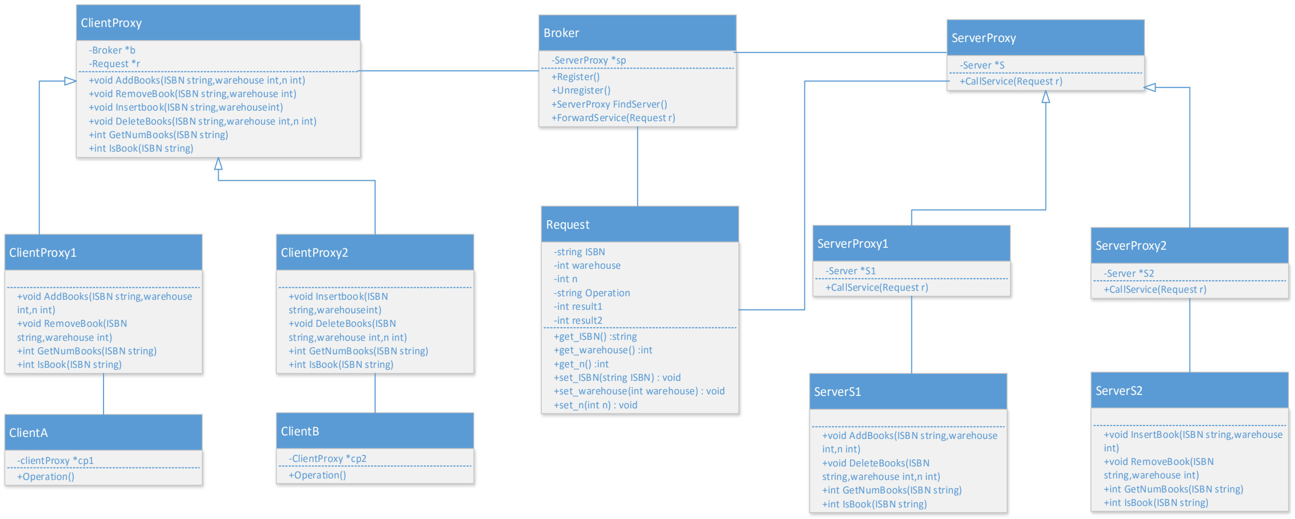
Homework 2

Question 1:


ClientProxy
-Broker *b
-Request *r
+void AddBooks(ISBN string,warehouse int,n int)
+void RemoveBook(ISBN string,warehouse int)
+void Insertbook(ISBN string,warehouseint)
+void DeleteBooks(ISBN string,warehouse int,n int)
+int GetNumBooks(ISBN string)
+int IsBook(ISBN string)
Broker
-ServerProxy *sp
+Register()
+Unregister()
+ServerProxy FindServer()
+ForwardService(Request r)
ServerProxy
-Server *S
+CallService(Request r)
ClientProxy1
+void AddBooks(ISBN string,warehouse int,n int)
+void RemoveBook(ISBN string,warehouse int)
+int GetNumBooks(ISBN string)
+int IsBook(ISBN string)
ClientProxy2
+void Insertbook(ISBN string,warehouseint)
+void DeleteBooks(ISBN string,warehouse int,n int)
+int GetNumBooks(ISBN string)
+int IsBook(ISBN string)
Request
-string ISBN
-int warehouse
-int n
-string Operation
-int result1
-int result2
+get_ISBN() :string
+get_warehouse() :int
+get_n() :int
+set_ISBN(string ISBN) : void
+set_warehouse(int warehouse) : void
+set_n(int n) : void
ServerProxy1
-Server *S1
+CallService(Request r)
ServerProxy2
-Server *S2
+CallService(Request r)
ClientA
-clientProxy *cp1
+Operation()
ClientB
-ClientProxy *cp2
+Operation()
ServerS1
+void AddBooks(ISBN string,warehouse int,n int)
+void DeleteBooks(ISBN string,warehouse int,n int)
+int GetNumBooks(ISBN string)
+int IsBook(ISBN string)
ServerS2
+void InsertBook(ISBN string,warehouse int)
+void RemoveBook(ISBN string,warehouse int)
+int GetNumBooks(ISBN string)
+int IsBook(ISBN string)

