HOMEWORK ASSIGNMENT #2

CS 586; Fall 2025
Due Date: **October 16, 2025**
Late homework: 50% off
After **October 21,** the homework assignment will not be accepted.

This is an **individual** assignment. **Identical or similar** solutions will be penalized.

Submission: All homework assignments must be submitted on the Blackboard. The submission **must** be as one PDF file (otherwise, a 10% penalty will be applied).

PROBLEM #1 (40 points)
There exist two servers **S1** and **S2**. Both servers support the following services:

Services supported by **server-S1**:
void Service1(string, int, int)
void Service2(string, int, int)
int Service3(string)
float Service4(string)

Services supported by **server-S2**:
void Service1(string, int)
void Service2(string, int)
int Service3(string)
float Service4(string)

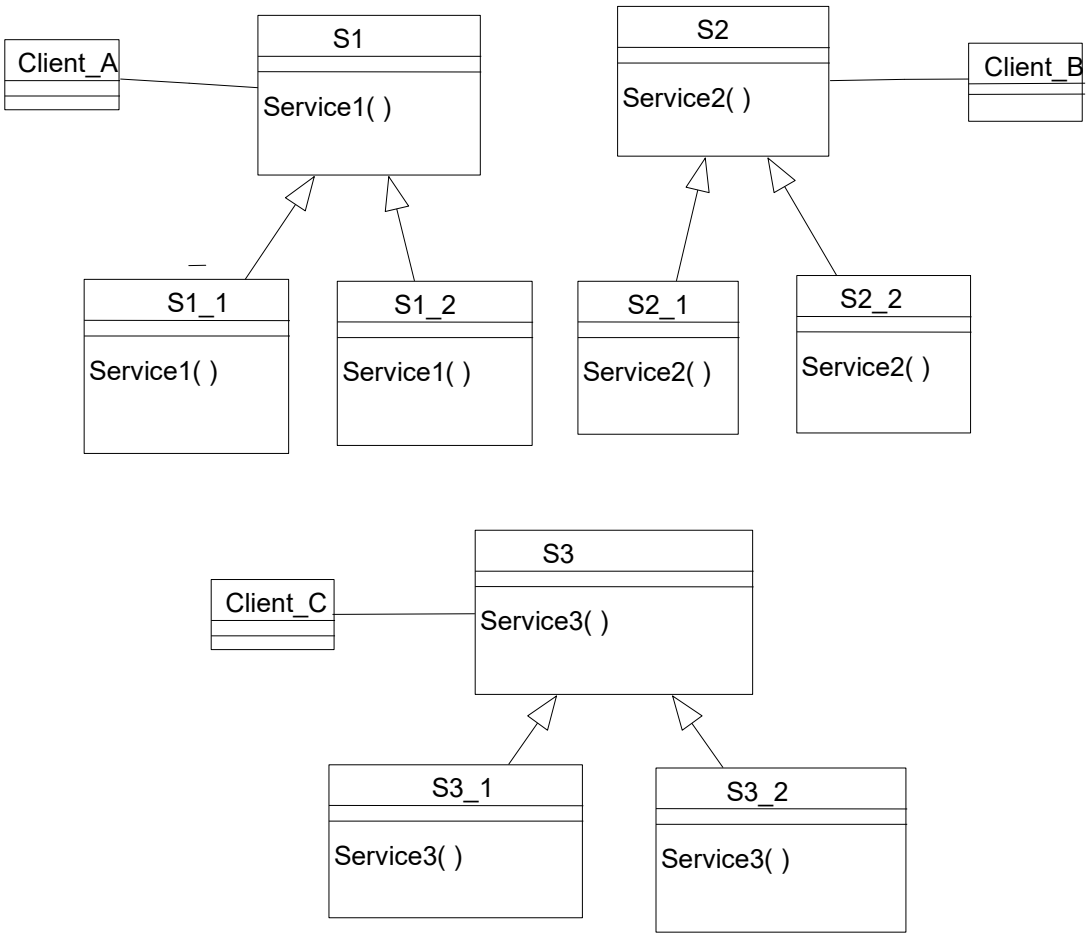
There exist two client processes, and they request the following services:

Client-A	**Client-B**
void Service1(string, int, int)	void Service1(string, int)
void Service2(string, int)	void Service2(string, int, int)
int Service3(string)	int Service3(string)
float Service4(string)	float Service4(string)

The client processes do not know the location (pointer) to servers that may provide these services. Devise a software architecture using a **Client-Broker-Server** architecture for this problem. In this design, the client processes are not aware of the location of the servers providing these services.

- Provide a class diagram for the proposed architecture. In your design, all components should be **decoupled** as much as possible.
- Provide the **pseudocode** for all operations of the following components/classes:
  - Broker
  - Client Proxy of *Client-A*
  - Server Proxy of *Server-2.*
- Provide a sequence diagram to show how *Client-A* gets "void *Service2(string, int)* service.

PROBLEM #2 (25 points)



A design of a system is shown above. In this system, *Client_A* uses objects of classes *S1_1* and *S1_2*, *Client_B* uses objects of classes *S2_1* and *S2_2*, and *Client_C* uses objects of classes *S3_1* and *S3_2*.

