1= ISBN
                b→ ForwardService(r)
                return r→ result
        }
}

Class ServerProxy_S2
{
        Server_s2 *svr

        CallService(Request r)
                if(r→op_signature=”Void InsertBoot(string,int)”)
                        S2→ insert the book
                else if(r→op_signature=”Void RemoveBook(string,int)”)
                        S2→ remove the book
                else if(r→op_signature=”int GetNumBooks (string)”)
                        return r→ result = S2→ GetNumBooks(r.n)
                else if(r→ op_signature=”int IsBook(string)”)
                        return r→ result = S2→ IsBook(r.n)
}
```

Sequence Diagram

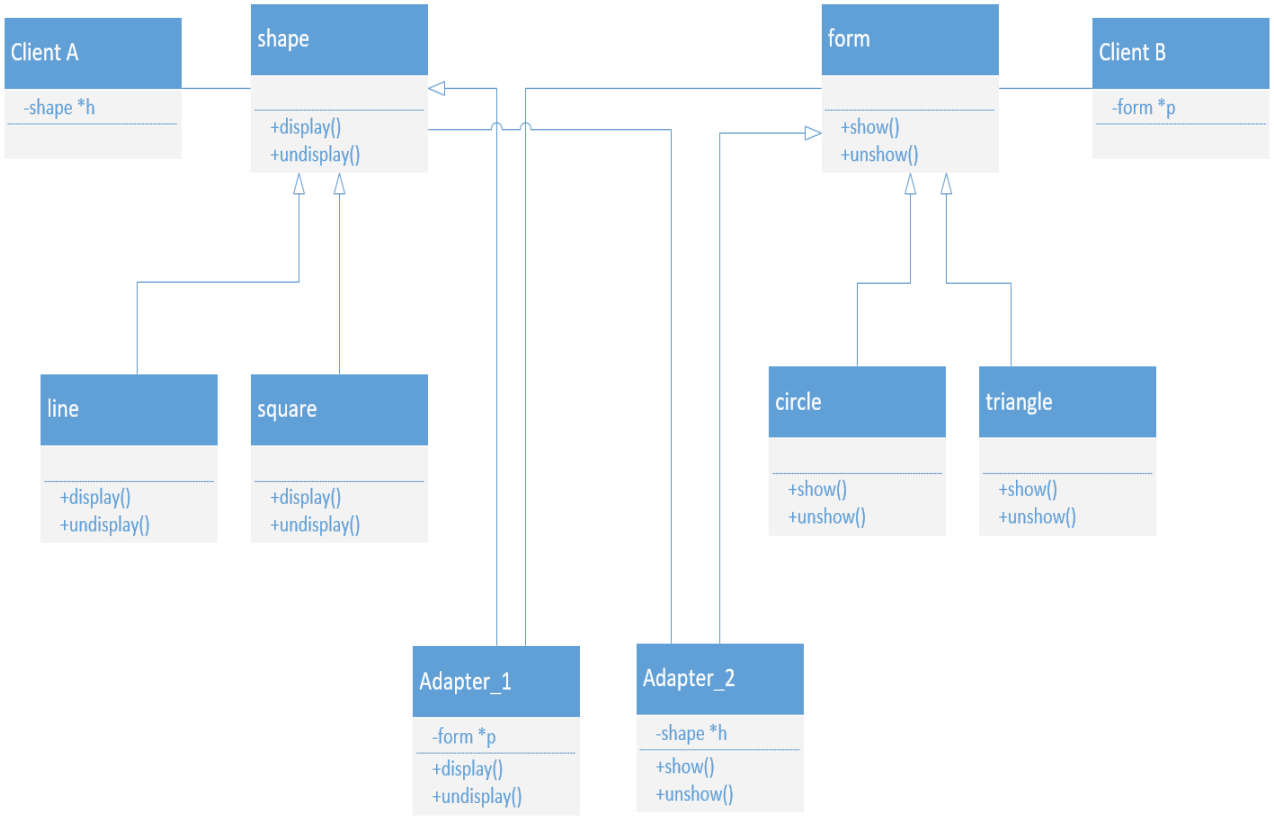


Problem #2

Adapter Design Pattern

Association-Based Version

Class Diagram



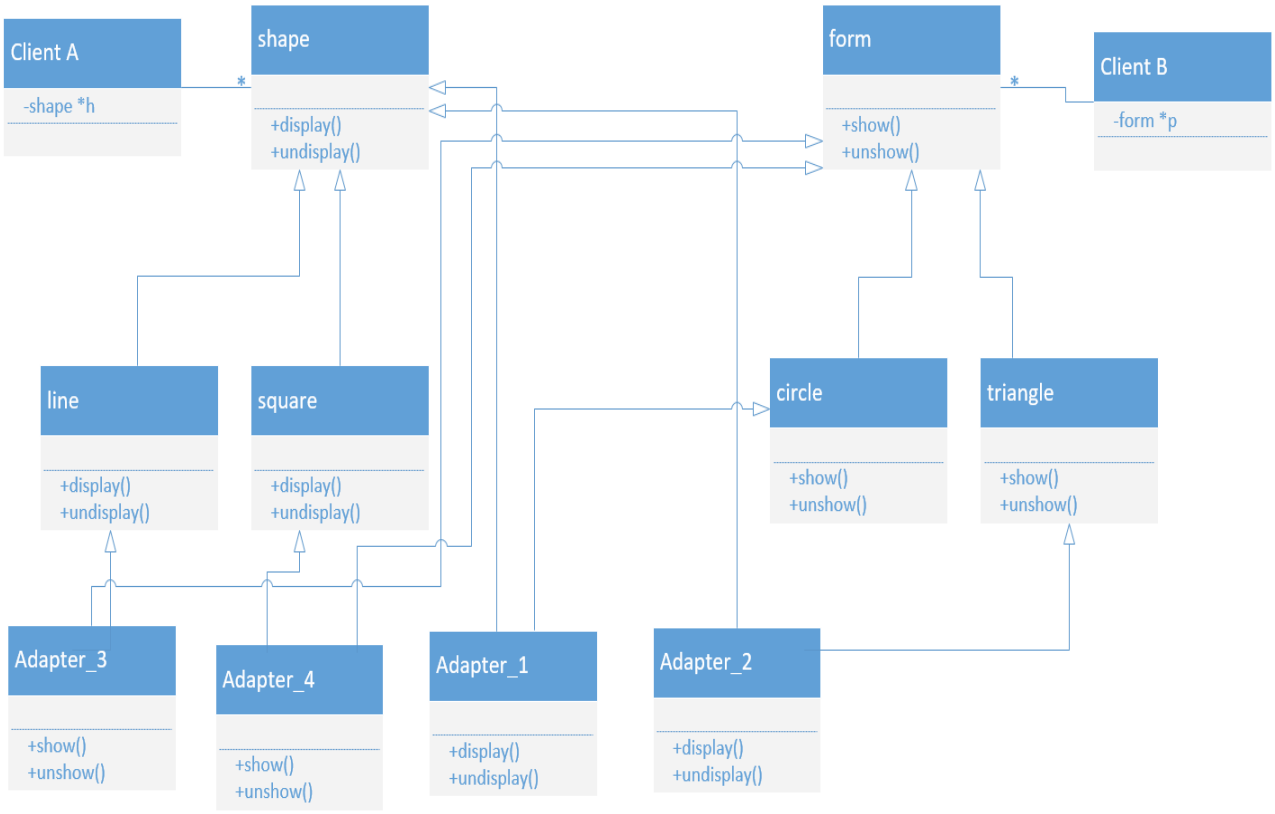
Pseudo-code