Pseudocode:

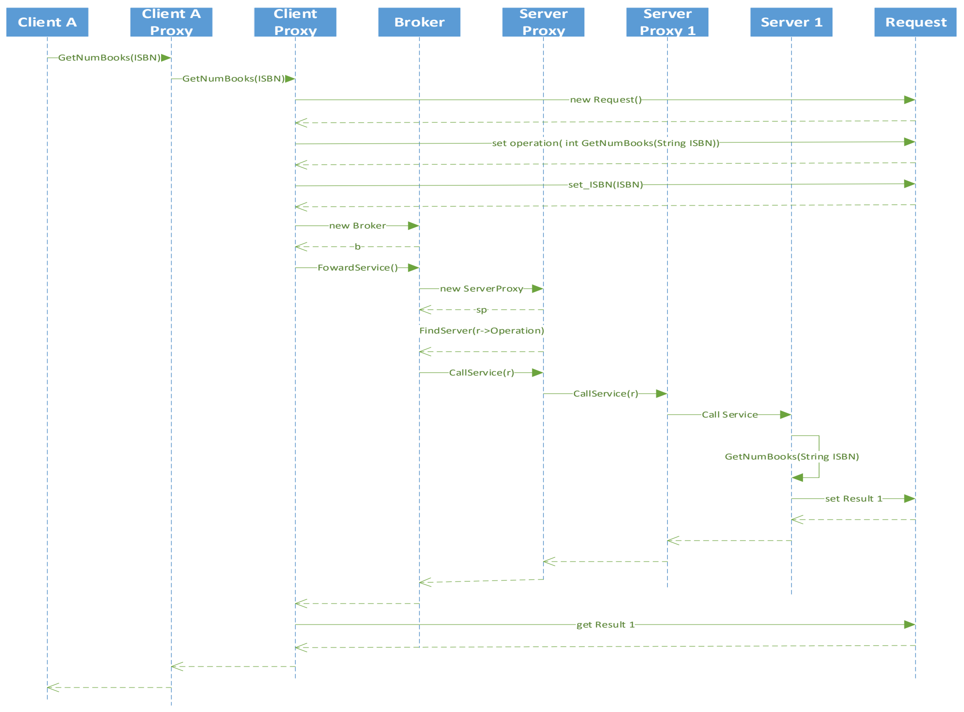
```
class clientA{
        operation(){
                if(cp1 != null){
                        ClientProxy1 cp1 = new ClientProxy1();
                }

                if(){
                        print(cp1->AddBooks(String ISBN, int warehouse, int n));
                }else if(){
                        print(cp1->RemoveBook(String ISBN, int warehouse));
                }else if(){
                        print(cp1->GetNumBooks(String ISBN));
                }
                else{
                        cp1->IsBook (String ISBN)
                }
        }
}
```

```
class clientB{
        operation(){
                if(cp2 == null){
                        ClientProxy cp2 = new ClientProxy();
                }

                if(){
                        print(cp2->DeleteBooks(String ISBN, int warehouse, int n));
                }else if(){
                        print(cp2->InsertBook(String ISBN, int warehouse));
                }else if(){
                        print(cp2->GetNumBooks(String ISBN));
                }
                else{
                        cp2->IsBook(String ISBN)
                }
        }
}

class ClientProxy1{
        int GetNumBooks(String ISBN){
                r = new Request();
                r->operation = "int GetNumBooks(String ISBN)";
                r->set_ISBN(this.ISBN);

                if(b == null){
                        Broker b = new Broker();
                }
                b->FowardService(r);

                return r->result1; //returns total no.of books
        }

        int IsBook(String ISBN){
                r = new Request();
                r->operation = "int IsBook(String ISBN)";
                r->set_ISBN(this.ISBN);

                if(b == null){
                        Broker b = new Broker();
                }
                b->FowardService(r);

                return r->result2;
                //returns 1, if a book exists; returns 0, otherwise
                if (found){
                        return 1;
                }
                else {
                        return 0;
                }
        }
        void AddBooks(String ISBN, int warehouse, int n){
                r = new Request();
                r->operation = "void AddBooks(String ISBN, int warehouse, int n)";
                r->set_ISBN(this.ISBN);
                r->set_warehouse(this.warehouse);
                r->set_n(this.n);

                if(b != null){
                        Broker b = new Broker();
                }
                b->FowardService(r);
        }

        void RemoveBook(String ISBN, int warehouse){
                r = new Request();
                r->operation = "void AddBooks(String ISBN, int warehouse)";
                r->set_ISBN(this.ISBN);
                r->set_warehouse(this.warehouse);

                if(b != null){
                        Broker b = new Broker();
                }
                b->FowardService(r);
        }
}

class ClientProxy2{
        int GetNumBooks(String ISBN){
                r = new Request();
                r->operation = "int GetNumBooks(String ISBN)";
                r->set_ISBN(this.ISBN);

                if(b == null){
                        Broker b = new Broker();
                }
                b->FowardService(r);

                return r->result1; //returns total no.of books
        }

        int IsBook(String ISBN){
                r = new Request();
                r->operation = "int IsBook(String ISBN)";
                r->set_ISBN(this.ISBN);

                if(b == null){
                        Broker b = new Broker();
                }
                b->FowardService(r);

                return r->result2;
                //returns 1, if a book exists; returns 0, otherwise
                if (found){
                        return 1;
                }