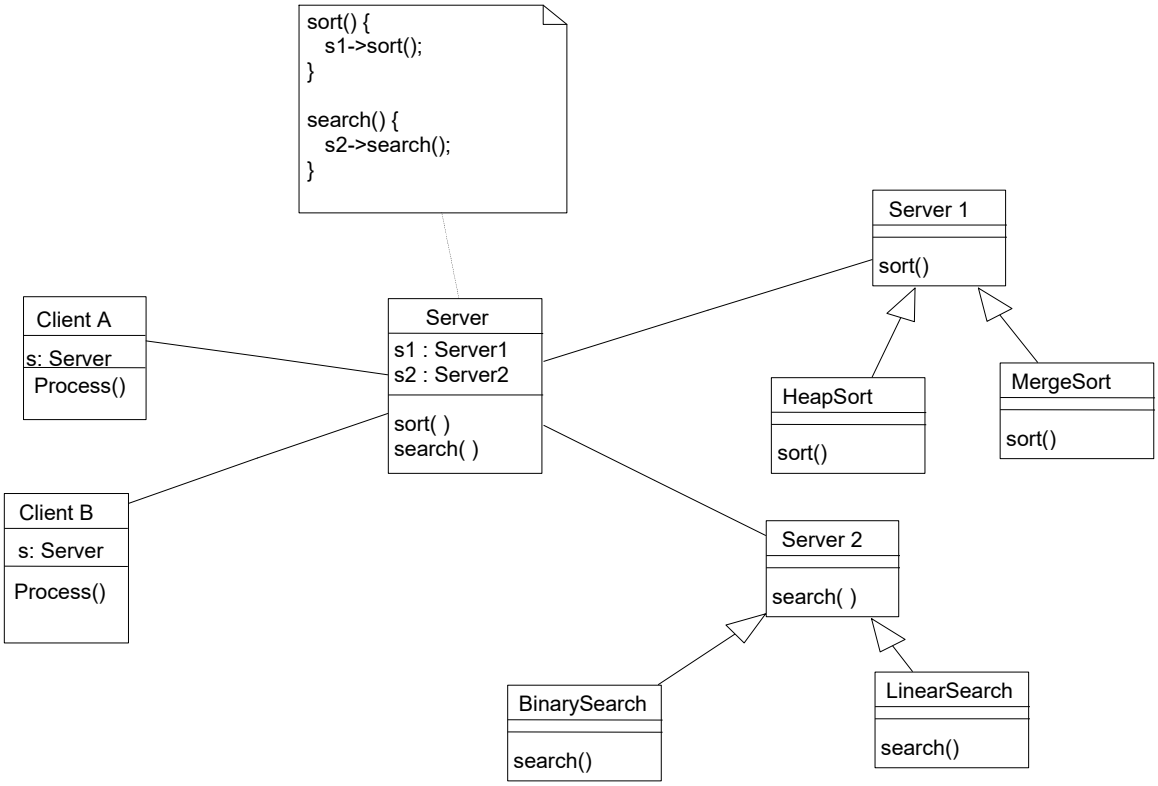
Client_A would like to use objects, operations *Service2()*, of classes *S2_1* and *S2_2* by invoking operation *Service1()*. *Client_C* would like to use objects, operation *Service1(),* of classes *S1_1* and *S1_2* by invoking operation *Service3()*. In addition, *Client_B* would like to use objects, operation *Service3(),* of classes *S3_1* and *S3_2* by invoking operation *Service2()*.

Provide a design with **minimal** modifications to the existing system using the **Adapter design pattern** in which (1) *Client_A* can use objects of classes *S2_1* and *S2_2* by invoking operations *Service1*(), (2) *Client_B* can use objects of classes *S3_1*, and *S3_2* by invoking operations *Service2(),* and (3) *Client_C* can use objects of classes *S1_1*, and *S1_2* by invoking operations *Service3()*. Notice that none of the classes shown in the above class diagram should be modified. Provide two solutions that are based on:

1. An **association-based** version of the Adapter pattern
2. An **inheritance-based** version of the Adapter pattern

Provide a class diagram for each solution. You do not have to provide any description for classes/operations of the above class diagram (only new classes/operations should be described using **pseudo-code**).

PROBLEM #3 (35 points)
Consider a design of the system shown below:



Client A and *Client B* get services, *sort()* and *search()*, directly from the *Server*. However, the *Server* gets the appropriate services from *HeapSort* or *MergeSort* servers for *sort()* service. In addition, the *Server* gets the appropriate services from *BinarySearch* or *LinearSearch* servers for *search()* service.

In this design, clients do not have control where the services are coming from. However, *ClientA,* by invoking *sort()* and *search()* on the *Server,* would like to get *sort()* service from *HeapSort* server and *search()* service from *BinarySearch* server. On the other hand, *ClientB,* by invoking *sort()* and *search()* on the *Server,* would like to get *sort()* from *MergeSort* server and *search()* from *LinearSearch* server. Notice that the current design does not support this option.

Use the **abstract factory** design pattern to solve this problem. In your solution, the *Client* classes should be completely **decoupled** from the issue of invoking services by the *Server* for appropriate versions of *sort()* and *search()*. Notice that none of the classes (and operations) shown in the above class diagram should be modified. However, new operations/classes can be introduced.

- Provide the class diagram and briefly describe the responsibility of each class and the functionality of each operation using **pseudo-code**. In your design, all components should be **decoupled** as much as possible. You do not have to provide any description for classes/operations of the above class diagram (only new classes/operations should be described).
- Provide a sequence diagram to show how *ClientB* gets *sort() service* from *MergeSort* server and *search()* service from *LinerSearch* server by invoking *sort()* and *search()* on the *Server*.