```
class Adapter_1
{
        Form *p
        display()
        {
                p→ show()
        }
        undisplay()
        {
                p→ unshow()
        }
}

class Adapter_2
{
        Shape *h
        show()
        {
                h→ display()
        }
        unshow()
        {
                h→ undisplay()
        }
}
```

Inheritance-Based Version

Class Diagram



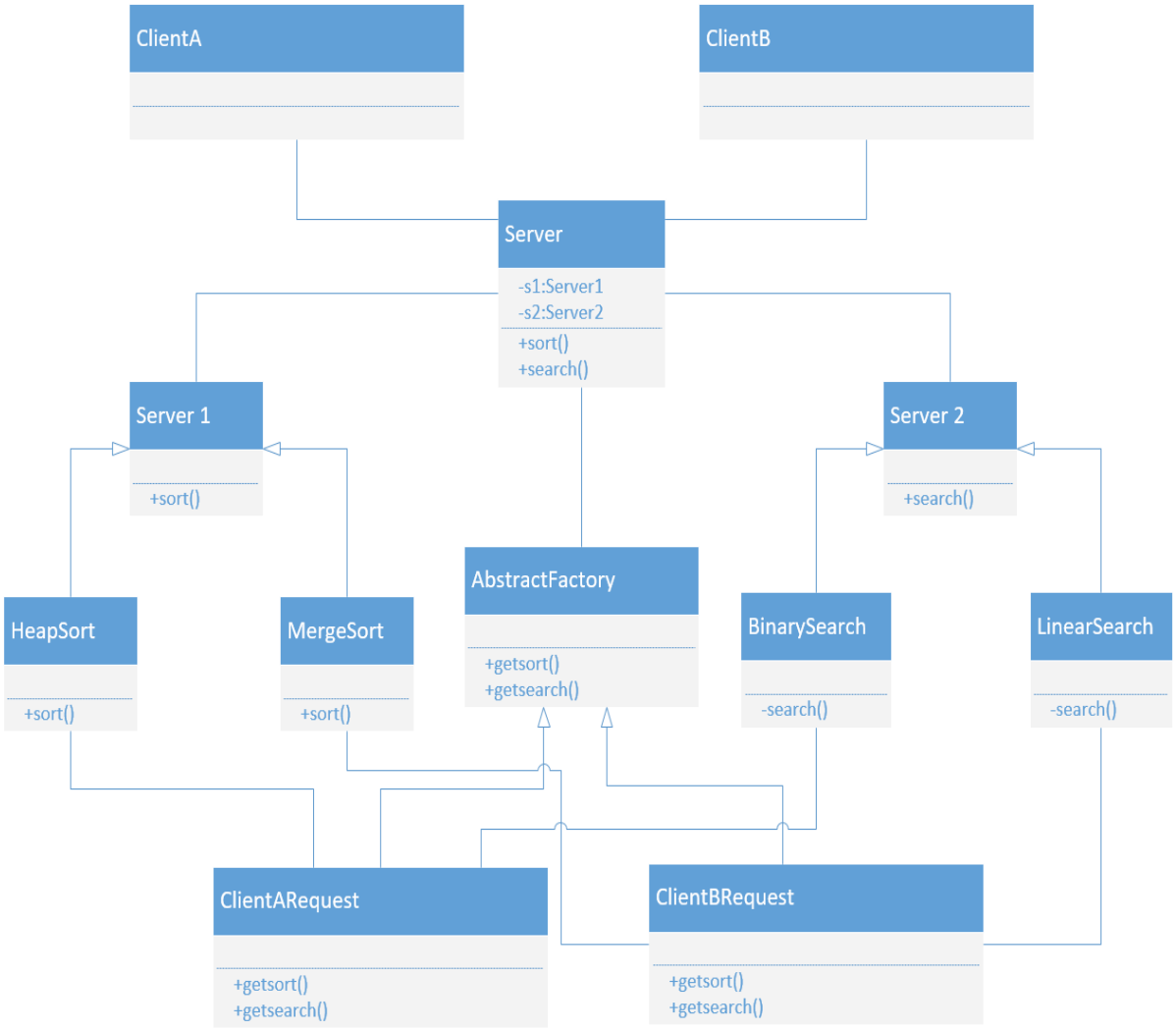
Pseudo-code

```
Class Adapter_1
{
        display()
        {
                show()
        }
        undisplay()
        {
                unshow()
        }
}
Class Adapter_2
{
        display()
        {
                show()
        }
        undisplay()
        {
                unshow()
        }
}
Class Adapter_3
{
        show()
        {
                display()
        }
        unshow()
        {
                undisplay()
        }
}
Class Adapter_4
{
        show()
        {
                display()
        }
        unshow()
        {
                undisplay()
        }
}
```

Problem #3

Abstract Factory Pattern

Class diagram



Pseudo-code

AbstractFactory

```
class clientARequest
{
        getsort()
        {
                return new heapsort // return object of HeapSort
        }
        getsearch()
        {
                return new binary search // return object of BinarySearch
        }
}

class clientBRequest
{
        getsort()
        {
                return new mergesort // return object of MergeSort
        }
        getsearch()
        {
                return new linearsearch //return object of LinearSearch
        }
}
```

Sequence Diagram