                else {
                        return 0;
                }
        }

        void DeleteBooks(String ISBN, int warehouse, int n){
                r = new Request();
                r->operation = "void DeleteBooks(String ISBN, int warehouse, int n)";
                r->set_ISBN(this.ISBN);
                r->set_warehouse(this.warehouse);
                r->set_n(this.n);

                if(b != null){
                        Broker b = new Broker();
                }
                b->FowardService(r);
        }

        void InsertBook(String ISBN, int warehouse){
                r = new Request();
                r->operation = "void InsertBook(String ISBN, int warehouse)";
                r->set_ISBN(this.ISBN);
                r->set_warehouse(this.warehouse);

                if(b != null){
                        Broker b = new Broker();
                }
                b->FowardService(r);
        }

}

class Broker{
        FowardService(Request r){
                if(sp == null){
                        sp = new ServerProxy();
                }
                if(sp != null){
                        sp->FindServer(r->Operation);
                        sp->callService(r);
                }
        }

        ServerProxy FindServer(String Operation){
                //Operations
                return ServerProxy;
        }
        Register(Server *s){
                sp.add(s);
        }

        Unregister(Server *s){
                Sp.remove(s);
        }

class ServerProxy{
        CallService(Request r){
        }
}

class ServerProxy1{
        CallService(Request r){
                if(r->Operation == "AddBooks(String ISBN, int warehouse, int n)"){
                        r->result1 = set(s->AddBooks(String ISBN, int warehouse, int n));

                }else if(r->Operation == "DeleteBooks(String ISBN, int warehouse, int n)"){
                        r->result2 = set(s->DeleteBooks(String ISBN, int warehouse, int n));
                }

                }else if(r->Operation == "int GetNumBooks(String ISBN)"){
                        r->result1 = get(s->GetNumBooks(String ISBN));
                }else (r->Operation == "int IsBook(String ISBN)"){
                        r->result2 = get(int IsBook(String ISBN));
                }
        }
}

class ServerProxy2{
        CallService(Request r){

                if(r->Operation == "RemoveBook(String ISBN, int warehouse)"){
                        r->result1 = set(s->RemoveBook(String ISBN, int warehouse));

                }else if(r->Operation == "InsertBook(String ISBN, int warehouse)"){
                        r->result1 = set(s->InsertBook(String ISBN, int warehouse));

                }else if(r->Operation == "int GetNumBooks(String ISBN)"){
                        r->result1 = get(s->GetNumBooks(String ISBN));

                }else (r->Operation == "int IsBook(String ISBN)"){
                        r->result2 = get(int IsBook(String ISBN));
                }
        }
}
```

```
class server1{

        void AddBooks(String ISBN, int warehouse, int n){
                //Add Operations

        }

        void DeleteBooks(String ISBN, int warehouse, int n){
                //remove Operations

        }

        int GetNumBooks(String ISBN){
                return;
        }
        int IsBook(String ISBN){
                return;
        }

}

class server2{

        void RemoveBook(String ISBN, int warehouse){

        }

        void InsertBook(String ISBN, int warehouse){

        }

        int GetNumBooks(String ISBN){
                return;
        }
        int IsBook(String ISBN){
                return;
        }
}

class Request{

        void set_ISBN(int ISBN){
                this.ISBN = ISBN;
        }

        int get_ISBN(){
                return ISBN;
        }

        void set_warehouse(int warehouse){
                this.warehouse = warehouse;
        }
```

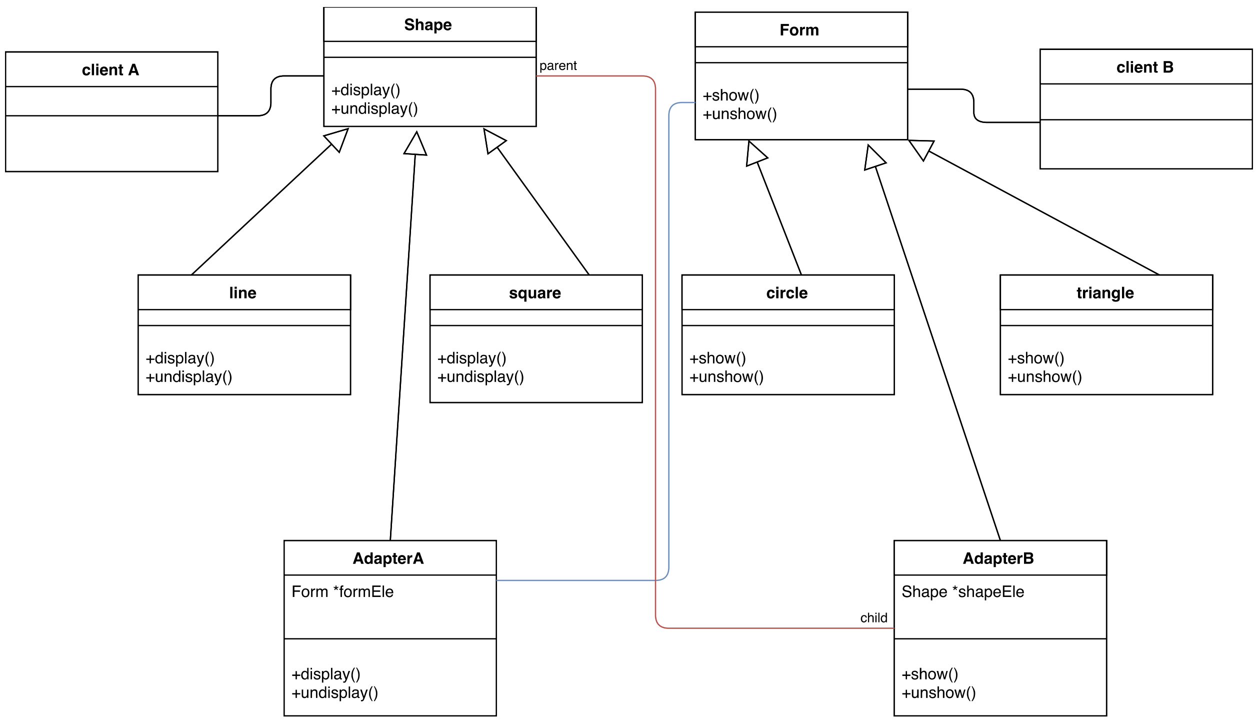
```
	int get_warehouse(){
		return warehouse;
	}

	void set_n(int n){
		this.n = n;
	}

	int get_n(){
		return n;
	}

	void set_result1(int result1){
		this.result1 = result1;
	}

	int get_result1(){
		return result1;
	}

	void set_result2(float result1){
		this.result2 = result2;
	}

	int get_result2(){
		return result2;
	}
}
```

Sequence Diagram:



Question 2:

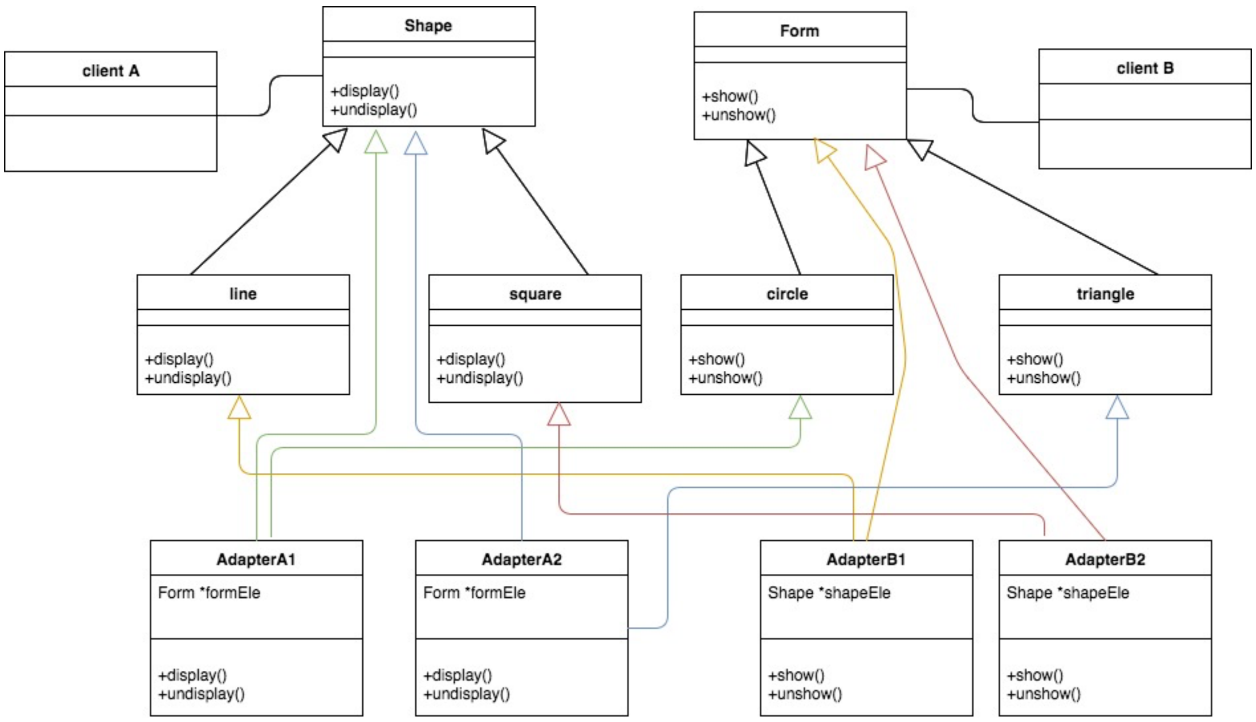
Association:



Pseudocode:

```
class Shape{
        display(){

        }
        undisplay(){

        }
}

class Form{
        show(){

        }
        unshow(){

        }
}

class line extends Shape {
        display(){

        }
        undisplay(){

        }
}

class square extends Shape {

        display(){

        }
        undisplay(){

        }
}

class circle extends Form {

        show(){

        }
        unshow(){

        }
}

class traingle extends Form {

        show(){

        }
        unshow(){

        }
}

class AdapterA extends Shape{
        Form *formEle
        display(){
                formEle->show();
        }

        undisplay(){
                formEle->unshow();
        }
}

class AdapterB extends Form{
        Shape *shapeEle
        show(){
                shapeEle->display();
        }

        unshow(){
                shapeEle->undisplay();
        }
}
```

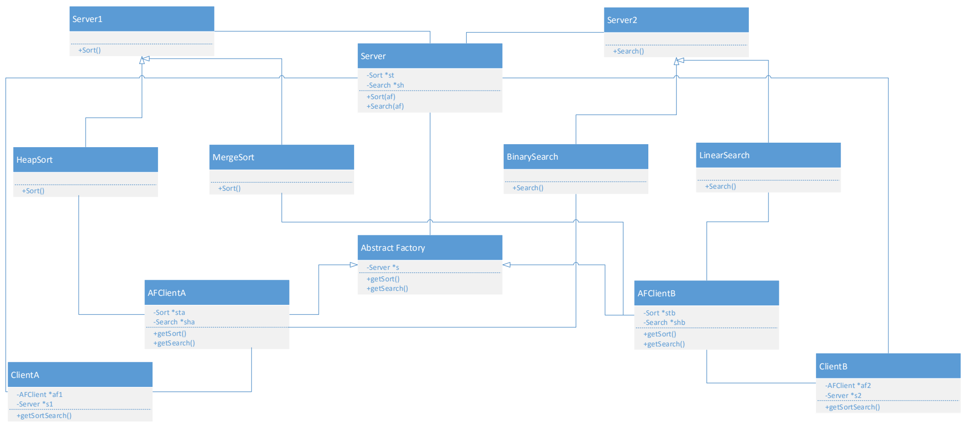
Inheritance:



Pseudocode:

```
class AdapterA1 extends Shape, Circle{
        display(){
                show()
        }

        undisplay(){
                unshow()
        }
}

class AdapterA2 extends Shape, Triangle{
        display(){
                show()
        }

        undisplay(){
                unshow()
        }
}

class AdapterB1 extends Form, Line{
        show(){
                display()
        }
        unshow(){
                undisplay()
        }
}

class AdapterB1 extends Form, Square{
        show(){
                display()
        }

        unshow(){
                undisplay()
        }
}
```

Question 3:

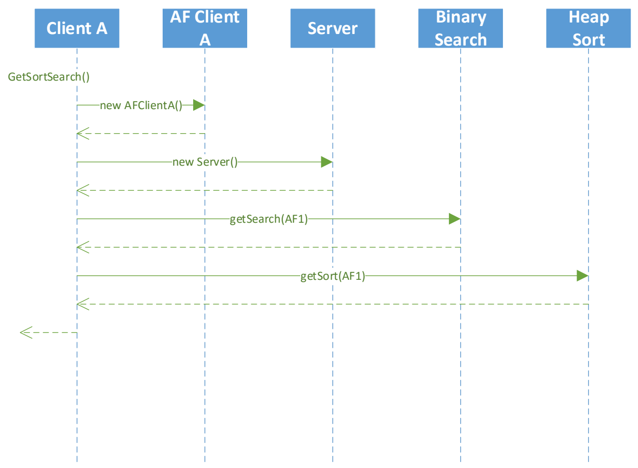


Pseudocode:

```
class AbstractFactory{
        getSort(){
        }
        getSearch(){
        }
}

class AFClientA extends AbstractFactory{
        getSearch(){
                if(sha == null){
                        sha = new BinarySearch();
                }
                sha.getSearch();
        }

        getSort(){
                if(sta == null){
                        sta = new HeapSort();
                }
                sta.getSort();
        }
}

class AFClientB extends AbstractFactory{
        getSearch(){
                if(sha == null){
                        sha = new LinearSearch();
                }
                sha.getSearch();
        }

        getSort(){
                if(sta == null){
                        sta = new MergeSort();
                }
                sta.getSort();
        }
}

class clientA{
        AFClientA *af1;
        Server *s1;
        af1 = new AFClientA();
        s1  = new Server();
        getSortSearch(){
                s1.getSort(af1);
                s1.getSearch(af1);
        }
}

class clientB{
        AFClientB *af2;
        Server *s2;
        af2 = new AFClientA();
        s2  = new Server();
        getSortSearch(){
                s2.getSort(af1);
                s2.getSearch(af1);
        }
}


class Server{
        Sort = new af.Sort();
        Search = new af.Search();
        time.getSort();
        date.getSearch();
}


class HeapSort extends Server1{
        //prints Heapsort
}

class MergeSort extends Server1{
        //prints Mergesort
}

class BinarySearch extends Server2{
        //prints BinarySearch
```

```
}

class LinearSearch extends Server2{
                //prints LinearSearch
}
```

Sequence Diagram:



Client A
AF Client A
Server
Binary Search
Heap Sort
GetSortSearch()
new AFClientA()
new Server()
getSearch(AF1)
getSort(AF1)